Listen

COMP 3430 – Operating Systems

Lab 1 – UNIX primer

Week of May 13th, 2019

Description

Lab 1 will (re)familiarize you with the CS UNIX facilities. You will be completing all of your work *remotely* – that is, you're going to be sitting physically at one machine, but completing all of your work at a *different* machine.

You will be required to submit your lab work electronically. Questions are to be answered in a Markdown formatted text file that will be submitted using the `handin` command. Questions to be answered are highlighted **in bold**.

Steps

Logging in

Everything you're going to be doing in this course will be completed on a Linux machine. You're welcome to do development on *any* Linux machine, but you **must** ensure that your code for labs and assignments compile on the CS Linux machines.

Log in to a CS Linux machine (aka, `aviary.cs.umanitoba.ca` – "the bird lab") using SSH. Note the name of the machine you are logged into. The CS Linux lab is E2-468. You are welcome to visit this lab at any time. (Tip: at *any* CS UNIX machine, run `combos` on the command line to see which rooms you have access to.)

Information about programs running on the system

In this part of the lab, you'll be spending some time learning about how to query the running OS for information about itself and the programs that are running on it. Complete the following steps:

1. All variants of UNIX/Linux operating system provide some version of the `ps` (process status) command. Type `ps` into the terminal (after you've `ssh`-ed into `aviary`) and find the PID (process ID) of your shell. You should also take the time to learn about some of the options that `ps` has, take a look at the `man` page for `ps` (type `man ps`). When you type `ps` it only shows *your* running programs – **How would you list *all* running programs and the usernames that are running them?**
2. One UNIX philosophy that you may not have encountered before is the idea that "Everything is a file". UNIX/Linuxes *also* provide information about running programs as files that you can read with standard tools. List the contents of the `/proc` directory (that is, type `ls /proc` into your terminal). Each process running on the machine has a numerically named directory in the listing (the number is the same as the process ID). Your shell process has a directory. Examine the status of your shell process by examining the contents of its status file (for example, if your shell process has PID 9827, use `less /proc/9827/status`). What is the state of your shell? **Can you explain why it is in this state?**
3. A process can examine its own status, without knowing its PID, by using the self directory, which is a link to the currently running process. Type the command `less /proc/self/status`. **What is the name of the currently running command, and what is its state?**

Information about the system

Download the file `os_info.c`. This is a C program that reads the files in `/proc/sys/kernel` and prints out information about the running OS.

You're going to need to get this onto the machine you're connected to in `aviary`. You may either:

1. `scp` the file from the `~/Downloads` folder of the machine your currently on (take a look at `man scp` for help), or
2. `wget` the file *on* the machine in `aviary` (run `wget https://www.cs.umanitoba.ca/~comp3430/code/labs/lab1/os_info.c`).

After you've got the code on your Linux machine, compile and run the program. You may use any compiler to compile the program.

Next, add new code to your program such that it will print the name of the computer (the host name), then compile it and run again. You can get the host name from another file located in `/proc/sys/kernel` (look around at the contents of the files). Tip: you can use the `grep` command to help you find out which files contain a certain string of text.

You're welcome to use any text editor to modify the code, but using `vi` on the remote machine is **strongly** recommended. Download a cheat sheet or complete a tutorial for `vim` if you are unfamiliar with how to use it.

Include in your *modified* `os_info.c` in your hand in.

Debugging 🐜 code

Download `broken.c`, and debug the file using `gdb` or `lldb`. Show the fixed file to the TA, and **include your *fixed* file in your hand in**. `broken.c` is also available at `https://www.cs.umanitoba.ca/~comp3430/code/labs/lab1/broken.c`.

Handing in your lab

When completed, use the `handin` command. Make sure that your deliverable includes a `README` and a `Makefile`. The `Makefile` should build your two programs, and the `README` should *minimally* explain how to invoke `make` to build your files.

```
handin 3430 lab1 whatever_my_lab1_folder_is
```

You must `handin` your work before Friday, May 17th @ 4:30pm. Late submissions will not be accepted.