A Brief Intro to Linux

When you open a shell (i.e. the command line or terminal) in Linux you will start in your home directory (yeah Linux folks like to call them directories rather than folders)

The prompt is often a dollar sign $

Your home directory is indicated by ~ . To get to you home directory type (don't type the $)

```
$ cd ~
```

or

```
$ cd
```

to determine what directory, you are in type

```
$ pwd
```

Now let's create another directory in your home directory

$ mkdir *myFirstDir*

Note here that mkdir creates a directory and items in *italics* are user defined, you could have called the directory a111, project2, hat, etc. Any name is ok as long as it is not a linux command. To check if something is a linux command type (for example)

```
$ man myFirstDir
```

the output is:

```
No manual entry for myFirstDir
```

Now let's look at what is in your home directory using

```
$ ls
```

The result is

```
myFirstDir
```

if you type

```
$ls -l
```

You'll see more info, for example

```
drwxrwxr-x  2 moorej moorej       10 Sep 19 08:35 myFirstDir
```

This tells you that myFirstDir is a directory (i.e. the line starts with d), it tells about your read and write permissions and when the directory was created. Now move to that directory using

```
$ cd myFirstDir
```

Note that if you start typing myFi and then hit Tab it will fill in myFirstDir, if there are multiple options hit Tab twice and they will be displayed

Now let's create a file

```
$ touch myFirstFile.txt
```

Note that this creates an empty file. Linux does not care too much about extensions, you could have called this myFirstFile.dat, myFirstFile.log, or simply myFirstFile. Let's do that

```
$ touch myFirstFile.dat
$ touch myFirstFile.log
$ touch myFirstFile
```

Now type

```
$ ls
```

To see your files

```
myFirstFile  myFirstFile.dat  myFirstFile.log  myFirstFile.txt
```

if you only wanted to see files with extension .log you can use the wildcard *

```
$ ls *.log

myFirstFile.log
```

To edit these files you can use a text editor. Vim and Emacs are popular, I recommend emacs.

```
$ vim myFirstFile.txt
Or
$ emacs myFirstFile.txt
```

I'll leave it as an exercise for the user on how to operate these text editors. For now, use

```
$ cat 'hello' >> myFirstFile.txt
```

To add the text 'hello' to myFirstFile.txt

Now you can see the context with

```
$ more myFirstFile.txt
```

```
hello
```

If you only wanted to see the end of the file, try

```
$ tail myFirstFile.txt
```

If you wanted to continuously look at the end of the file (this is great for watching Abaqus's progress) use

```
$ tail -f myFirstFile.txt
```

Control c kills the tail -f command.

You can also look for text in the file using

```
$grep hello *
```

```
myFirstFile.txt:hello
```

This tells you that the text hello was in file myFirstFile.txt.
If you want to copy myFirstFile.txt to a new file called mySecondFile.txt use

```
$ cp  myFirstFile.txt mySecondFile.txt
```

If you want to remove myFirstFile.dat then you can use the rm command. BE CAREFUL rm will remove the file, it will be gone, it will not go to a recycle bin for later use. Linux will not ask you "are you sure you want to remove this file?" It will be very much gone. BE CAREFUL.

```
$ rm myFirstFile.dat
```

Now let's look at what we have

```
$ls
```

```
myFirstFile  myFirstFile.log  myFirstFile.txt  mySecondFile.txt
```

Another note of caution: if you copy a file to another file name (for example cp myFirstFile.log to mySecondFile.txt, it will overwrite mySecondFile.txt, the previous content of mySecondFile.txt will be gone, be careful)

This is probably a good time to recommend using git. Git will track all file changes and the history of the code you write. You can create an account of github or bitbucket and use these as a repo to store you tracked changes.

Finally let's look for a file

```
$ cd ..
```

This will go back one directory. Now let's look for myFirstFile.txt

```
$find . -name myFirstFile.txt
```

```
./myFirstDir/myFirstFile.txt
```

And there it is. The find command can take a long time, so try to get as close to the directory of the file you are looking for as possible.

Finally, to exit the terminal use

```
$ exit
```

Linux is super powerful, and you can do much much more than is discussed here, but these commands will get you going. The learning curve can be slow, but many find they prefer the control on the command line over GUI point-and-click after a while. Have fun.