

Assignment 1

COMP 206, Fall 2016

Due: Monday October 3rd, (23:59) via My Courses

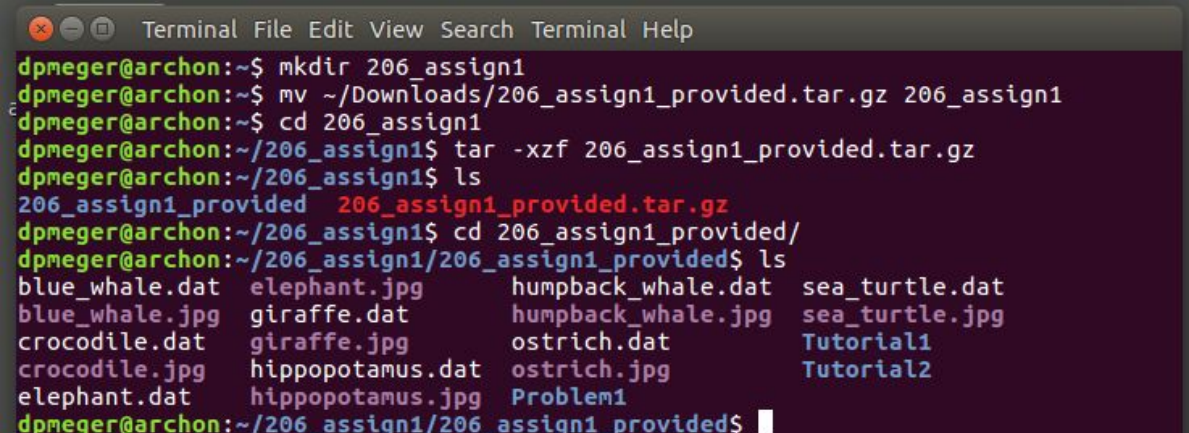
100 marks total

Please read the entire pdf before starting.

You must complete this assignment individually. Discussion and sharing ideas with peers or the TAs is accepted, but you must code and test each solution alone.

Read the question descriptions carefully and follow the filename conventions specified for your submission. We will mark your submissions using an automated script that depends on you using the right file names (including capitalization, use of underscores, etc).

Getting Started: Download the provided archive file from My Courses and extract it in the directory where you will work on the assignment.



```
Terminal File Edit View Search Terminal Help
dpmeger@archon:~$ mkdir 206_assign1
dpmeger@archon:~$ mv ~/Downloads/206_assign1_provided.tar.gz 206_assign1
dpmeger@archon:~$ cd 206_assign1
dpmeger@archon:~/206_assign1$ tar -xzf 206_assign1_provided.tar.gz
dpmeger@archon:~/206_assign1$ ls
206_assign1_provided  206_assign1_provided.tar.gz
dpmeger@archon:~/206_assign1$ cd 206_assign1_provided/
dpmeger@archon:~/206_assign1/206_assign1_provided$ ls
blue_whale.dat  elephant.jpg  humpback_whale.dat  sea_turtle.dat
blue_whale.jpg  giraffe.dat  humpback_whale.jpg  sea_turtle.jpg
crocodile.dat  giraffe.jpg  ostrich.dat  Tutorial1
crocodile.jpg  hippopotamus.dat  ostrich.jpg  Tutorial2
elephant.dat  hippopotamus.jpg  Problem1
```

Warm-Up Exercises

Not for marks. The TAs or instructor will solve these for you in office hours, if you wish.

Tutorial 1

When using Unix files to perform useful tasks, like keeping a journal or TODO list, it's often handy to keep date-stamped archives, so that we can remember what was going on in the past. Create a script **tut1.bash** to do this conveniently for yourself, following the spec:

- Run with command `$ bash tut1.bash filename`
- **Part a)** Results in a dated copy of the file named with a time-stamp. Eg. if filename was `file1.txt` it would create a copy such as `2016-09-19_file1.txt`.
- **Part b)** Challenge: To make it a bit harder, see if you can get it so that the date appears after the name of the file, but before its extension, such as `file1_2016-09-19.txt`.

Hints:

- The command **date** will be useful, and check out its formatting string arguments.
- The commands **basename** or **cut** are helpful to work with filenames

Tutorial 2

When working with UNIX folder structures, we often want to make changes that affect a large number of files or directories, such as copying all files from “Source” folders into corresponding “Destination” folders, one by one. Create a script **tut2.bash** that solves this problem for one commonly occurring case:

- Run with command `$ bash tut2.bash source_directory destination_directory`
- Assume `source_directory` and `destination_directory` have the same number of sub-folders, but that their names differ. E.g. The Tutorial2 folder provided.
- Result is the files within each sub-folder of the source directory are copied to the corresponding sub-folders in the destination directory. In the example:
 - `Source/S1/1.txt` is copied to `Dest/Dest1/`
 - `Source/S2/2.txt` is copied to `Dest/Dest2/`, etc...
- Do not make any additional assumptions about the folder structure of Source or Destination (i.e., your code should be general, not just handle this one example).

1. Permission to Answer (10 marks)

Part a) (5 marks) The provided “**Problem1**” directory contains a file named “**answer.txt**”, which is a text file that holds the answer. Try to show the contents of a file with cat. For example:

```
$ls Problem1/  
answer.txt  
$cat Problem1/answer.txt
```

(Hint, you are NOT supposed to see the contents of this file yet!)

Write a description of the outcome of this command in your solution text file, **q1.txt**. Your answer can be just a few words, but make sure to include the other commands that you ran in order to diagnose any failures along with their output.

Part b) (5 marks) Fix the problem with the “**answer.txt**” file so that you are now successfully able to use the “cat” command above to view the file contents. Copy the sequence of commands that you used and their outputs into your solution text file, **q1.txt**. Make sure to copy the contents of the file into your written answer, to prove it worked.

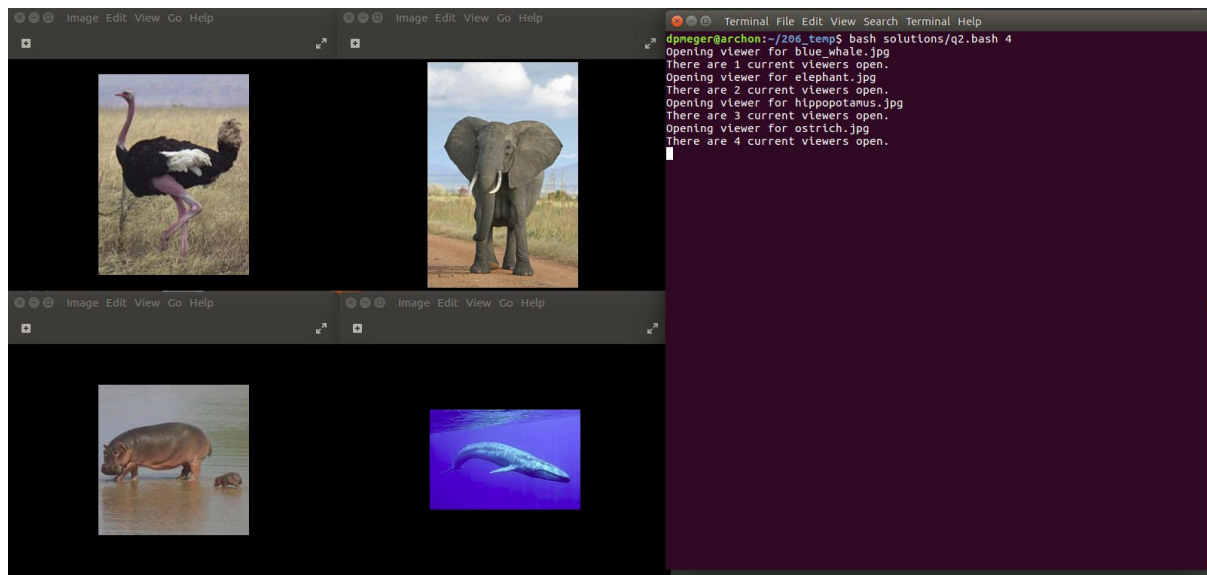
2. Your Personal Zoo (40 marks)

One of the most common and lucrative uses of computers today is spending hours looking at cute animal photos. We have provided 8 of these in the archive file, and you must write a BASH script, **q2.bash**, that displays N of them at a time (N is a command-line argument), using Linux's **eog** image viewer, cycling endlessly until your script is killed.

Usage: `$ bash q2.bash N` -> Quickly opens N images for viewing. No more and no less.

- The order of images shown must be “fair”: that is you must show each image once before repeating any, each twice before repeating any three times, etc.
- When the user closes an image viewer, a new image must be shown in less than 2 seconds, ensuring optimal time-wast... entertainment is achieved, and keeping exactly N viewers open at all times.
- Your script must only end when interrupted with Ctrl-c
- Hint: Ensure to use the “eog -n” option, which makes sure each image you open uses an independent (new) process.
- Hint: ps is a helpful command to determine how many viewers are open currently.
- Helpful tip: After sending ctrl-c to your script, your eog processes will likely stay open. To prepare for a new run of your script, run “\$ killall eog”.

Sample session:





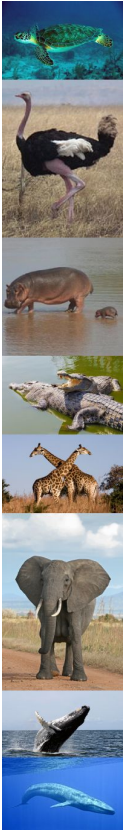
3. Sort the Animals (50 marks)

Along with each animal image from Question 2, we have also provided a “.dat” files with a few properties of each, like their weight and height. You must create a BASH script, **q3.bash** that sorts the animals based on a property given on the command line:

- \$ bash q3.bash alpha -> sorts by increasing alphabetical order of animal name
- \$ bash q3.bash weight -> sorts by increasing weight
- \$ bash q3.bash length -> sorts by increasing length

You must output your result in 2 ways. First, the ordered list must be printed as standard output (shown on the terminal), one name per line. Second, you must combine all of the animal images into one file called “result.jpg”, with the ordering of animals shown respecting the requested sort order. Look at the “convert -append” command for producing the joined-image output.

The table below shows desired results when run with each arguments:

Alpha	Weight	Length
<div> <div>blue_whale</div> <div>crocodile</div> <div>elephant</div> <div>giraffe</div> <div>hippopotamus</div> <div>humpback_whale</div> <div>ostrich</div> <div>sea_turtle</div> </div> 	<div> <div>ostrich</div> <div>sea_turtle</div> <div>crocodile</div> <div>giraffe</div> <div>hippopotamus</div> <div>elephant</div> <div>humpback_whale</div> <div>blue_whale</div> </div> 	<div> <div>sea_turtle</div> <div>ostrich</div> <div>hippopotamus</div> <div>crocodile</div> <div>giraffe</div> <div>elephant</div> <div>humpback_whale</div> <div>blue_whale</div> </div> 

--	--	--

Submitting your solutions:

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.

One last time, be very careful that your code runs on mimi or the lab Trottier machines (they are identical, so either is fine). Code that runs on your own laptop but does not work at Trottier may receive as low as zero since we are simply not able to investigate the differences for each student and will have no way to confirm your code is functional.

Submit a single zip file to My Courses called “**A1_solutions.zip**” that contains 3 files:

1. **q1.txt**
2. **q2.bash**
3. **q3.bash**

You can create your hand-in archive with this file with the command:

```
$ zip A1_solutions.zip q1.txt q2.bash q3.bash
```

The submission deadline is 1 minute before midnight on Monday, October 3rd.

Good luck, and do use Office Hours and Slack if you get stuck!