NATIONAL INTEGRATED CYBERINFRASTRUCTURE SYSTEM
CHPC

NITheP
National Institute for Theoretical Physics

# SUMMER SCHOOL 2021

## Files and Folders

**Dr Krishna K. Govender**

### Summary of commands used in this lecture

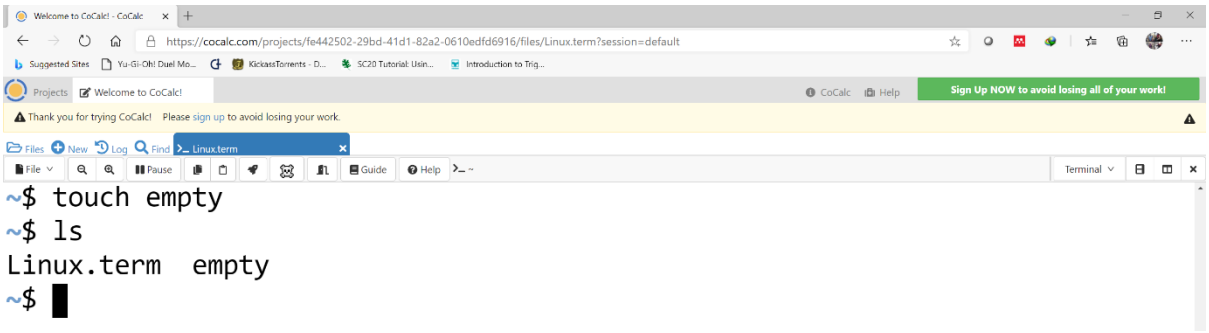
| Command | Meaning |
|---|---|
| **touch** | Used to create an empty file or update the timestamp of a file |
| **ls** | List items in current directory |
| **cat** | Concatenate a file/s |
| **echo** | Print text to screen |
| **mv** | Used to move/rename a file/directory |
| **cp** | Used to make a copy of a file/directory |
| **rm** | Used to remove a file/directory |
| **pwd** | Present working directory |
| **mkdir** | Used to make a new directory |
| **cd** | Change directory |

- As in the previous lecture all commands provided in **BOLD** and center aligned from here onwards are commands that can be typed/executed directly in a terminal.

- Let's create a file called empty which contains no information and then check that the file exists by listing the items in our current directory:

**touch empty**

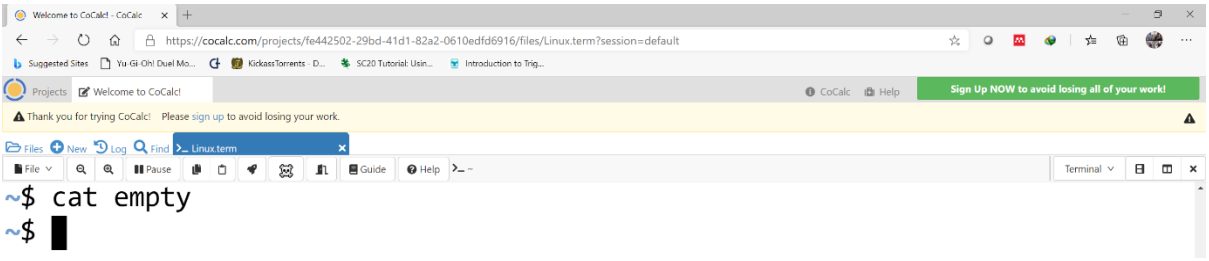
**ls**




- Those of you using CoCalc will recall from the previous lecture that the file Linux.term was already there, and we see this file once more, but there is also a new file called empty which we just created.

- We can display the information contained in the file named **empty**:

**cat empty**

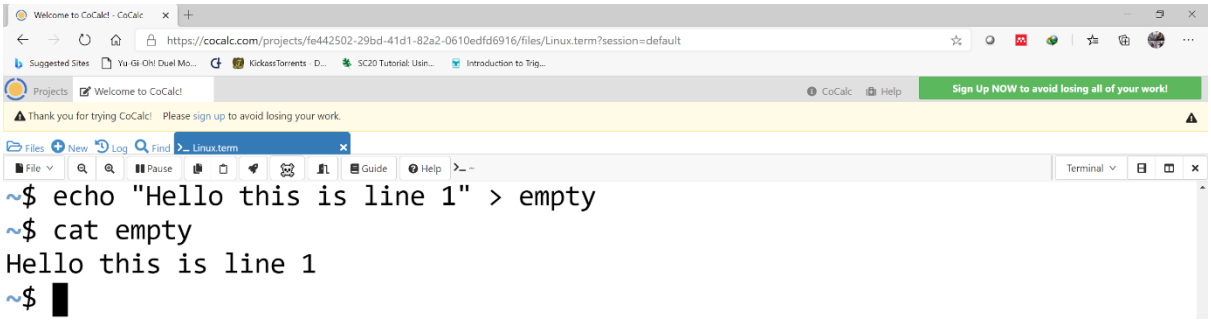



- Since the file has no information inside it, we find nothing being displayed.

- Now let us put in some information into the file called empty.
  The are various ways of adding information to a file, but for the current lecture we will use the following approach:

**echo 'Hello this is line 1' > empty**

**cat empty**




- As we saw in the previous lecture **echo** allows us to print information to screen, but instead of printing the information to screen we are redirecting (>) this to the file called **empty**.

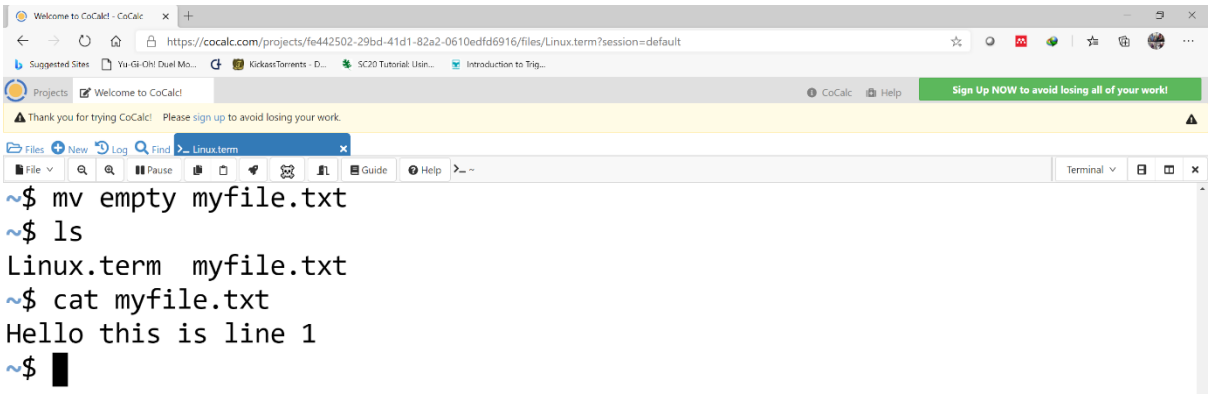
  Next, we display the contents of the file **empty** (**cat empty**) and we find that the text from the **echo** statement is contained therein.

- Now that our file contains some information it does not make sense to have it named **empty**, so the file needs to be renamed:

  **mv empty myfile.txt**

  **ls**

  **cat myfile.txt**




- **mv** allows us to move a file from the name **empty** to **myfile.txt**.

  By listing (**ls**) the items in the current folder we find that there is no longer a file called empty, but there is a file called myfile.txt.

  Displaying the contents of **myfile.txt** (**cat myfile.txt**) we see that the information is exactly that same as that which was contained in the file **empty**.

| |
|---|
| You will notice that I decided to add a **.txt** extension to the new filename and this is merely so that I know that this file contains text. Giving files extensions allows you to know what is inside these files. |

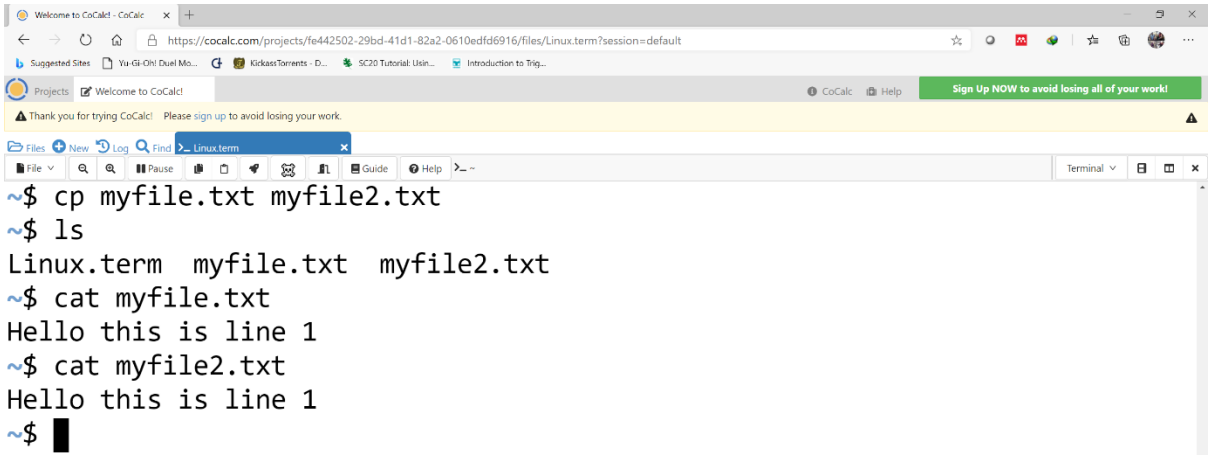
- Should we need to make a copy of our file:

**cp myfile.txt myfile2.txt**

**ls**

**cat myfile.txt**

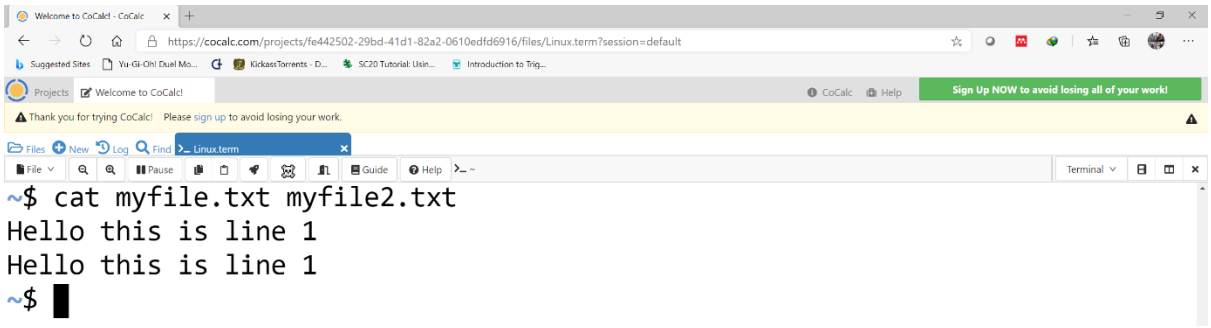
**cat myfile2.txt**




- Now we have two files (**myfile.txt** and **myfile2.txt**) that contains the same information.

- It is also possible for us to display the information contained in two files at the same time:

**cat myfile.txt myfile2.txt**



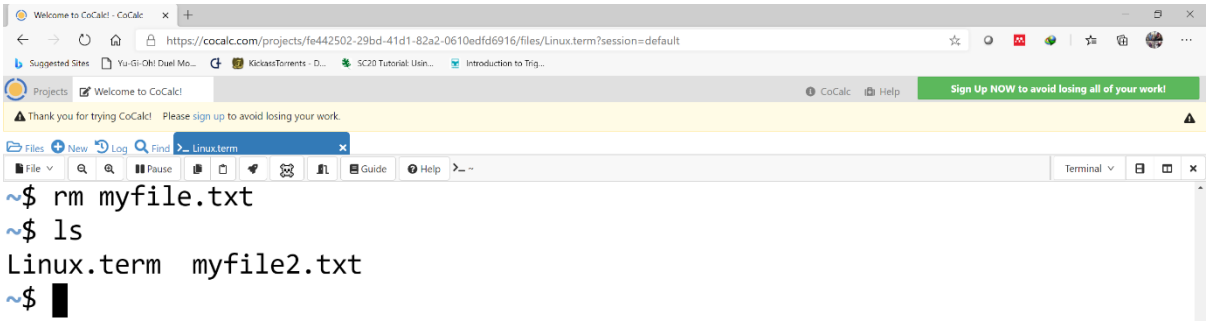

- Off course in the current case, since both files have the same information, it does not make sense to display them at the same time.
  However, this is something that can be useful when you have data in separate files that you wish to view simultaneously.

| Note that anything you display on screen in Linux can always be redirected (>) to a file. |
|---|

- Now that we have made a copy of **myfile.txt** we can remove the original file:

**rm myfile.txt**

**ls**



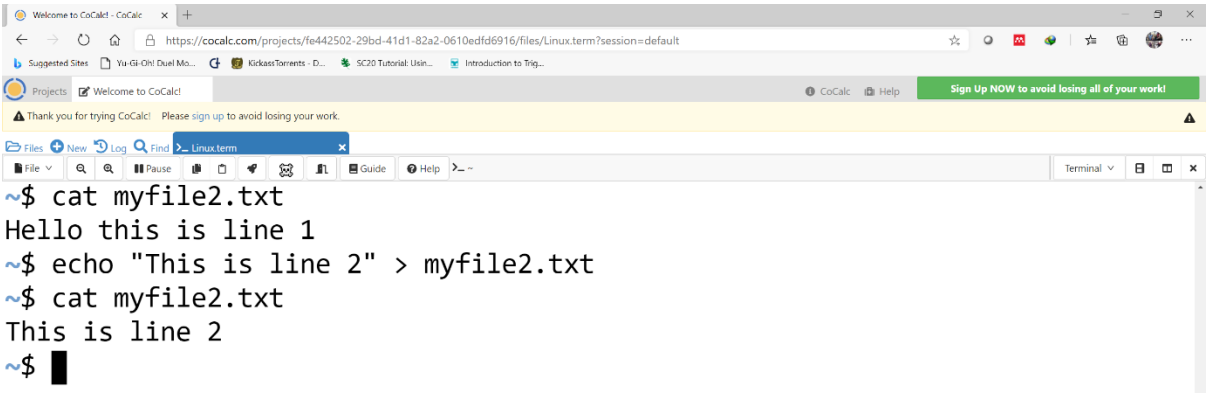

- We can see from the image above that the file **myfile.txt** no longer exists.

- Let's try to add an additional line of information into **myfile2.txt**:

**cat myfile2.txt**

**echo 'This is line 2' > myfile2.txt**

**cat myfile2.txt**




- At the end of the above process **myfile2.txt** only has the text **This is line 2**.
  The text **Hello this is line 1** seems to have been removed.
  Instead of adding (appending) onto the file **myfile2.txt** we seemed to have overwritten the information.
  This is because > is a sign that's used to write and overwrite information, whereas >> is used to append to a file.

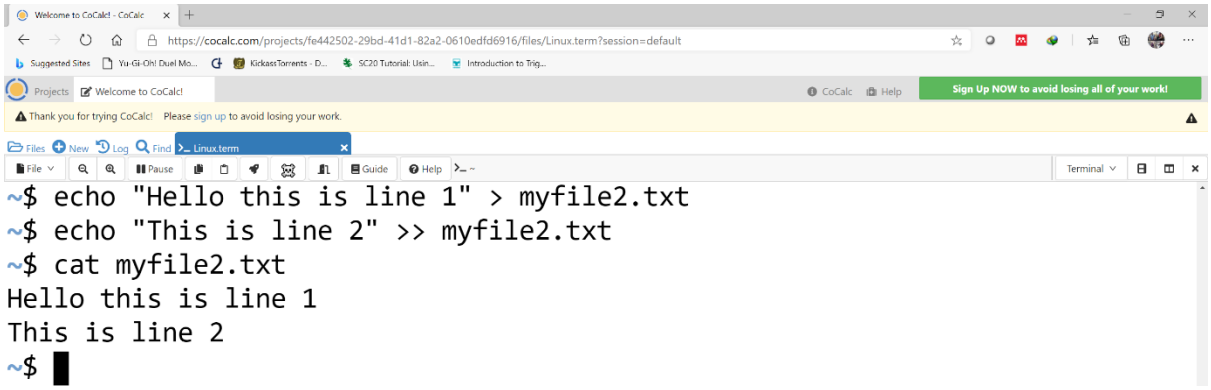
| Note that once a file is removed in Linux it **cannot** be retrieved, so be very careful before removing any files. |
|---|

- Now let's try to get **myfile2.txt** with both the lines we would like to see:

**echo 'Hello this is line 1' > myfile2.txt**

**echo 'This is line 2' >> myfile2.txt**

**cat myfile2.txt**




- The first **echo** statement replaces any information that might be in the file **myfile2.txt** with the text **Hello this is line 1**, while the second statement adds the text **This is line 2** to **myfile2.txt** without removing anything.

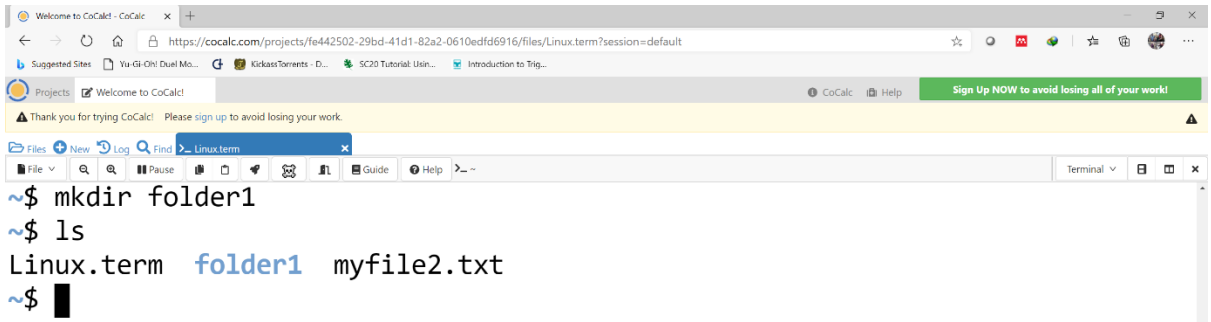
  By displaying the contents of **myfile2.txt** (**cat myfile2.txt**) we find that we now have two lines of text contained therein.

- Up to this point we worked only with files, but now we will look at directories (also known as folders).

  We start by creating a directory:

**mkdir folder1**

**ls**




- You will notice that files and folders are colored differently, with the case above showing **files** in black and **folders** in blue.

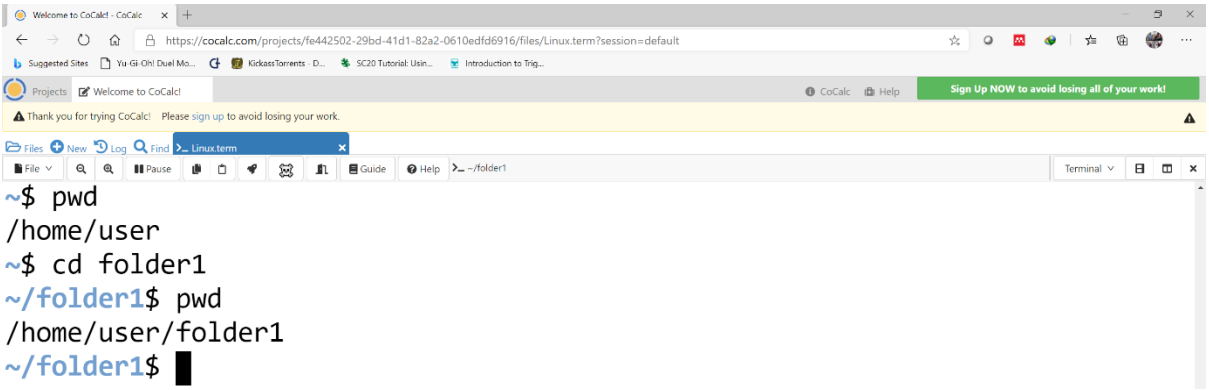
| Note that the colors for files and folders in Windows, Linux and MacOS may differ from those provided, but all Linux systems will always have files and folders in different colors. |
|---|

- Next let's have a look at how to move into a folder:

**pwd**

**cd folder1**

**pwd**

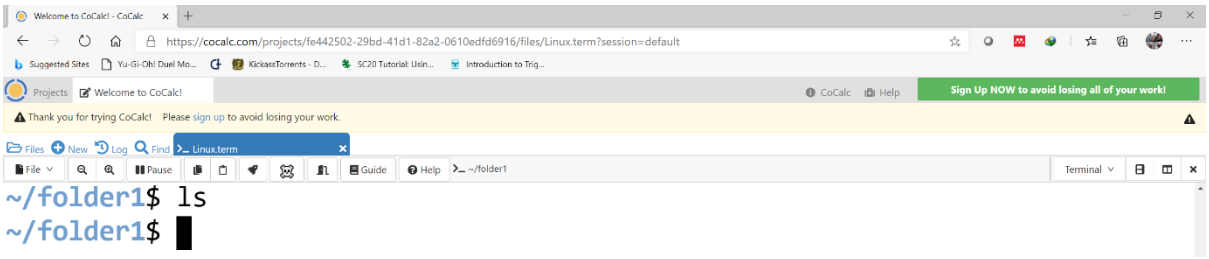



- First, we check our present working directory (**pwd**), then change directory into our new folder (**cd folder1**) and finally have a look at our new present working directory (**pwd**).

- You will notice that our folder shifts from **/home/user** to **/home/user/folder1**

- List the items in this new folder:

**ls**



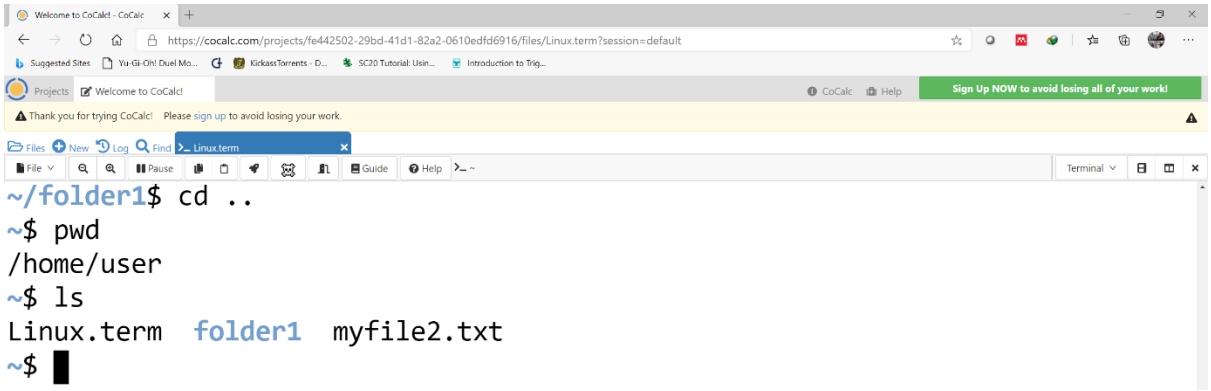

- There appears to be nothing in **folder1**, but what happened to the files **Linux.term** and **myfile2.txt** which where present when we first created **folder1**?

  Those files are still there, but they are not in **folder1**, instead they are located one directory back:

**cd ..**

**pwd**

**ls**




- To get one directory back we need to first change directory (**cd ..**), where the .. tells Linux to move back one directory.
  We then end up back where we started from when we first created **folder1**.
  Now you see that the files **Linux.term** and **myfile2.txt** are still present.

- Make a new directory called **folder2** and move it into **folder1**:

**mkdir folder2**

**ls**

**mv folder2 folder1**

**ls**

**pwd**

**cd folder1**

**pwd**

**ls**

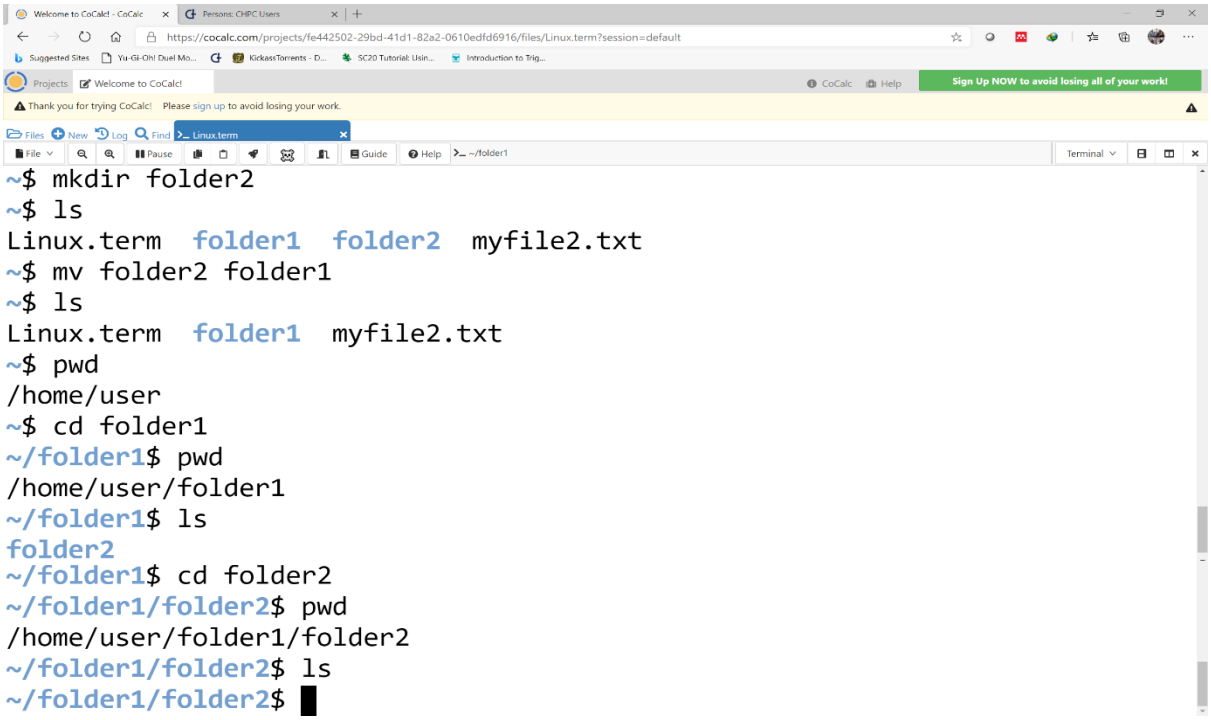
**cd folder2**

**pwd**

**ls**

Some important syntax to keep note of when changing directories:

1) **cd .** (Means stay in the current directory).
2) **cd ..** (Means go one directory back).
3) **cd ../../** (Means go two directories back).




- After creating **folder2** we move it into **folder1**.

  Unlike in the case with the **empty** file above where the name merely changes, directories work slightly differently.

  Since **folder1** already exists a move will NOT merely rename **folder2** to **folder1**, instead it moves **folder2** into **folder1**. As a result, we still have 2 directories with one being inside the other (**folder2** is inside of **folder1**).

  Listing the items inside each folder as we change directory shows us that we have **folder2** inside of **folder1**, while **folder2** has no additional files or directories.

- Now we want to return to the directory where we originally started from and being in **folder2** it puts us two directories from that position:

  **pwd**

  **cd ../../**

  **pwd**

  **ls**

| |
|---|
| Note that regular files typically possess information, such as the data you may have generated from a simulation.<br>Directories/folders are like filing cabinets used to store the regular files. |


- Copy **myfile2.txt** into **folder1** and ensure the contents of the file are still the same:

**cp myfile2.txt folder1/**

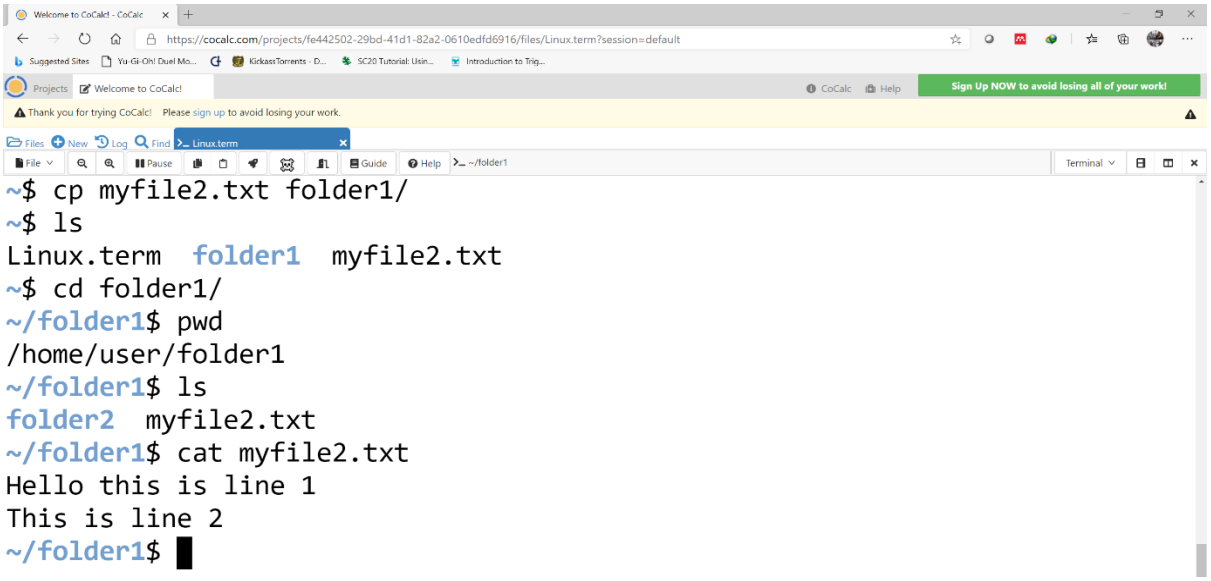
**ls**

**cd folder1**

**pwd**

**ls**

**cat myfile2.txt**




- Since **cp** makes a copy of the file we see that the original **myfile2.txt** is still present in **/home/user**.

  There is also a copy of **myfile2.txt** present inside of **folder1**.

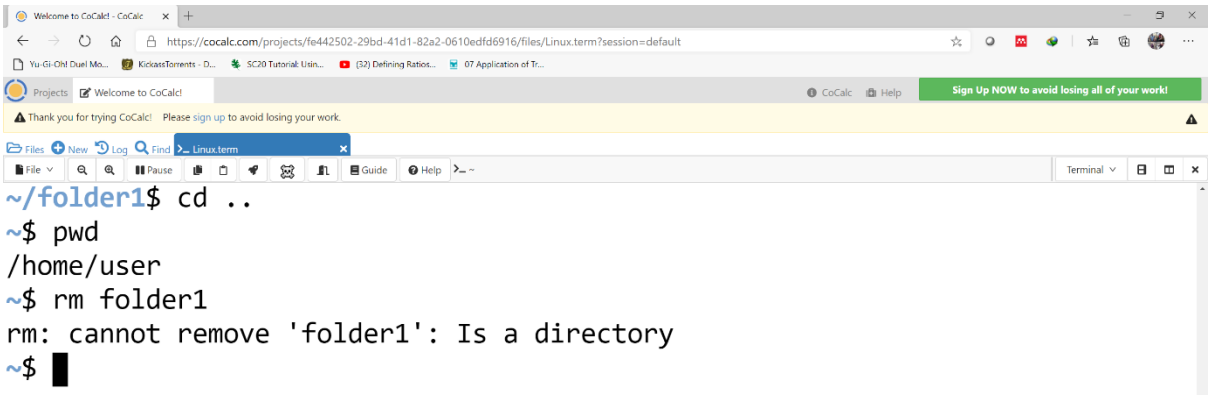
  To ensure that this copy has the same information as the file we created earlier in the lecture we display its contents (**cat myfile2.txt**).

- Finally, we will remove **folder1** and everything contained therein (**myfile2.txt** and **folder2**):

**cd ..**

**pwd**

**rm folder1**

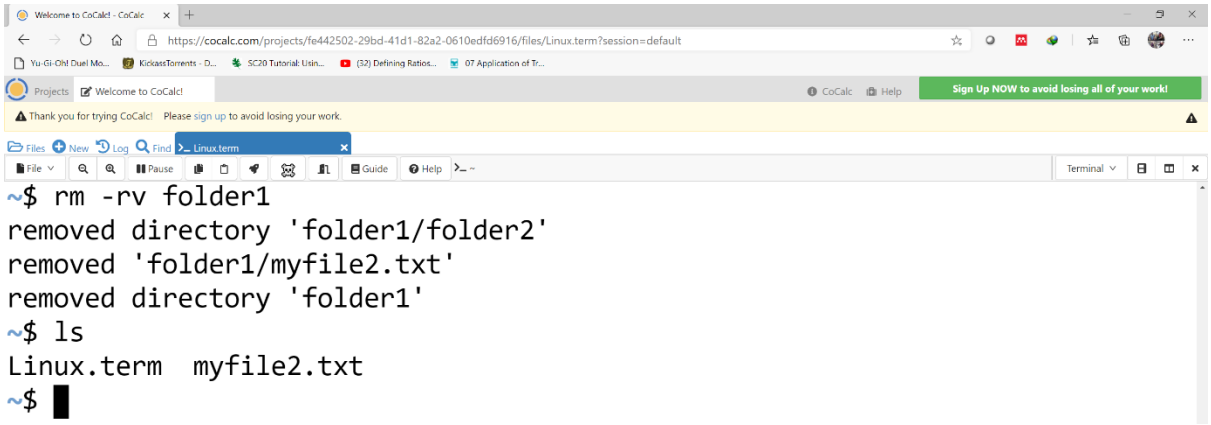



- Since we are already inside of **folder1** we first need to move one directory back (**cd ..**). When we try to remove the folder there is an error stating that **folder1** is a directory.

- While **rm** on its own works for removal of files we need to use an additional flag when dealing with directories:

**rm -rv folder1**

**ls**




- We make use of the -r flag when removing directories to tell Linux to remove the files recursively.

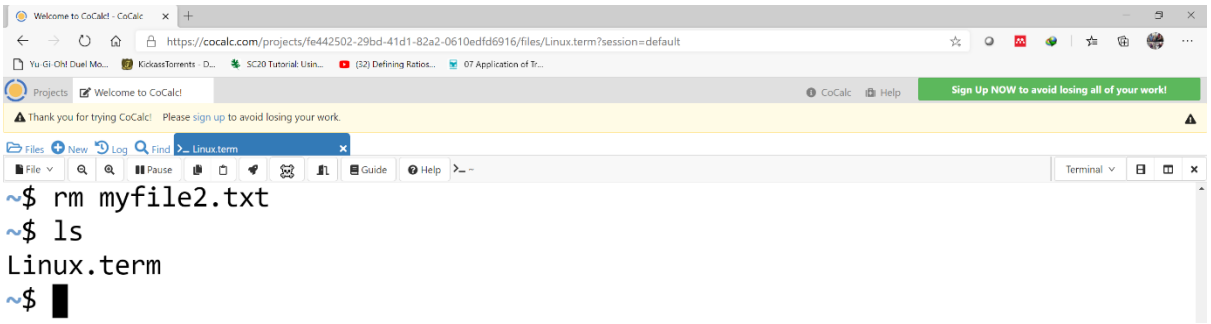
  I included the additional -v option, which stands for verbose, to display the removal process as it happens.

You will notice that the first thing that gets removed is **folder2**, followed by **myfile2.txt** and finally **folder1**.

- Based on the list command at the end of the previous image we see that there still is a file called **myfile2.txt** that exists in our home directory so we will remove this as well, since we don't need it anymore:

**rm myfile2.txt**

**ls**




- This brings us to the end of **Files and Folders**.

- At this point you should give **Exercise 2** a try.