NE 24 - Homework #1
Chris Lamb
6 February, 2017

Section 1: Shell Files and Directories

1. If *pwd* displays */Users/thing*, what will *ls ../backup* display?

   Answer: ../backup: No such file or directory

2. For a hypothetical filesystem location of */home/amanda/data/*, select each of the below commands that Amanda could use to navigate to her home directory, which is */home/amanda*

   Answer: cd /home/amanda *or* cd ~ *or* cd .. *or* cd

3. If *pwd* displays */Users/backup*, and *-r* tells *ls* to display things in reverse order, what command will display:
   pnas-sub/ pnas-finals/ original/

   Answer: Either ls -r -F or ls -r -F /users/backup but not ls pwd

4. What does the command *cd* without a directory name do?

   Answer: It changes the working directory to the user's home directory.

5. What does the command *ls* do when used with the *-s* and *-h* arguments?

   Answer: When s is used with ls, it shows the sizes of the files in blocks and when h is used with ls, it turns the sizes in a human readable format.

Section 2: Creating Things

1. Suppose that you created a *.txt* file in your current directory to contain a list of the statistical tests you will need to do to analyze your data, and named it: *statstics.txt*

   After creating and saving this file you realize you misspelled the filename! You want to correct the mistake, which of the following commands could you use to do so?

   Answer: cp statstics.txt statistics.txt

2. What is the output of the closing *ls* command in the sequence shown below?

```
$ pwd
/Users/jamie/data
$ ls
proteins.dat
$ mkdir recombine
$ mv proteins.dat recombine
$ cp recombine/proteins.dat ../proteins-saved.dat
$ ls
```

Answer: recombine

3. Jamie is working on a project and she sees that her files arent very well organized:

```
$ ls -F
analyzed/ fructose.dat raw/ sucrose.dat
```

The *fructose.dat* and *sucrose.dat* files contain output from her data analysis. What command(s) covered in this lesson does she need to run so that the commands below will produce the output shown?

```
$ ls -F
analyzed/ raw/
$ ls analyzed
fructose.dat sucrose.dat
```

Answer: mv fructose.dat /analyzed
mv sucrose.dat /analyzed

4. What does *cp* do when given several filenames and a directory name, as in:

```
$ mkdir backup
$ cp thesis/citations.txt thesis/quotations.txt backup
```

What does *cp* do when given three or more filenames, as in:

```
$ ls -F
intro.txt methods.txt survey.txt
$ cp intro.txt methods.txt survey.txt
```

Answer: When given multiple filenames, it will copy and move those multiple files into the chosen directory. The second command gives an error because it is trying to move the files to a folder but no folder was given.

5. The command *ls -R* lists the contents of directories recursively, i.e., lists their sub-directories, sub-sub-directories, and so on in alphabetical order at each level. The command *ls -t* lists things by time of last change, with most recently changed files or directories first. In what order does *ls -R -t* display things?

Answer: Typing ls -R -t will give the full directories in order alphabetically and in order of most recently edited.

Section 3: Pipes and Filters

1. If we run *sort* on this file:

```
10
2
19
22
6
```

the output is:

```
10
19
2
22
6
```

If we run *sort -n* on the same input, we get this instead:

```
2
6
10
19
22
```

Explain why *-n* has this effect:

Answer: The first time it sorts, it goes through the first digit and then goes through the number as a whole.

2. What is the difference between:

```
wc -l < mydata.dat
```

and:

```
wc -l mydata.dat
```

Answer: The first one will read the data from the standard input and send the output to mydata.dat while the second one will just take mydata.dat as the input.

3. What is the difference between:

   ```
   echo hello > testfile01.txt
   ```

   and:

   ```
   echo hello >> testfile02.txt
   ```

   Hint: Try executing each command twice in a row and then examining the output files.

   Answer: The first one will only place the new data in the file and replace the old data while the second one will continuously add hello as many times as you put the command in.

4. In our current directory, we want to find the 3 files which have the least number of lines. Which command listed below would work?

   Answer: wc -l * | sort -n | head -3

5. The command *uniq* removes adjacent duplicated lines from its input. For example, if a file *salmon.txt* contains:

   ```
   coho
   coho
   steelhead
   coho
   steelhead
   steelhead
   ```

   then *uniq salmon.txt* produces:

   ```
   coho
   steelhead
   coho
   steelhead
   ```

   Why do you think *uniq* only removes adjacent duplicated lines? (Hint: think about very large data sets.) What other command could you combine with it in a pipe to remove all duplicated lines?

   Answer: Its easier to check if the current line is the same as the next or previous line than it is to compare it to the whole data set especially when the whole set is giant. To remove all of the same lines in the whole dataset, organize the set alphabetically with sort and then use uniq.

6. A file called *animals.txt* contains the following data:

```
2012-11-05,deer
2012-11-05,rabbit
2012-11-05,raccoon
2012-11-06,rabbit
2012-11-06,deer
2012-11-06,fox
2012-11-07,rabbit
2012-11-07,bear
```

What text passes through each of the pipes and the final redirect in the pipeline below?

```
cat animals.txt | head -5 | tail -3 | sort -r > final.txt
```

Answer: The first pipe takes the first 5 lines then the next pipe takes the last three of the remaining 5. The third pipe reverses the order of the lines and makes a new file called final.txt and outputs the result in there.

7. The command:

```
$ cut -d , -f 2 animals.txt
```

produces the following output:

```
deer
rabbit
raccoon
rabbit
deer
fox
rabbit
bear
```

What other command(s) could be added to this in a pipeline to find out what animals the file contains (without any duplicates in their names)?

Answer: The entire command would be: cut -d , -f 2 animals.txt | sort n | uniq