**PART 1**

I have used the Terminal before navigating between directory and files, while in this lesson, I learnt to navigate files and also use cd to move back and forward between directories. I think using cd to go back to home directory and cd .. to move back one directory is quite helpful. While removing files in a directory, the “interactive” flag –i really helps since adding –i will ask us to confirm before deleting files. nano is also new to me, but this text editor is simple to use to write a file.

I think the most helpful thing I learn from working with files and directories is that instead of manually create new folder and files, the terminal can simply do all these things by writing short commands. For example, mkdir makes a new directory, mv can changes the files name, and cp copies the file. Also, current directory can be represented by . as the second parameter. The Unix Shell has very easy to use command but also considers the details while using it. For example, from the sort command I know that sort command itself sort its content alphabetically, while adding –n flag sort its content numerically, so I should consider using the leading zeros sometimes.

In addition, | allows the shell to print the output with different criteria. The for loop in Unix Shell is similar to what I learnt in python. The for loop operate on lists of items and repeat a set of commands for every item in a list, and its really flexible to just write one for loop and have both print output and saved file.

I feel like learning shell scripting is easier than learning Python, because there is not many rules. I can just do it from bash and working with files and getting data from one program into another, and don’t need to install anything. For example, I can write Python and SQL scripts from Unix shell. One of the interesting features it has is the tab completion, where I just need to press tab so that the shell will automatically gives me the name of the file I want to write.

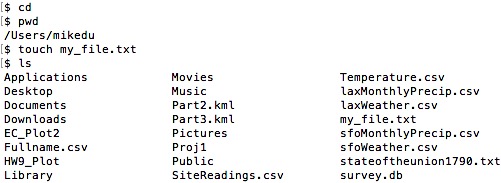
Even though the Unix shell does not have a trash bin that we can recover deleted files from, it records each command and I could easily access it from history. The wildcard expression is similar to regular expressions while I am not sure how wildcard expression works if I want to express multiple file names or extensions. The most common ones are \* which matches zero or more characters and ? which only matches a single character. There are few commands and symbols to remember, but they are all pretty straightforward. For example, >> append the content of one or more files to another file and saved it in the directory. grep and find to find files and things in files.

**PART 2**

**In Working With Files and Directories (lesson 03)**

* Creating Files a Different Way

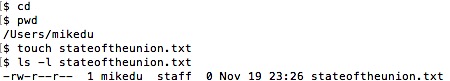
1. The touch command creates a new empty text file called my\_file.txt in my home directory. When I look at my home directory using ls, the file I created shows up.



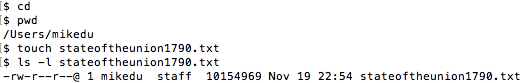
1. Use ls -l to inspect myfile.txt, the file is 0 in size.



1. I use touch command to create a file with the name already exist in another directory, and the touch command creates a new empty text file with the same name.



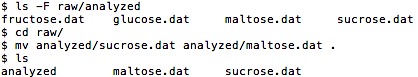
I then use touch command to create a file with the name exist in the home directory, and the touch command keeps the size of the original file and only updates the date last modified of the file.



So touch command can be use to create a file when the file doesn’t exit or when I want to update the timestamp of an existing file the directory.

* Moving to the Current Folder

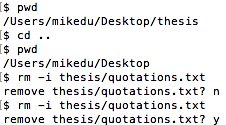
analyzed is the directory of sucrose.dat and maltose.dat currently in, so mv keeps the filenames of these two files but put at somewhere new, . means to put the file in the directory name we just specified cd raw/



* Using rm Safely

When we type rm -i thesis/quotations.txt, the shell asks whether we are sure about removing the file.

If we are concerned about what we might be deleting, adding the “interactive” flag -i to rm will ask us for confirmation before each step, so that we won’t delete file accidentally and cannot recover the file.



**In Pipes and Filters (lesson 04)**

* What Does sort -n Do?

sort command sort its content alphabetically, while adding –n flag specify that the sort is **numerical** instead of alphabetical.

* What Does < Mean?

< redirects its input

$ wc -l mydata.dat gets a command line parameter telling it what file to open.

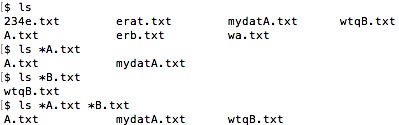


$ wc -l < mydata.dat, wc doesn’t have any command line parameters, so it reads from standard input, but we have told the shell to send the contents of mydata.dat to wc’s standard input.



* Wildcard Expressions

1. \* matches zero or more characters in a filename, so \*A.txt \*B.txt matches all files ending in A.txt or B.txt.

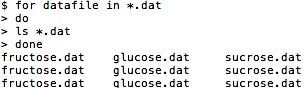


**In Loops (lesson 05)**

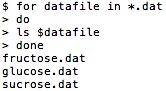
* Variables in Loops

The first for loop is for each file ending with .dat, list all the files ending with .dat.

Since there are total three files ending with .dat, the loop returns three files for three times, so each file is repeated three times.



The shell starts by expanding \*.dat to create the list of files it will process. This for loop is for each file ending with .dat, list that file. There are three files in total, so loop three time returns one file each time.

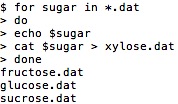


* Saving to a File in a Loop - Part One

**Prints fructose.dat, glucose.dat, and sucrose.dat, and the text from sucrose.dat will be saved to a file called xylose.dat.**

The shell starts by expanding \*.dat to create the list of files it will process. The loop body then executes two commands for each of those files. The first, echo, since the shell expands $sugar to be the name of a file, echo $sugar just prints the name of the file.

cat stands for “concatenate” and prints the contents of each file one after another. The greater than symbol, >, tells the shell to redirect the contents of the file to a file called xylose.dat instead of printing it to the screen. The temporary file is created in the same directory. The final output of xylose.dat only contains the content of sucrose.dat.



* Saving to a File in a Loop - Part Two

**All of the text from fructose.dat, glucose.dat and sucrose.dat would be concatenated and saved to a file called sugar.dat.**

The shell starts by expanding \*.dat to create the list of files it will process. The loop body then executes the commands for each of those files. cat prints the contents of each file one after another. >> tells the shell to append the contents of $sugar to a file called sugar.dat instead of printing it to the screen. The final output of sugar.dat contains the content of each \*.dat file.



**In Shell Scripts (lesson 06):**

* Why Record Commands in the History Before Running Them?

The shell always adds commands to the log before running them so that the history contains all the commands in the session, It prints the history list, so that one can recall and edit particular commands and for setting parameters such as the number of past commands to retain in the list, and also allows one to search command.

* Show me screenshots that you followed along with making Nelle’s Pipeline:

Creating a Script. Also attach the shell script that you created in the process.

