Name: ______________________ **Student ID #:** ______________________

Lab #3: COMP 3000 (Operating Systems)

Please answer all questions below. This lab is intended to be an introduction to the Linux kernel and provides background which will be required in order to complete lab 4. 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. Because of the hard drive requirements for this lab, the SCS network is not a suitable environment for working on the lab.

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

In this lab, you will be building and using a Linux kernel. The kernel will be run in a machine emulator called `qemu`. All work will be done with a copy of the files contained in `/lab-data`. You need to create working copies of all the files in the `/lab-data` directory by performing the following steps:

1. Create your own working directory using the `mkdir` command. Name the directory after your username.
2. Change into that directory.
3. Copy all files in `/lab-data` into your working directory by using `cp` with the `-a` option.
4. The rest of this lab assumes you are working from within your local working directory.

1.1 Virtual Machine Emulators

This lab uses a virtual machine emulator to allow you to modify and run a copy of the Linux operating system without destroying the lab computers. For this and the next lab, we will be running the emulator `QEMU`.

Most system emulators use a file to store a disk image. That is, a large file on disk is represented as a hard drive in the emulated system environment. The operating system which is emulated does not have any access to files outside of its disk image. `QEMU` follows this convention with the exception that the Linux kernel which is booted does not have to be stored inside the disk image. This allows us to edit and compile the kernel without the overhead of emulation while testing the modified kernel under the emulated system. For this and the next lab, we will be modifying the Linux kernel, version 2.6.19.1.

Run the pre-built Linux kernel in the emulated environment by running the `runme.sh` file contained in your working directory. The login to the emulated environment is root with no password (If you ever damage your emulated disk image, you can simply copy it across again from the clean copy located at `/lab-data`).

1. **[2]** By using the `lspci` command, determine what PCI bus hardware is being emulated. Specifically, what video card does `QEMU` report to the operating system?

2. **[2]** Compare the output of the `lspci` on the emulated vs the real system. What is one piece of hardware available on the real system that is not available on the emulated system?

1.2 The Linux Kernel

1. **[2]** The Linux kernel is stored in the `linux-2.6.19.1` directory. What command line can be given to configure the kernel with a random assortment of configuration options?

2. **[2]** Using the configuration command `make menuconfig`, browse through the menus until you find the option for *Magic SysRq Key*. What menu is the option contained in?

3. **[2]** Enable support for the ISA SoundBlaster 16 card under ALSA (Advanced Linux Sound Architecture) and rebuild the kernel using `make`. Where did you find the option for the SoundBlaster 16 PnP ISA card?

4. **[2]** Run the new kernel you built with SoundBlaster support. The kernel will now detect a sound card on the emulated system. The command `dmesg` will display the debug messages output while the system is booting. What IRQ is the SoundBlaster device tied to?

2 Part B

2.1 Virtual Machine Emulators

1. **[5]** This lab used `QEMU` as a virtual machine emulator to allow us to build, modify, and debug a Linux kernel without having to reboot the real machine. Very briefly, how does a virtual machine emulator work (i.e. what does it do)?

2. **[10]** Using the documentation for `QEMU` (and the documentation for `GDB`) (both of which are available online), answer the following questions:

   (a) **[2]** What function does the kernel hang in when you shut down your emulated kernel? To get the kernel into the *hung* state, run the kernel in the emulator, log in as root, and then type `halt`. Wait for the *System halted* message.

   (b) **[4]** What option did you have to pass into `qemu` to enable gdb debugging of the running kernel?

   (c) **[4]** What command did you have to give to the gdb command line to debug the kernel running in `qemu`?

3. **[5]** By reading *QEMU, a Fast and Portable Dynamic Translator* by Fabrice Bellard [1], determine why QEMU is capable of running quickly compared to other emulators like Bochs[2].

[1]http://www.usenix.org/publications/library/proceedings/usenix05/tech/freenix/bellard.html
[2]http://bochs.sourceforge.net/

2.2 The Linux Kernel

In this section, you will modify the Linux kernel. If you end up breaking the kernel beyond repair, you can always retrieve a fresh copy from the `/lab-data` directory.

1. **[10]** By using any available resource, determine why operating systems are almost always traditionally written in C (hint: examine the original design purpose and use of the C language). Your answer should be about a page long and cite at least 2 reasons. Be sure to cite appropriate references.

2. **[10]** Modify the system kernel to print *Bye for now.* after displaying the *System halted* message. In addition, have it print the system state by calling `show_state`. The files *arch/i386/kernel/reboot.c* and *kernel/printk.c* may be useful. What function did you modify? What process is running?