COMP 206, Fall 2018, Assignment 1

Objective

To provide you with your first experience programming the Linux shell. Try out important commands for working with files and data. Ensure that you can create a shell script that precisely implements our given **specifications**.

Getting Started

Obtain and unzip the provided file, COMP206Fall2018_Assign1_Provided.zip, which holds all of the test data that you'll need to do the problems. Here is an example set of Linux commands that will do the job:

```
$ mkdir COMP206_A1
$ cd COMP206_A1
$ mv ~/Downloads/COMP206Fall2018_Assign1_Provided.zip .
$ unzip COMP206Fall2018_Assign1_Provided.zip
(unzip output removed for space)
$ ls
COMP206Fall2018_Assign1_Provided.zip  Q1  Q2  Q3
```

You will now continue to complete each part by creating solution files within the COMP206_A1 directory. Pay very close attention to required file names **in bold**, because we require that you hand in files with precisely the same name to test your code automatically.

How to Hand-in

Please make sure that you've run all of the tests mentioned in this document, plus some more that you can think of for each script. We will always run your code on at least one situation not listed in the document, so you can never hard-code the specifics of the given data, but rather ensure you've written a general program that implements the specification. Ensure your code is understandable if opened by a TA (comments are a good idea). Ensure that your code runs on mimi.cs.mcgill.ca or the Trottier lab machines (they are identical, so either is fine).

Submit a single zip file to My Courses, through the Assignment 1 submission folder. Create this zip file with the command:

$ zip A1_solutions.zip **q1_written_answers.txt q2_encrypt.bash q2_decrypt.bash q3_image_sorter.bash**

The submission deadline is 23:59 Monday, September 24th (note one day later than planned because I've been late handing out A1 – all future due dates will be Sundays).

Good luck!

Question 1: Using the Linux Terminal to Browse Animals (20 points)

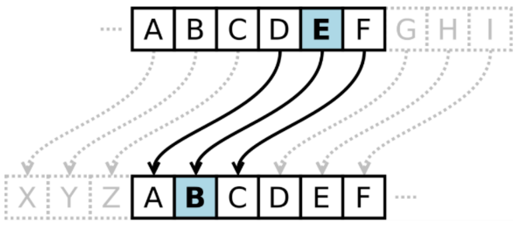
The Q1 folder contains several files that we want you to examine using only the Linux terminal. You are not supposed to use any other operating system features, especially not file browsers and window-based editors, for this question only.

Using BASH shell commands, answer these questions about the animals in the Q1 folder and explain the shell commands you executed to find the answer. While most questions can be answered simply by viewing the files and computing things in your head, try to automate as much as you can: more points for more complete automation. Write your responses in **q1_written_answers.txt**.

For example, the question "How many animals are there?" would be answered "8: I discovered this by running ls *.dat | wc which counts the number of animal data files."

a) List all the weights, in increasing order (ignore names here).
b) List all the lengths, in decreasing order (ignore names here).
c) Which is heavier, the hippo or the elephant?
d) An unfortunate fish swam into the crocodile pen, please increase its weight by 2 kg.
e) The giraffe is annoyingly tall, please remove all traces of it from our zoo.

Question 2: Sending Secret Messages (30 points)



The first "computers" were human beings with the job of encrypting (scrambling messages to hide their contents) and decrypting (reversing the process to read the message). One of the simplest encryptions is called "Codebook", and it simply means replacing the letters of the alphabet using a pattern that's written down in a codebook. The Q2 folder contains a sample codebook.txt file, shown here:

```
$ cat Q2/codebook.txt
cijklmpqrstxyzabnodefuvwgh
CDEMNOPQRSTGHUVXYZAIJKLWBF
$
```

It has one line for the pattern of small letters and a second for the pattern of capitals. Each line is 26 characters long and indicates a re-mapping of each input letter based on it's position in the standard alphabet.

Ex1: An "a" in the input is the first letter in the alphabet, so encrypting replaces it by the first letter in the first line of the codebook.txt file, which is "c". While decrypting, "c" goes back to "a".

Ex2: An "H" in the input is the 8th letter in the alphabet, so encrypting replaces it by the 8th letter in the second line of the codebook.txt file, which is "Q". While decrypting, "Q" goes back to "H".

You must write two BASH programs:

a. **q2_encrypt.bash**: Reads the codebook file provided as its first argument, reads the original message file provided as the second argument and prints the encrypted message on standard output.
b. **q2_decrypt.bash**: Reads the codebook provided as the first argument, applies the decryption on the encrypted file provided as the second argument and prints the recovered original message on standard output.

Sample Program Output:

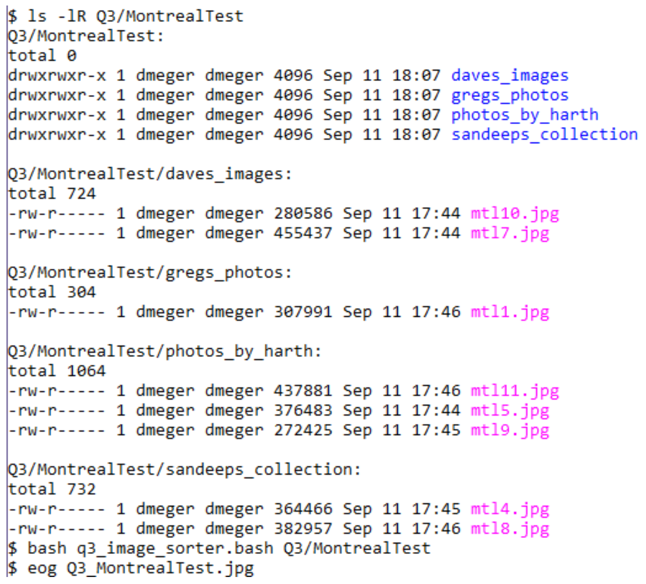
```
$ bash q2_encrypt.bash Q2/codebook.txt Q2/test_input1.txt > Q2/test_input1_encrypted.txt
$ cat Q2/test_input1_encrypted.txt
Qlxxa vaoxk.

$ bash q2_decrypt.bash Q2/codebook.txt Q2/test_input1_encrypted.txt
Hello world.

$ bash q2_decrypt.bash Q2/codebook.txt Q2/secret_message.txt
```

Question 3: Sorting Vacation Photos (50 points)

As Computer Science has evolved, it now covers many more useful and enlightening applications, such as helping us view photos of cats on the Internet and organize our tourist photos. Suppose you return from a trip having spread your images across multiple directories within the Q3 folder and you want to re-create the timeline in chronological order.

Write a BASH program, called **q3_image_sorter.bash** that creates a timeline image that summarizes all photos within the directory (and recursive sub-directories) of the first and only command-line argument.

- You can assume all input files have ".jpg" extension.
- The time-line must be sorted with the oldest on top to newest at the bottom, using the modification date of each file (to match our example output, ensure you do not modify the images yourself after you've unzipped)
- Call the Linux program "convert -append" to create the timeline image, which takes a list of N image name arguments. The first N-1 are treated as inputs, to be read and stacked vertically in the same order that the files are listed, and the final (Nth) argument is the output filename.
    - If you do not have convert on your home system and want to develop there, you need to install it, with a command such as "$ sudo apt-get install imagemagick"
- The time-line image must be created in the shell's present working directory and should match the path given as the first argument, but with "_" (underscore) characters replacing any slashes. For example, the command "$ bash q3_image_sorter.bash Q3/SimpleTest" needs to output the file "Q3_SimpleTest.jpg".
- We recommend using shell command "eog" to view the files, but many other methods work too. If you are doing your work remotely on mimi, transfer the image to yourself to view it using scp.



```
$ ls -lR Q3/MontrealTest
Q3/MontrealTest:
total 0
drwxrwxr-x 1 dmeger dmeger 4096 Sep 11 18:07 daves_images
drwxrwxr-x 1 dmeger dmeger 4096 Sep 11 18:07 gregs_photos
drwxrwxr-x 1 dmeger dmeger 4096 Sep 11 18:07 photos_by_harth
drwxrwxr-x 1 dmeger dmeger 4096 Sep 11 18:07 sandeeps_collection

Q3/MontrealTest/daves_images:
total 724
-rw-r----- 1 dmeger dmeger 280586 Sep 11 17:44 mtl10.jpg
-rw-r----- 1 dmeger dmeger 455437 Sep 11 17:44 mtl7.jpg

Q3/MontrealTest/gregs_photos:
total 304
-rw-r----- 1 dmeger dmeger 307991 Sep 11 17:46 mtl1.jpg

Q3/MontrealTest/photos_by_harth:
total 1064
-rw-r----- 1 dmeger dmeger 437881 Sep 11 17:46 mtl11.jpg
-rw-r----- 1 dmeger dmeger 376483 Sep 11 17:44 mtl5.jpg
-rw-r----- 1 dmeger dmeger 272425 Sep 11 17:45 mtl9.jpg

Q3/MontrealTest/sandeeps_collection:
total 732
-rw-r----- 1 dmeger dmeger 364466 Sep 11 17:45 mtl4.jpg
-rw-r----- 1 dmeger dmeger 382957 Sep 11 17:46 mtl8.jpg
$ bash q3_image_sorter.bash Q3/MontrealTest
$ eog Q3_MontrealTest.jpg
```