2G03 Homework 0, 2020

Due End of Day, Monday Sept 14th

In this introductory assignment, we will be going through the fundamental operations you will need to get around and work within a Unix environment. The analogous operations would be opening your file explorer in Windows or Mac OS and learning how the create new folders, navigate between different folders, create files, and copy and move files. You will need these skills as all coursework will be completed on the phys-ugrad server, which is a Unix environment. For this assignment you will already need to know how to log in and access phys-ugrad via a terminal and `ssh`.

The grading for this and all assignments is based partly on what you hand in and partly on what is made in your home directory on phys-ugrad. **So for full credit, be sure to actually type in the commands from each question and create the directories and files**! When you are prompted to record outputs from the terminal here, record them in a text file or Word doc, labelled by the question or section number. You will hand that in as a part of your assignment on Avenue under Assignments page which is accesible via the Assessments tab at the top. This assignment is due by **End of the day on Monday, September 14**.

1 Where am I?

First, log into phys-ugrad with the `-Y` option. You will land in the home directory. The command pwd (which stands for 'print working directory') tells us what directory we are currently in. Type

`pwd`

into your terminal on the command line, and press the 'Enter' or 'Return' key on your keyboard to run the command. ('Return' is the key will execute any command currently typed onto the terminal command line.) **Record the output. Do this by copy and pasting or typing the output from the terminal into the text document you are recording your answers in. (A screenshot is not necessary.) For this assignment, be sure to label each recording in your document with the question that it came from.**

Lastly, for this output of `pwd` above, explain what you're seeing.

2 "What does a command do again?"

If you forget what a command does, there are many resources on these basic commands on the internet which you can find with a search engine. There is also an in-terminal reference: the Linux manual `man`. For example, typing `man pwd` will bring up the manual for the `pwd` command. You can quit the manual by typing 'q'.

(There is nothing to record for this question.)

3 Creating a directory

Creating directories (or folders) is a good way to organize your work. We can make a new directory with the command

`mkdir` ⟨directory name⟩

Let's replace ⟨directory name⟩ with the name we want to use. Make a new directory called `HW0` with `mkdir`.

(There is nothing to record for this question.)

4 What do I have?

On the command line, we can see what's inside of a directory using the `ls` command. Typing `ls` alone into the terminal (that is, by itself, with nothing else after `ls`) gives a list of the files in our *current directory*, printed to the terminal. ("How do we figure out what directory we are in again?": `pwd`). **Run the command**

`ls`

Record what you see.

5 Moving around

So we know where we are with `pwd`, and we know how to see what's here with `ls`. How do we move around? The command

`cd` ⟨directory name⟩

stands for 'change directory', and it allows us to move around the Linux file system. Use the `cd` command to move into the `HW0` directory you made in the previous question. **Use `pwd` to confirm that you've done this correctly. Record the output.**

All work for this assignment will be done within `HW0`, make sure when you're adding files and directories, it is within the `HW0` directory.

6 Files vs Directories

In this course we're going to be writing LOTS of text files. There are a couple of ways to make a text file in Linux. The easiest way to make an *empty* file is to use the command

`touch` ⟨filename⟩

command. Use this command to make a new file '`example.txt`'. **Record the output of the `ls` command after running the `touch` command**.

7 Home again

We'll often want to return to our home directory. Linux offers us a convenient way to to do this by running

`cd`

by itself, with no input. Do this now. Run `pwd` to ensure that you're back home. Now run

`ls`

The output should be the same as in Q3. Now use `ls` to check the content of the directory that you made in Q4: you can use the command

`ls` ⟨directory name⟩

to see the content of any directory. **Do this now for your directory `HW0`, and record the output**.

8 Copying

Now we know how to create directories and move around the file system, it's time to learn how to copy and move directories. First, let's make a new directory for this exercise. Change into the `HW0` directory, and make a new directory called 'bottom' (without the quotes).

The Linux copy command is

`cp` ⟨filename⟩ ⟨destination⟩

⟨filename⟩is the file or directory we are going to copy, ⟨destination⟩is the location or directory where we are copying it to. Use `cp` to copy the `example.txt` file into the 'bottom' directory. Move into the 'bottom' directory and use the `pwd` and `ls` commands to confirm that you've done the above steps. **Record the output of those commands.**

9 Copying 2

What if we want to copy a whole directory? To do this we need to copy the directory and any files that may be inside it. In Linux commands, extra options can be added using 'flags' such as `-r`. These flags or options appear after command name and unlock more functionality in the command. A more complete syntax for the `cp` then is

`cp` [options] ⟨filename⟩ ⟨destination⟩

The '-r' flag stands for recursively. Using it with `cp` allows us to copy whole directory structures.

Let's return to our `HW0` directory (by typing `cd` and then `cd HW0`). Inside of `HW0` make a new directory called '`middle`'. (Recall, if you've gotten lost in your directory structure, you can always go home by running `cd` by itself.) Use the command

```
cp -r bottom middle
```

to copy the directory `bottom` and all of its contents into `middle`. **From `HW0`, use the `ls` command on the 'middle' directory and record the output**.

10 A useful command line tip

In the last question, we just typed out the command '`ls` middle' to list the contents of the 'middle' directory. Linux offers a *very* handy utility that will save lots of time in typing commands. Pressing the 'tab' key on your keyboard will auto-complete any Linux command.

For example: Move to the `HW0` directory structure if you're not already there. Without pressing 'Enter' or running the command, type `ls` by itself, with one space after it, and press the 'tab' key twice. If you've followed all previous steps exactly, you should see the directories 'bottom/', 'middle/' and the original file 'example.txt' printed to the terminal. They are printed here because the 'tab' key is trying to auto-complete the `ls` command, and these are the two directories/files in `HW0` that are present, either of which you could pass as inputs to `ls`. Now, add the letter 'm' to the command line so that your terminal prompt reads exactly '`ls m`', and press 'tab'. **Describe and record what happens.**

Clear the command line after this: you can do this by removing your command from the prompt with 'backspace' or executing the command with 'Enter' or 'Return'.

11 Moving

If we want to move a whole file, including a directory (this is like 'cut and paste' from Windows or Mac OS), we use the command

`mv` ⟨filename⟩ ⟨destination⟩

In `HW0` make a new directory called 'top'. Now let's move the directory structure from the previous question into the new directory. Run

```
mv middle top
```

After moving 'middle' into 'top', change into the the directory 'top'. Use `ls` to list the contents. Use `pwd` to print where you are. **Record the results. Change into the directory bottom and run the same commands and record those results too.**

12 Removing

Have you accidentally created any files or directories during this exercise? You can remove them using the `rm` or `rmdir` commands. **Note**!!: Linux does not have a 'trash bin' like Windows and Mac OS, where files go after they are 'deleted' and, if the delete was a mistake, where they can be recovered. That is, in Linux, once a file or directory is removed, it is **gone for good.** Be careful when using these commands! By default, when using `rm` you will be prompted with a question asking if you are sure you meant to remove that file, type 'yes' to confirm, 'no' to cancel.

Let's change into `HW0` and delete our original copy of the '`example.txt`' file that we had. To do this use the command

rm ⟨filename⟩

Type *yes* if you are asked if you are sure you want to delete the file. Record the output of `ls` in the directory `HW0` after you've done this.

You should use `rm` to remove any files you may have accidentally created. The command `rmdir` will remove a directory only if it is empty, otherwise you will get an error message, so if you would like to remove a full directory you have to remove every files inside of them first.

Change directories to `top`. You should see two directories: `bottom` and `middle`. Navigate to `bottom` and remove the file stored inside. Then navigate back into `top` and run `rmdir bottom`. **Type `ls` and record the output.**

13 A few command line tips for navigating

At this point our directory has one branch that goes in a straight line: from 'top' to 'middle' to 'bottom'. We can of course add more directories at any of these levels to branch out our file system structure. Let's add a new directory, but explore a useful command line tip along the way.

Navigate to your `HW0` directory. Now let's navigate all the way to the 'bottom' directory, but note that to get to 'bottom', you don't need to execute several `cd` commands separately–i.e. `cd top`; Return; `cd middle`; Return, etc. You can get there all in one go! From `HW0`, executing `cd top/middle/bottom/` will get you all the way to 'bottom' (remember you can use the 'tab' key to auto-complete the prompt and save you time typing).

Now that we are in 'bottom' we are going to back to 'top' to make a new directory. We could get back to 'top' by first executing `cd` by itself, with no inputs, which would take us home, and then we can get to 'top' from `HW0`. Instead, let's move *upwards* from 'bottom', back to top. To do this, we can use the characters '`../`' (without quotes) in the input to `cd`. The characters '`../`' are a short-hand for going up one directory in our filepath. That is, executing '`cd ../`' from 'bottom' will take us to the 'middle' directory, and executing that command again from 'middle' will take us to top.

Also, you can chain these '`../`' characters together! Executing '`cd ../../`' from 'bottom' will take us directly to 'top'.

These '`../`' characters are very useful, they can be used whenever you are trying to input a file to a command to modify the filepath upwards one directory.

Now, navigate to 'top', and create a new directory 'side'. In the directory side, create a new file 'sideexample.txt' with the `touch` command. **Record the outputs of `ls` and `pwd` from the 'side' directory**.

14 Editing files

We're going to make another '.txt' file, but put some content in it this time. Navigate to 'bottom'. (Check with `ls` that there is only the 'example.txt' here.)

To add content to any file, you can use a number of text editors. In this class we recommend using gedit because it is similar to Notepad in Windows. If you have experience in the terminal and are interested in using other editors, you are of course welcome to.

Using gedit will require us to be able to get graphical interfaces from phys-ugrad. This means you will have to have logged in with the `-Y` option. (The option shouldn't be necessary on MobaXTerm, this graphical forwarding option should be done by default, in the background.)

To create a new text file using gedit, run the command

`gedit` ⟨filename⟩ `&`

(The ampersand `&` ensures gedit runs in the background of the terminal, so that you can still use the command line there.) Use the above command to create a file 'textexample.txt'. If all goes well, you should see this window pop up on your screen:
If you see something like this error message:

```
Gtk-WARNING **:  cannot open display:
```

this is due to an error in getting graphical interfaces. Ensure that you have logged into phys-ugrad with the `-Y` option.

Once you have that text editor window, we are free to start writing text into the file! Go ahead and write a nice message to yourself. If you are clicked into the gedit window and you see the cursor blinking, it is ready for your commands. Once you've finished, click the save button in the top right. (Notice that when the file has unsaved changes, there is an asterisk `*` to the left of the filename. You'll know there are no unsaved changes with this does not appear.)

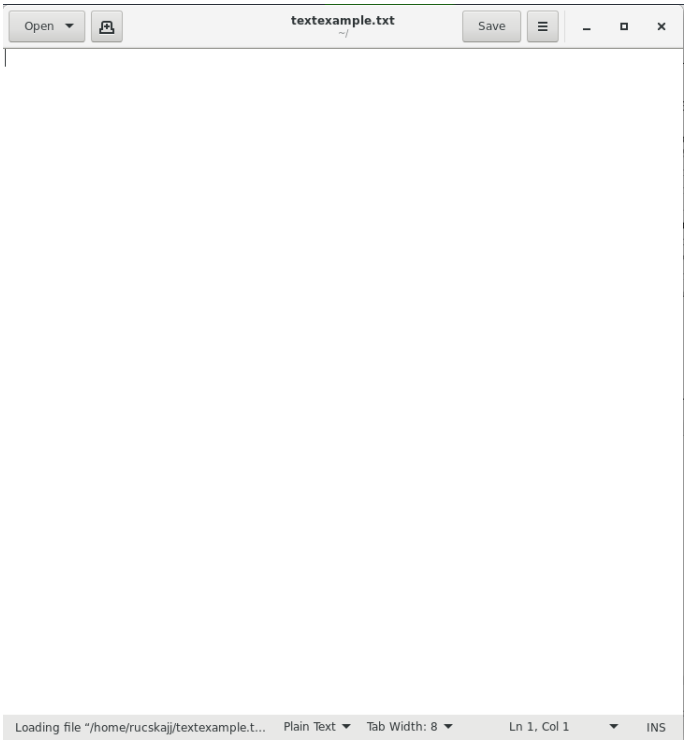

Figure 1: A screenshot of the text editor gedit.

As you type and save in gedit, other warnings will appear in the terminal window. It is okay to ignore these.

Once you've saved your message, close gedit by clicking on the 'x' in the top right. Go back to the terminal window. If you don't see the prompt: `[macid@phys-ugrad]$`, use the key combination `Ctrl + C` to end the gedit process, or hit 'Enter'/'Return'. The prompt should reappear. **Run `ls` to ensure your 'textexample.txt' saved properly. Record the results.**

15 Inspecting text files from the command line

The command

`more` ⟨filename⟩

will print the contents of a text file to the terimanl, one terminal page at a time. **Run `more` on your 'textexample.txt' file to double-check the contents you put there in gedit were saved to the file properly. Record the output to the terminal.**

16 Renaming files

Recall the syntax for `mv`:

`mv` ⟨filename⟩ ⟨destination⟩

If ⟨destination⟩ is *not* a previously existing file or directory, then the `mv` command renames the file given by ⟨filename⟩ to the name given by ⟨destination⟩.

We should currently be in the 'bottom' directory, and have two text files in the directory. Use `pwd` and `ls` to confirm this. The file 'example.txt' was created using the `touch` command and is thus empty, it has no content. **Use `more` on 'example.txt' to confirm this. Record a description of what happens**. Use `mv` to rename 'example.txt' to

'emptyexample.txt', which is more descriptive. **Run `ls` to confirm this renaming was successful. Record the output**.

It is useful to always try and use descriptive names for files and directories (and variable names, which we will get into when we start programming).

17 Inspecting text files from the command line

A useful option for `ls` is the '`-l`' option, which activates the "long listing format" for the output. From 'bottom', run

```
ls -l
```

The output to the terminal will contain a lot more information than just the filenames, which we see with `ls` by itself. The combination of `r`'s, `w`'s, `x`'s and dashes (`-`) describes the permissions that apply to the files, the usernames say who owns the file, and the date and time tell when the file was last written to. The filename appears at the end of the line. The number in between the usernames and the date is the size of the file, in bytes. **Run `ls -l` in 'bottom'. Record the output. What part of the output is different between the two text files, and why?**

18 The great tree of Linux

Through this exercise, we've added quite a lot of structure within our `HW0` directory. **Type out/sketch the file tree for your `HW0` directory**. (From the very top, at `HW0`, you can "probe" the entire structure with the '`ls` ⟨directory name⟩' command, recall!) Place your `HW0` directory on top and include all the directories and files that we've made during this assignment.

A sample directory tree made in plain text is shown below. You should try and make something similar to this, though the format is not strict. Anything that clearly marks the separate directories and their contents will do.

```
Project
 |
 +-- file 1
 |
 +-- dir 2
 |  |
 |  +-- file 2.1
 |  +-- dir 3
 |      |
 |      +-- file 3.1
 |      +-- file 3.2
 |
 +-- dir 4
 |  |
 +  |-- dir 4.1
```

19 Submitting your work

Note, you will need to have the exact structure of files and directories–which means the exact same directory and file names–that we've outlined here to receive full marks. Use the `rm` and `rmdir` commands to modify your `HW0` directory as needed, if you've made mistakes along the way.

Recall that part of your submission is not just the recorded outputs from this exercise which you will submit as an extra document, but the actual file structure on your phys-ugrad account. All assignments in this course will follow this structure: you will submit a document with answers to questions and the instructors will reference the code you've created on phys-ugrad as part of your submission.