Exercises

We are about to create from command line the structure for a basic web application just with directories to contain things like HTML, CSS and JS. Our application also will have log directory which is frequently used to store a file to write on it application debugs and errors.

Exercise one

Open a terminal and create the following structure:

```
terminal-exercises-1
    src
        css
        html
        js
        log
        static
```

To check it out we can run ls with R option. It should look like this:

```
$ ls -R
bootstrap.sh         exercise.md          exercise.sh          terminal-exercises-1

./terminal-exercises-1:
src

./terminal-exercises-1/src:
css    html   js     log    static

./terminal-exercises-1/src/css:

./terminal-exercises-1/src/html:

./terminal-exercises-1/src/js:

./terminal-exercises-1/src/log:

./terminal-exercises-1/src/static:
```

Once created return to root folder /terminal-exercise-1 by **cd ..** up to its level

Exercise two

Create the following files; index.html, main.css app.js and app.log. Once created we can use **ls** command to take a look to what we've created so far.

```
$ ls -l
-rw-r--r--  1 jose  staff    0 Jun  2 16:19 app.js
-rw-r--r--  1 jose  staff    0 Jun  2 16:19 app.log
-rw-r--r--  1 jose  staff    0 Jun  2 16:19 index.html
-rw-r--r--  1 jose  staff    0 Jun  2 16:19 main.css
drwxr-xr-x  7 jose  staff  224 Jun  2 16:09 src
```

Exercise three

Move the files to its proper directory. They are self referential. Once moved it must look:

```
$ ls -R
./src/css:
main.css

./src/html:
index.html

./src/js:
app.js

./src/log:
app.log

./src/static:
```

Exercise four

Rename **static** dir to **tests**, which looks like this before to be rename

```
$ ls
css    html   js     log    static
```

Once renamed it must look

```
$ ls
css   html  js    log   tests
```

Now create three files called **user.test.js**, **app.test.js** and **store.test.js** inside test directory and these two **user.js** and **store.js** inside **js** folder. Once done it must look like this:

```
$ ls -R
./css:
main.css

./html:
index.html

./js:
app.js store.js user.js

./log:
app.log

./tests:
app.test.js store.test.js user.test.js
```

So far we must have this structure:

```
$ ls -R
terminal-exercises-1

./terminal-exercises-1:
src

./terminal-exercises-1/src:
css   html  js    log   tests

./terminal-exercises-1/src/css:
main.css

./terminal-exercises-1/src/html:
index.html

./terminal-exercises-1/src/js:
app.js   store.js user.js

./terminal-exercises-1/src/log:
app.log

./terminal-exercises-1/src/tests:
app.test.js   store.test.js user.test.js
```

Exercise five

Open from command line the index.html file and add this.

```
<p>Hi world</p>
```

Add this to css file

```
p { font-size: 1.75rem; }
```

Use cat command to show up what there's at index.html file. It must look like this.

```
Hi world
```

Exercise six

Create a directory called **tmp** and add a file called **app.cache.txt**. Once created, show it recursively and then remove **tmp** directory.

```
 ./tmp:
total 0
-rw-r--r-- 1 jose staff 0 Jun  3 09:56 app.cache.txt
css  html  js  log  tests
```

Exercise seven

Move to terminal-exercises-1/src. List directories by size.

`js tests css html log` List directories by size but this time in reverse order

`log html css tests js` Now it's time to list them by creation date.

`js tests log css html` Show them by creation date but in reverse order

`js tests log css html`

We need to list hidden files.

`. .. css html js log tests`

Clean the terminal with **clear** command. So is time to show the file size in human readable format. Let's say in Megabytes. Here we need to add a ls option called **block-size** in order to show the size on Megas. So you can use the **– help** command in order to find out how to use this option to get this output.

`drwxr-xr-x 3 jose staff 1M Jun 2 16:27 css drwxr-xr-x 3 jose staff 1M Jun 2 16:27 html drwxr-xr-x 5 jose staff 1M Jun 3 09:36 js drwxr-xr-x 3 jose staff 1M Jun 2 16:27 log drwxr-xr-x 5 jose staff 1M Jun 2 16:41 tests`

Advanced

We have an access log which logs the clients connections to our web server. It usually logs IP, the date, the HTTP verb which was used to connect, the protocol, status code, bytes sent to client, referer, and client user agent. We are going to need show some lines which match with some parameters we have.

For this situation in which we have a big unreadable file we can use a command called **grep**. The grep command is used to search text. It searches the given file for lines containing a match to the given strings or words. It is one of the most useful commands on Linux and Unix-like system.

```
$ grep -h usage: grep [-abcDEFGHhIiJLlmnOoqRSsUVvwxZ] [-A num] [-B num] [-C[num]] [-e pattern] [-f file] [--binary-files=value] [--color=when] [--context[=num]] [--directories=action] [--label] [--line-buffered] [--null] [pattern] [file ...]
```

We have two special characters called anchors which indicate the beginning **^** and the end **$** of a string. We can use them with grep command to try to find when a string starts with anything for instance to find what lines start with **2.** which would be **^2.**. And to find those which end with the same mark we need to find we would use **2.$**

With this in mind, we are going to investigate a little the options for this command in order to finish the exercises that follow.

Exercise

We need to show how many connections were this exact date: **19/Jun/2019:19:35:00**

```
31.13.115.20 - - [19/Jun/2019:19:35:00 +0200] "GET /contact HTTP/1.0" 302 497 "-" "facebookexternalhit/1.1 (+http://www.facebook.com
31.13.115.8 - - [19/Jun/2019:19:35:00 +0200] "GET /contact HTTP/1.0" 302 497 "-" "facebookexternalhit/1.1 (+http://www.facebook.com/
31.13.115.24 - - [19/Jun/2019:19:35:00 +0200] "GET /contact HTTP/1.0" 302 497 "-" "facebookexternalhit/1.1 (+http://www.facebook.com
31.13.115.21 - - [19/Jun/2019:19:35:00 +0200] "GET /contact HTTP/1.0" 302 497 "-" "facebookexternalhit/1.1 (+http://www.facebook.com
31.13.115.5 - - [19/Jun/2019:19:35:00 +0200] "GET /contact HTTP/1.0" 302 497 "-" "facebookexternalhit/1.1 (+http://www.facebook.com/
31.13.115.13 - - [19/Jun/2019:19:35:00 +0200] "GET /contact HTTP/1.0" 302 497 "-" "facebookexternalhit/1.1 (+http://www.facebook.com
31.13.115.17 - - [19/Jun/2019:19:35:00 +0200] "GET /contact HTTP/1.0" 302 497 "-" "facebookexternalhit/1.1 (+http://www.facebook.com
31.13.115.14 - - [19/Jun/2019:19:35:00 +0200] "GET /contact HTTP/1.0" 302 497 "-" "facebookexternalhit/1.1 (+http://www.facebook.com
```

We also need to show how many connections were from **iPhone** but this time we want to show the exact number.

```
926
```

Now we need to show how many connections were not from **iPhone** and also in number.

```
2793
```

We need to count how many lines start with this IP pattern: **212.**

```
4
```

And also how many lines end with this pattern: **error**

```
0
```