Week 2: Problems

For these problems, bring your solutions and results with you to your sessions so you can discuss them with your mentor.

Setup

For these, use the image called `pementorship/w2_problems`. As a reminder, you just need to run:

```
$ sudo docker run -it pementorship/w2_problems
```

You might need to `docker pull` it first.

Questions

1. **man ls**: Run `man ls`. Construct invocations of `ls` that will output:

- The author/last editor of the file
- File sizes in KB instead of bytes
- inode (or index node) numbers of the files

2. **seq**: Linux has a command called `seq` that takes in two integers and generates every number between them. Read `seq --help` and the `man` page of `seq` to learn more about this program. Once you're done, generate a file called "`numbers.txt`" containing every even number between 10 and 200 (inclusive on both ends) using `seq` (hint: much like how we used | to pipe between programs, you can use > to pipe to a file, just supply the file name on the right side).
    1. How would you do this for the odd numbers?
3. **find words**: Many Linux installations come with a file containing a (mostly) full English dictionary with no definitions. It's just a list of English words. It lives in the file `/usr/share/dict/words`. For this problem:
    1. Obtain every single word with the ending "-tion"
       HINT: grep will be useful here, with the `-E` flag. If you're not sure what to do, read `man grep`, and take a look at this tutorial on regular expressions, or regexes. Trust me, they'll be useful in the future.
    2. How many words contain the "-tion" ending in the dictionary?
       HINT: Use the `wc` (short for word count) program. Read the man page to figure out how to get line counts.
4. **CSV madness**: We've provided you with a CSV in your home directory called `people.csv` (just `ls` when you log in and you'll see it). If you look at it with something like `less`, you'll see it's a large file with fake people's data. It contains their extremely precise height in cm, full name, and split out first and last names. CSV's are quite simple files. Imagine this as a massive table, where each column in a row is separated by commas, and the rows are each new line in the file. Notice the first row is used to give each column a name. In this exercise, you'll compute a lot of things based on this data, so it'll be a multi-parter:
    1. Look over the file. How many rows (excluding the header) does this contain? How do you get and print this number in bash?
       HINT: Take a look at 3.2 for how to count lines. You might also want to take a look at `tail` for how to exclude the header. You'll need it for later.
    2. Next, how do you get bash to print out only the full names in the file?
       HINT: you'll need to make use of the awk program. Read the man page and search online for how it can help you in this task.

3. In the home directory, we also provided a program called `avg.py` (feel free to read the code with `less`, it's a relatively simple piece of python). This computes the average of all the numbers you feed in from standard input. Test it out by running `./avg.py`, typing one number in each line, and pressing Ctrl+D (for EOF) once you're done. Does it work as expected? With what edge cases does this program fail?
4. Now, using what you've just learned and pipes (`|`), compute the average height of the people on the file.
5. What's the most common first name in this dataset?
6. Are there any repeated full names? How do you find them?
7. How many unique first names are there in this dataset that start with the letter A?
8. **EXTRA**: can you figure out what distribution I used to generate people's heights? You might want to use something like numpy and matplotlib in python to see the data.