CSE 391, Autumn 2019
Assignment 1: Basic Unix Shell Commands
Due Tuesday, October 8, 1:00 PM

This assignment focuses on using the `bash` shell to execute common Unix commands. Some of the questions are Unix commands you must figure out, and others are general questions about the particular Linux system you are using. Note: Unless otherwise specified, the answers to each question in Task 4 can be found entirely using commands shown in the lecture slides from the first week. You may use other commands if you like, but you should constrain yourself to those from lecture or from the *Linux Pocket Guide* textbook. Ask the instructor if you are unsure whether a particular command is allowed. For Task 3, there are reference cards posted on the course website but you are also allowed to google around. The goal of Task 3 is to become comfortable with using a command line text editor by practicing, not to quiz you or challenge you to find the "answer."

For Task 1 and Task 2 there is nothing to submit. For Tasks 3 and 4, you will submit your responses to Gradescope using the link on the course website.

Task 1: Log in to a Linux environment

First, **log in to a Linux environment.** You have several options to choose from here. Everyone should be able to install the CSE Virtual Machine (VM) Image. Other options include installing Linux on your machine, using one of the Linux computers in the CSE basement labs or using the CSE department's `attu` Linux server using the `ssh` program. We refer you to the Working At Home link on our course home page for information on options available. If you aren't able to successfully log in to a Linux environment, please contact us for help or ask a classmate.

If logged on to a Linux desktop environment (the CSE VM, a Linux machine in the basement), launch a **Terminal** window and a **text editor** from the Linux user interface, generally from the top left drop-down applications menu. You can usually find the terminal program under System Tools or Accessories. If logged on to `attu`, you can access editors like `emacs` or `vim` by typing them at the command line. You can also edit your text file on your own machine.

Task 2: Prepare a directory

We have set up a ZIP archive full of support files that you must download to your Linux environment. Do the following:

- Create a directory inside your home directory named `391` .
- Create a sub-directory inside `391` named `hw1` .
- Download our file `hw1.zip` and save it into your new `hw1` directory. You can do this in one of two ways:
    1. By opening a web browser on your Linux environment, browsing to our course web site, clicking the Homework link, finding the link to `hw1.zip`, right-clicking it, choosing Save Link Target As..., and browsing to the hw1 folder;
    2. Or, by typing the following command into your terminal window, when the current directory is `hw1` :
        - **`wget`** http://courses.cs.washington.edu/courses/cse391/19au/homework/1/hw1.zip
- Unzip the `hw1.zip` file's contents into your `hw1` folder. You can do this in one of two ways:
    1. By running a file browser/manager (in the CSE virtual machine you should have an icon that looks like a file cabinet that will open up a file manager) and browsing to the hw1 folder, then right or double-clicking on the `hw1.zip` file, and using the graphical unzipping program to extract the files;
    2. Or, by typing the following command into your terminal window, when the current directory is `hw1` :
        - **`unzip`** `hw1.zip`

        (We are trying to persuade you that doing things in a terminal can sometimes be the easier way!)

If you did everything correctly, you should now have several files and directories within your `hw1` directory, such as `java/`, `website/`, `animals.txt`, `Burrot.java`, `numbers.txt`, and `song1.txt` .

(Continued on next page)

Task 3: Getting Comfortable With a Text Editor

The following are exercises and questions are meant to help you become more comfortable with a text editor that is built into the command line. You can choose either vim or emacs. It does not matter which editor you choose, but we recommend that you pick one and stick to learning it for the remainder of the quarter. While the answers to the questions themselves are relatively easy to find by simply looking them up, **the real learning will come from you actually practicing these commands yourself**. While we won't be able to know whether you've really been practicing, this is not for our benefit, it's for yours. We also recommend getting even more practice by writing the answers to your task3.sh and task4.sh files using this editor ☺

1. Which text editor are you choosing to use? emacs or vim?
2. From the hw1 directory, how do you open animals.txt in the text editor of your choice?
3. Practice moving your cursor around the file. Move your cursor up, down, left and right. Assuming your cursor is at the beginning of the first line of the file, what are the keystrokes to move your cursor to the end of the line and append the text "animal"?
4. Next, what are the keystrokes to move your cursor back to the front of the line and insert the word "animal"?
5. How do you save your changes to the file?
6. How do you exit the file and return back to your command line prompt in the shell?

Task 4: Linux Bash shell commands

For each of the numbered items below, **determine a single `bash` shell statement that will perform the operation(s) requested**. Each of your solutions must be a single one-line shell statement and should not use Linux's multi-statement joining operators such as `|`, `&&`, `||`, and `;` . (We will learn about these next week.) Most of the questions below entirely use commands shown in the week 1 lecture and/or slides. Several questions require you to learn new parameters to those commands; find these out by looking at `man` pages or the *Linux Pocket Guide*.

To test your commands, you should have unzipped `hw1.zip` into the current directory. You can assume you are in the "hw1" directory when doing these problems.

In response to each question, you will provide **the command that will perform the task described**, not the output that the command produces. Write your commands in on the indicated lines in the `task4.sh` file in the `hw1` folder.

1. Copy the file `MyProgram.java` from the current directory to the `java` subdirectory.
2. List all files in the `/var` directory, in reverse alphabetical order. (Notice the slash!)
3. List all files in the current directory, in "long listing format".
4. Print a text calendar for the month of July 2019.
5. Rename the file `Burrot.java` to `Borat.java` . *(Hint: Renaming is done using the same command as moving.)*

(Continued on next page)

6. Delete the files `diff.html` and `diff.css` . Note that your answer must be a single command and not multiple commands. *(Hint: Many commands can accept more than one parameter.)*
7. *(Self-Discovery)* Set the file `MyProgram.java` to have a last-modified date of January 1, 2019, 4:15am. *(Hint: the `man` page for the proper command describes the timestamp 'STAMP' format to use. Look for this!) (Also note that Linux is case-sensitive when you are specifying file or directory names.)*
8. *(Self-Discovery)* You can use a `*` (asterisk) as a "wild-card" character to specify a group of files. For example, `*foo` means all files whose names end with `foo` , and `foo*` means all files whose names begin with `foo` . You

can use a wildcard in the middle of a file name, such as `foo*bar` for all files that start with `foo` and end with `bar`.

List all web page files (files whose names end with the extension `.html` or `.css`) in the current directory. Note that the `ls` command can accept more than one parameter for what files you want it to list (e.g. `ls website/ java/`).

9. Copy all the text files (files whose names end with `.txt`) from the current folder to the `java` subdirectory.
10. *(Self-Discovery)* The `cat` command outputs the contents of a file to the terminal.
    The `less` command outputs the contents of a file to the terminal, page by page, pausing for you to press a key.

    Use whichever command is best suited to display the contents of the file `lyrics.txt`.
11. Display the **contents** of all files whose names begin with `song` and end with the extension .txt, such as `song1.txt` and `song2.txt`. *(Write a single command that displays all their contents concatenated.)*
12. *(Self-Discovery)* The `head` and `tail` commands output only the first or last few lines (respectively) of a file to the terminal.

    Display **only the first 7 lines** of the file `animals.txt` from the current directory on the terminal.
13. *(Self-Discovery)* The `wc` command outputs how many bytes, words, lines, etc. a file occupies.
    Display **only** the number of **lines** in the file song3`.txt` (It is fine to display the filename as well as the number of lines, just don't display the number of bytes and words.)