**CSE 278 – Computer Architecture**

**Exercise #1**

Max Points: 10

**Note: If you are using your personal machine then prior to commencing work on this exercise, you may need to install XMing, Putty, and WinScp as illustrated in LinuxEnvironment.pdf (in the Readings folder).**

|  |
| --- |
| **You should save/rename this document using the naming convention MUid\_Exercise1.doc (example: alomarhw\_Exercise1.doc).**  **Objective**: The objective of this exercise is to gain some familiarity with accessing and working on a Linux machine while learning some Linux concepts.  Fill in answers to all of the questions. For almost all the questions you can simply copy-paste appropriate text from the shell/PuTTY window into this document. You are expected refer to [LinuxEnvironment.pdf](https://niihka.miamioh.edu/access/content/group/c5f42e94-d8a3-4b74-a6f8-2c8fc3f903c1/Handouts _ Video Tutorials/LinuxEvironment.pdf) document available in Readings folder on Canvas. You may discuss the questions with your instructor (preferably only when all else fails as this is a learn-by-doing style exercise). |

|  |  |
| --- | --- |
| **Name:** | **Samuel Mallamaci** |

1. Wait for your instructor to cover the administrative preliminaries and introduce the class to the basics of working in the new Linux laboratory environment.
   1. When you log onto a Linux machine in the lab, it is best for you to start a Terminal. In the terminal you will be presented with a shell (**$**) prompt. You need to perform various tasks by typing commands at the shell prompt and pressing the enter () key.

|  |  |
| --- | --- |
|  | **Note the following important details:**   * Ensure you use the correct path to save your files. Otherwise you will not find your files on other machines. * You need to be on Miami VPN in order to access the machines from off-campus. Using VPN is mentioned in R-Jan-28’s slides. * Off-campus graphics experience may vary. Consequently, it is important to get used to both the text and graphical modes of emacs. |

1. Log out of the Linux machine (if you are still logged-in) and repeat the login process to ensure you are familiar with the process.
2. When you start a Terminal or log onto the Linux server, you will start off in a default directory called your **home** directory. You should create all your files and save your work off the appropriate sub-directories (as directed by your instructor) under your home directory. To figure out what your home directory is, you need to use the pwd (*print working directory*) command (that is, type pwd at the shell ($) prompt and press enter key, which is indicated by ) as shown below:

* pwd 

|  |  |
| --- | --- |
| What is your home directory: | /home/mallamsg |

Note: The home directory will be similar to /home/alomarhw. The home directory changes for every user and could be at different locations on different machines or operating systems.

1. Briefly describe what operations the following commands do (refer to [LinuxEnvironmet.pdf](https://niihka.miamioh.edu/access/content/group/c5f42e94-d8a3-4b74-a6f8-2c8fc3f903c1/Handouts _ Video Tutorials/LinuxEvironment.pdf) on Canvas):
   1. **mkdir**:

|  |
| --- |
| This creates a new directory inside your current directory |

* 1. **cd**:

|  |
| --- |
| This changes the current working directory to the new one specified |

1. Using the above two commands, perform the following operations:
2. Using the pwd command, verify you are in your home directory.
3. Using the cd command switch to the location where your files are stored on the M drive (so that the files are accessible from any machine). For example:

|  |
| --- |
| $ cd /home/alomarhw/ |

1. Create a working directory called cse278. You will be using the cse278 directory for performing various lab exercise and homework exercise in the future.
2. Now change directory to the newly created cse278 directory.
3. Use the pwd command to verify that you are within the newly created directory.
4. What is the output you get when the following sequence of commands are typed and executed at the shell ($) prompt?

$ cd ~/ 

$ pwd 

|  |  |
| --- | --- |
| **Output:** | Home/mallamsg |

1. In your own words, explain what you think the above commands are achieving?

|  |
| --- |
| The above commands are changing the working directory back to the default home directory |

1. Based on the information from the man(ual) page for the ls command (type man ls at the shell/terminal $ prompt) briefly describe what the command line option “-l” (ell) does (refer to the LinuxEnvironment.pdf for syntax on how to use the man command at the shell prompt).

|  |
| --- |
| Ls -l prints the files in the directory in a list with detailed information on the file permissions and modification information |

1. Now let’s find out some details on the CPU for the Linux machine we are logged onto. In Linux, almost all of the system information is made available by the kernel via a virtual file system called proc. Information regarding all the CPUs/cores (computers may have multiple processors and each processor may have multiple cores). For this view the file /proc/cpuinfo using the less command (refer to LinuxEnvironment.pdf for details on less command) as shown below:

|  |
| --- |
| **$** less /proc/cpuinfo  |

NOTE: Use arrow keys to navigate information displayed and press q to quit out of less.

Using the output from the above command answer the following questions:

|  |  |  |
| --- | --- | --- |
| 1. | What is the vendor\_id value(s): | GenuineIntel |
|  |  |  |
| 2. | What is the model name of CPU(s): | Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz |
|  |  |  |
| 3. | What is the CPU speed in MHz: | 2399.99 MHz |
|  |  |  |
| 4. | What is the cache size (in KB) | 20480 Kb |
|  |  |  |

1. Determine some basic information about the operating system by typing the command “uname –rs” and noting the name of the operating system (first word) and version of the kernel (second word).

|  |  |  |  |
| --- | --- | --- | --- |
| OS Name: | Linux | Kernel Version: | generic |

1. Now use the **ps** command to get a snapshot of the process running on the machine by typing ps -fe at the shell ($) prompt (and press enter key). The output from the ps command will be in the form of columns where each row corresponds to a unique process running on the machine. The first 3 columns correspond to: user-id (login id), PID (process id) and PPID (parent process id). The last column (CMD) corresponds to the actual command being run. What is the PID and PPID for the "ps -fe" command run by you (take care that there are multiple users running the same command. Ensure you are looking at the right one). If the number of processes are too large and they scroll across your screen use the command ps –fe | less to see page-by-page of the output (press spacebar to scroll to next page and q to quit)

|  |  |  |  |
| --- | --- | --- | --- |
| PID: | 29370 | PPID: | 19843 |

1. Now inspect the above output from ps to locate the process entry corresponding to the PPID of the ps -fe command you ran. In other words, you need to locate the line of output whose PID is the same as the PPID value you noted in earlier question. Once you have located the appropriate line, fill in the following information:

|  |  |  |  |
| --- | --- | --- | --- |
| PID: | 19843 | PPID: | 19842 |

|  |  |
| --- | --- |
| CMD: | -bash |

1. Similar to previous question, starting all over with the ps –fe command iteratively use the PPID value corresponding to each process and locate the process ID corresponding to it. In other words, walk up the process hierarchy tree using the PID and PPID values until the PPID value is 0 (zero). List the sequence of processes you traverse in the table below (add more rows to the table as needed):

|  |  |  |
| --- | --- | --- |
| ***PID*** | ***PPID*** | ***CMD*** |
| 31223 | 19843 | Ps -fe |
| 19843 | 19842 | -bash |
| 19842 | 19793 | Sshd: mallamsg@pts/14 |
| 19793 | 6507 | Sshd: mallamsg [priv] |
| 6507 | 1 | /usr/sbin/sshd -D |
| 1 | 0 | /sbin/init |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

1. The ps command provides a one-time snapshot of the processes running on the machine. Alternatively you may use the **top** command to obtain a constantly refreshing list of processes. Type top at the shell ("$") prompt (and press enter key). The top command will run and show all process running on the machine. Try to see if you can use the help on top (or man page for top) to list only the processes run by you in the top display. Indicate the command in top that you used to list only the processes run by you below:

|  |  |
| --- | --- |
| top Command: | Top -u mallamsg |

1. Often you may find yourself running a program that has a bug in it causing it to get stuck in an infinite loop. In such cases, you will need to forcibly abort the process using the kill command. In order to experiment with the kill command perform the following steps:
2. Start emacs (as background process) by typing emacs**&** (don’t miss the ampersand at the end) at the shell prompt. emacs is a general purpose text editor that we will be using to code programs in this course. This command should start emacs and you should get your shell prompt back (you will be able to type more commands while emacs is running in the background). When emacs starts up you should see a graphical window popup on your machine.
3. Using the ps command figure out the PID for your emacs process.
4. Now use the kill command to terminate emacs by typing **kill <pid>** (where <pid> is the PID for emacs you determined in the previous sub-step) at the shell prompt. Note if emacs gets terminated. Note that sometimes you may have to force the process to shut down by using the command **kill -9 <pid>**.
5. In most operating systems (including: Linux, Unix, and Windows) processes can be started in the foreground or in the background. Foreground processes run in a blocking fashion, that is, the shell will wait until the process is completed. On the other hand, the shell does not wait for background processes to complete but continues to operate concurrently with other processes. In Linux (and Unix) foreground processes are started by simply typing in the command at the shell prompt. If a “&” (ampersand) is added to the end of the command, then the program is started as a background process instead of a foreground process. Immaterial of how a program is started (as a foreground or a background process that is), in Linux, you can toggle this state using the fg (foreground) and bg (background) commands. Experiment with these commands in the following manner:
6. Start emacs at the shell in foreground (don’t type an & at the end)
7. Change focus to your shell window and type in commands (like ls and pwd). Notice that the shell does not process them because it is waiting for emacs (that is running in the foreground) to complete.
8. Now in the shell window type control-z (you need to press z key while holding down the control key on your keyboard). This should cause emacs to stop (and you should see a message to that effect). Change focus to the emacs window and type some information (such as your name) to see if it shows up in emacs (and if you are wondering it should not show up)
9. Now, use the bg command (simply type bg at the shell prompt) to change the status of emacs to a background process. Now type some data in emacs.
10. Now, change back to the shell window and type fg at the shell prompt. This changes the status of emacs to a foreground process. Type some data in the emacs window and a couple of commands at the shell prompt to ensure that the status has changed.
11. Repeat sub-steps **ii** through **v** a couple of times to get the hang of changing the status from foreground to background process. However, note if the PID of your emacs process changes when you place emacs in the background.

|  |  |
| --- | --- |
| Did PID of emacs change (**yes**/no)? |  |

|  |  |
| --- | --- |
| If **yes**, what were the PID values? |  |

1. Now let’s practice how to copy files from the Linux machine to your local PC. For this first create a simple text file using the following command:

|  |
| --- |
| **$** ps > dump.txt |

1. When reporting errors or when attempting to secure help to troubleshoot problems I will often ask you to make a screenshot and include it as a part of your email. You can create screenshots of your desktop by pressing the “Print Screen” / “prt sc” button on your keyboard. Note that pressing the button will have no apparent effect other than placing a screenshot in your home directory. You can then insert the screenshot into a document. Perform this operation now by creating a screenshot of your entire desktop and pasting it below (size the image appropriately so that it fits width-wise on this page)

|  |
| --- |
|  |

1. Now, let’s try compiling and running a simple C++ program shown below on the Linux machine by following these steps:
   1. Open a C++ file in emacs using the following command:

|  |
| --- |
| **$** emacs WordCount.cpp **&** |

* 1. Copy-paste the skeleton code for WordCount C++ program shown below into emacs and save the program.

|  |
| --- |
| // A simple word count program that counts words from standard input  // Student name: Gill Bates  // Section: F-  // Compile: g++ -g -Wall -std=c++11 WordCount.cpp -o WordCount  **#include** <string>  **#include** <iostream>  **#include** <iterator>  **int** **main**() {  **int** wordCount = 0;  std::istream\_iterator<std::string> word(std::cin), eof;  **for**(; (word != eof); word++, wordCount++);  std::cout << "Number of words: " << wordCount << std::**endl**;  **return** 0;  } |

* 1. Next compile and run the program using the following commands:

|  |
| --- |
| **$** g++ -g –Wall –std=c++11 WordCount.cpp –o WordCount  **$** ./WordCount  one two  three |

Press control+d to issue a logical End-of-File (EOF) to the program and ensure that the output from the program is correct.

* 1. Next verify that the word count program is operating correctly by comparing its output against the output from the Linux wc (word count) command as shown below:

|  |
| --- |
| **$ cat WordCount.cpp | wc -w**  64 |

1. Once you successfully completed the aforementioned exercises, upload:
   1. This completed document
   2. Your version of WordCount.cpp
   3. The dump.txt file generated in question 16.

Ensure you actually **submit** the files after uploading them to Canvas.