CSE 391, Autumn 2019
Assignment 6: Users, Groups, Permissions
Due Tuesday, November 12, 1:00 PM

This assignment continues to practice using the `bash` shell and basics of combining commands using redirection and pipes. Advance warning: it can be tricky to "see" files that start with a dot in file dialog boxes when you are trying to, for example, upload them to Gradescope. Check out the link on the homework page for tips on viewing hidden files.

Task 1: Bash shell commands (recommended not on attu)

For each item below, **determine a single `bash` shell statement that will perform the operation(s) requested**. Each solution must be a one-line shell statement, but you may use operators such as `>`, `>>`, `<`, `|`, `&&`, and `;`. For all commands, do not create any files except those indicated.

To test your commands, you should have unzipped `hw6.zip` into the current directory. You can assume you are in the `hw6` 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 `task1.sh` file in the `hw6` folder.

Note: We highly recommend you work on this section on the VM, not attu. CS support dislikes a surge of people needing admin intervention due to messed up permissions, and you might inadvertently open up your files for others to access :)

1. Set the file `example1.txt` in the current directory (you do not need to use `find` here) so that its owner and others can execute the file. *(*All other permissions should remain unchanged.)

2. (a) Set all files with extensions `.java` and `.txt` in the current directory (you do not need to use `find` here) to be readable (but not writable or executable) by their owner, and to allow no access by any users other than the owner. Do this using the standard letter code arguments for granting and removing permissions.

   (b) Write this same command using the octal number code arguments to grant/remove the permissions.

3. Set all files in the current directory and all its subdirectories (and sub-sub-directories, etc. recursively) to be owned by the group `wheel`. (If you can't get this to work see the tips on the homework page.)

4. Set all `.java` files in the current directory and all its subdirectories (and sub-sub-directories, etc. recursively) to have read permission for all users. *(You can't achieve this simply with a `chmod` command; you will need other commands taught recently for finding a group of files and processing each one as a command-line argument.)*

5. *(Self-Discovery)* The `umask` command is used to specify what permissions are given by default to newly created files. Check the slides from lecture to see a more detailed discussion of how to use `umask`. Its format might be the *opposite* of what you expect—it actually specifies what permissions will be *taken away*.

   Use `umask` to set the default permissions for new files to be read (but not write or execute) permission to you (the owner), but no permissions to anyone else.

6. *(Self-Discovery)* Give a command that *would* print out the contents of the file `password.secret` **as the root super-user**. Notice that you do not have the permissions simply print the contents of this file normally, as you will get a Permission denied error. Running the command to print the file as the root super-user will allow you to see the contents of the file without needing to change the file's permissions. (Note: depending on where you are trying to do this, the actual command may not work. Specifically, on `attu` you do **not** have permissions to do this, but you should on your VM. Give the command that you *would* use, even if it doesn't work in your environment.)

(continued on next page)

Task 2: .bashrc

As we have discussed in class, `.bashrc` is a script that runs every time you start a new Bash shell (e.g. by typing `bash` at the bash prompt. `.bashrc` is also run when opening a new terminal window on the CSE VM). For this part of this assignment, you should modify the `.bashrc` file in your home directory on your Linux environment so that it sets the following aliases. If you have an account **on `attu`** or another shared server, you probably already have a `.bashrc` file there that you can modify. If your system does not have such a file, you can just create a file named `.bashrc` and add your aliases there.

For the items below, **your modified .bashrc file.**

1. Add an alias so that typing `attu` connects you to `attu.cs.washington.edu` via SSH. (This alias isn't very useful if you're testing on `attu` itself, but set it up anyway.)
2. (a) Add an alias so that when trying to overwrite a file during a move operation, the user is prompted for confirmation first. *(Hint: Set the operation to run in "interactive mode".)*

   (b) Add an alias to create the same behavior for the copy operation.

Task 3: .bash_profile

As we have discussed, `.bash_profile` is a script that runs every time you **log in** to a Bash shell. This will happen every time you log on to `attu`. (On the VM things are bit more complex due to how they have configured things. A terminal window is also not considered a "login shell". You can run your `.bash_profile` by starting a shell by typing `bash -l` ("el") in a terminal window.) In general if you want to make personalizations to your environment that produce output (e.g. print something to the screen) you should put them in your `.bash_profile` rather than in your `.bashrc`. If you have an account on `attu` or another shared server, you probably already have a `.bash_profile` file there that you can modify. If your system does not have such a file, you can just create a file named `.bash_profile` and add your changes there.

For the item below, **submit your modified .bash_profile file.**

1. Display a personalized message when logging on to the system. (e.g. "Welcome Zorah! Have a great day!") You can do this by just listing the command you would execute at the bottom of your `.bash_profile`. Feel free to get creative with this! Anything will be fine here as long as it prints some output to the screen that would not have happened otherwise. (but please keep messages positive and respectful!)