Shell Scripting 2020 fourth week

Format of the learning diary and scripts

The tasks are returned in moodle.

Return the learning diary in **PDF format**. Preamble the document with the following information:

- Name and student number

The learning diary should contain the answers to questions made in the "Put in your answer" parts. Also example output can be put into the answer.

Include the code in week's directory as a tarball with scripts named with the task name as "task#.sh", i.e. for task 2 the name would be "task2.sh".

Deadline for this set of tasks is Monday 30th of November 2020 and I will release the next set of tasks on 23th of November.

If you run into problems, try to read the man pages and try google. If you are still stuck with the task send me a msg in moodle and I will answer during normal office hours (9-17).

ASCII art (31)

This is the shortcat:

Shortcat

```
   /\___/\
  /       \
 |  #    # |
 \     @   |
  \   _|_ /
  /       \______
 / _______ ___   \
 |_____   \   \__/
  |    \__/
  |       |
  /        \
 /   ____   \
 |  /    \  |
 | |      | |
/  |      |  \
\__/      \__/
```

Shortcat is saved in shortcat.txt

Write a script called longcat.sh which accepts a single, numerical (greater than 0) command line parameter. Requirements:

- invoking longcat.sh 1 will print the above shortcat

- a numerical parameter larger than 1 will extend the stomach of the cat to match the number of requested lines
- shortcat's stomach is just one row long, and it begins below the front paws and above the rear paws
- passing anything else than a numerical parameter will print a helpful message and exit with a status of 1

`Hints: head and tail.`

Put in your answer:

- Present your script.

Example output for longcat.sh 3:

```
Longcat 3

  /\___/\
 /       \
|  #    # |
\     @   |
 \   _|_ /
 /       \______
/ _______ ___   \
|_____   \   \__/
 |    \__/
 |       |
 |       |
 |       |
 /        \
/   ____   \
|  /    \  |
| |      | |
/  |      |  \

\__/      \__/
```

Plotting (32)

Plotting

Perhaps one of the most difficult tasks in practical data analysis is the correct representation of data. Unfortunately, the drawing speed will also become an issue with larger data sets. Drawing a simple time series graph with LibreOffice can easily take 20 minutes if one has 20k data points.

In those cases, it is time to call for backup. A venerable combat veteran who will answer your summons is GnuPlot (http://www.gnuplot.info/index.html). GnuPlot is lightning fast in drawing most graphs, but very, very picky about its commands. Like a good commanding officer, one has to be very precise in telling GnuPlot precisely what to do.

Then again, as shell programmers, you are already familiar with cryptic, dangerous, unforgiving, powerful command invocations. You will have no problems with GnuPlot... Given sufficient practice. GnuPlot is usually learned through a series of examples (http://gnuplot.sourceforge.net/demo/) which produce example graphs. We will do the same.

First Gnuplot script

You can enter a gnuplot shell by typing the gnuplot command. To quit the shell type quit

A better way to use gnuplot is gnuplot script so you can easily reproduce a sequence of gnuplot commands.

Let your first gnuplot script be test.gnuplot, that plots the cosinus of x

you can modify the ranges printed for x and y with xrange and yrange respectively

test.gnuplot

```
set terminal dumb

set xrange [-5:5]

plot cos(x)
```

There are a lot of terminal you can use to produce (jpg, png, html, LaTeX, eps, etc..), try set terminal in a gnuplot shell to see the list of available terminal.

We pick dumb as it produce ASCII art

```
Try: gnuplot test.gnuplot
```

Modify your script to print two curves, one for cos(x), and the other for sin(x)

Plotting data from file

Create a script "create-random-data.sh" that takes one numerical argument n, and produces n lines with two random numbers. For example, create-random-data.sh 5 could produce:

```
7440 6271

16564 10906

31425 13410

13382 27114

11510 28041
```

Put in your answer:

- Present your solution in your report. Hint: look at shell variables for random numbers
- Create now a random-data.txt file with at least 100 lines using your "create-random-data.sh" script. Create a gnuplot script that will print that data points (x,y) from random-data.txt using the first random number as x and the second as y. Show your solution and the output of your program in your report.

Let's plot some real data points (33)

Let's plot some real data points

Gnuplot is a design for plotting, consequently it is important to prepare data with your newly obtained shell scripting skills to facilitate the task of gnuplot.

In this exercise, we want to reuse the data from the temperature files of Week 3 exercises.

For the month of November 2011, make a script that prints the date and the maximum temperature for each day of the month, separated by space, to a temporary file and then call a gnuplot script to generate an eps file "max-temps-2011-11.eps" showing the maximum temperatures for each day of that month in a linespoints format (check set style data in the documentation)

Add a title, labels for x and y-axes

Put in your answer:

- Present your solution.

`Hint: http://gnuplot.sourceforge.net/demo/timedat.html, for example with date format`

Let's put some context (34)

Let's put some context

Extend your script to add a third column with minimal temperature for each day.

Modify your gnuplot script to print on the same graph, both the minium and maximum temperatures

Extend your script to automatically create the eps file "min-max-temps-2011-11.eps" showing the minimum and maximum temperatures for each day of that month in a linespoints format

Put in your answer:

- Present your solution

Let's generalize (35)

Let's generalize

Do a script that takes as parameter a folder representing a month (example: ...../lost24/monitor/2011.10* for October 2011) and generate the according plot "min-max-temps-2011-10.eps"

Put in your answer:

- Present your solution

Let's make more refined commands (36)

Let's make more refined commands

Now, you should've completed your min-max-temperature script. It has a single argument format: a directory name. It will search through the contents of the directory, and parse through a great number of files to find either maximum and minimum temperatures.

We will write a wrapper script that presents a user interface to the, well, user. While tuning a graphical user interface can be extremely time consuming, shell scripts are much easier. No paper doll tryouts. No videotaping users.

The trick will be to use getopts, which takes care of almost all of the magic for you. But there is one thing that it will not do. By now you have read a great number of man pages.

You should've noticed that some programs have very different ways of accepting parameters. Due to historical reasons, there are (at least) three different types. From man ps:

```
This version of ps accepts several kinds of options:

1. UNIX options, which may be grouped and must be preceded by a dash.

2. BSD options, which may be grouped and must not be used with a dash.

3. GNU long options, which are preceded by two dashes.
```

getopts is limited to UNIX-style arguments, but even that will take us a long way. This tutorial will show you that way:

- http://wiki.bash-hackers.org/howto/getopts_tutorial

Make a script using getopts and switch that can take a few options:

- -c: for coldest temperatures
- -w: for warmest temperatures
- -b for both temperatures
- -a to produce ASCII output instead of eps
- -h for help

and produce the desired figure

Put in your answer:

- Present your solution