The Unix (Linux) Command Shell

To solve the following exercises we sugest that you read (and also try the exercises) of this excellent introduction to the Bash Shell by Ryan Chadwick. The commands (and corresponding command line options), operators and special symbols that you should be familiar with are the following:

```
absolute and relative paths:
* directories (/, . e ..)

shell special symbols:
* globs (^, ?, *, [], {})
* I/O redirection (>, >>, <, |)
* detach process I/O from shell (&)

basic commands:
* filesystem navigation (pwd, cd, tree)
* creating/removing files/directories (touch, mkdir, rmdir, rm, rm -rf)
* copy/moving files/directories (cp, cp -a, cp -r, mv)
* file manipulation (cat, more, less, head, tail, chmod, file)
* process management (ps, ps -aux, ps -u, jobs, top, kill -9)
* file searching (find, find -size -name -print, ls, ls -l, ls -lR)
* command searching and manual pages (which, man)
* finding/processing data in files (grep, grep -v, cut, cut -d -f, sort)
* editing files (sed s/X/Y/g ou s/X/Y/n, n=1,2...)
* I/O redirection ("<", ">", ">>", "|")
* archiving (tar, zip, gzip)
```

Open a shell terminal in your computer and solve the following exercises. For each exercise you should first try to anticipate the result based on what you know about the Bash commands involved. Check your predictions by running the commands and watching the results.

1. What directory are you in after executing each of the following commands? Check your guesses by running the command `pwd`.

```
$ cd ~
$ cd
```

home directory both

2. Use the `mkdir` command to create the following subtree in your current directory:

```
dir1
|-- dir2
|   |-- dir4
|   |   |-- dir6
|   |-- dir5
|-- dir3
```

3. Starting at the current directory, in what directory would you end up after running the following commands?

```
$ tree dir1
dir1
|-- dir2
|   |-- dir4
|   |   |-- dir6
|   |-- dir5
|-- dir3
$ cd dir1/dir2/dir4/dir6/../..
```

goes to last directory inputed in command, in this case dir6, then goes back 2 directories so it goes to dir2 (/../.. makes it go back 2)

4. Use the `touch` command to create empty files with the names as below within the subtree you created.

```
dir1
|-- dir2
|   |-- dir4
|   |   |-- g22.doc
|   |-- f1.txt
|   |-- h22.txt
    |-- g368.pdf
|-- dir3
    |-- f3a.txt
    |-- g56.doc
    |-- g3x.pdf
```

5. What is printed by the last two commands in this sequence?

```
$ tree dir1
dir1
|-- dir2
```

```
|   |-- dir4
|   |   |-- g22.doc
|   |-- f1.txt
|   |-- h22.txt
    |-- g368.pdf
|-- dir3
    |-- f3a.txt
    |-- g56.doc
    |-- g3x.pdf
$ find dir1/dir2 -name "[fg][35][4-7a-z].txt" -print
$ find dir1/dir3 -name "[fg][35][4-7a-z].txt" -print
```

nothing cuz theres 2 .txt

dir1/dir3/f3a.txt

6. What is the structure of the subtree with root at `dir1` after running the last command in this sequence?

```
$ tree dir1
dir1
|-- dir2
|   |-- dir4
|   |   |-- g22.doc
|   |-- f1.txt
|   |-- h22.txt
    |-- g368.pdf
|-- dir3
    |-- f3a.txt
    |-- g56.doc
    |-- g3x.pdf
$ rm -rf dir1/dir2
```

deletes dir2, therefore only d3 appears under dir1

7. Assume that at directory `dir3` you have the following scenario. Write the permissions of the 3 files in octal, indicate the owners of the files, their creation dates and size in bytes.

```
$ ls
f3a.txt g3x.pdf j52.docx
$ ls -l
total 28712
-rw-r--r--  1 lblopes  staff      1412 Dec 29 15:43 f3a.txt
-rw-r--r--@ 1 lblopes  staff  13923695 Dec 29 15:44 g3x.pdf
-rw-r--r--@ 1 lblopes  staff     13793 Dec 29 15:47 j52.docx
```

owners size date created

8. What permissions have "user", "group" and "others" over the file `doit` after you execute each of the following commands?

```
$ chmod 755 doit
$ chmod u-wx doit
```

read, write and execute permissions to the file owner, read and execute permission to Group and Others

write and execute permissions to the file owner

```
$ chmod go-rx doit
$ chmod 644 doit
```
remove read and execute permission for the group or others

read and write permission to owner, only read permission to group and others

9. What is printed by the last two commands in this sequence?

```
$ cat > trees.txt
pine:253:221:1.2
oak:144:123:0.9
birch:92:83:1.6
yew:65:63:4.3
alder:12:5:9.6
^D
$ cat trees.txt | cut -d ':' -f 1,4
$ cat trees.txt | cut -d ':' -f 1,4 | sort
```
remove tudo entre os :

igual ao de cima mas ordena-os por ordem alfabética

10. Consider the following file with a quote written in a single line (no newlines). What is the output of the final three commands in this sequence? Explain why.

```
$ cat > q1.txt
Three Rings for the Elven-kings under the sky, -->
Seven for the dwarf-lords in their halls of stone, -->
Nine for Mortal Men doomed to die, -->
One for the Dark Lord on his dark throne, -->
In the Land of Mordor where the Shadows lie. -->
One Ring to rule them all, One Ring to find them, -->
One Ring to bring them all and in the darkness bind them -->
In the Land of Mordor where the Shadows lie.
^D
$ cat q1.txt | sed 's/Ring/Sword/g' > q2.txt
$ cat q2.txt | grep -v Mordor
$ wc -l q2.txt
```
creates q2.txt, copy of q1, except it exchanges all Ring for Sword

prints q2 but except every line that contains word Mordor

prints the number of total lines

11. What are the results of the `diff` commands in the sequence below?

```
$ cat > f1.txt
I don't know half of you
half as well as I should like;
and I like less than half of you
half as well as you deserve
^D
$ sed 's/half/two\ thirds/g' < f1.txt > f2.txt
$ diff f1.txt f2.txt
$ sed 's/like/do\ not\ like/g' < f1.txt > f2.txt
$ diff f1.txt f2.txt
```
copies f1 to f2 but changes half to two thirds

shows the differences between both files

1,4c1,4=1,4 c 1,4=lines 1 to 4 are the lines that differ
c just means that there was a modification

copies f1 to f2 but changes all "likes" to "do not like"

12. What is the output of the last two commands in this sequence? Explain why.

```
$ cat > numbers1.txt
66
43
77
22
91
^D
$ cat > words1.txt
Rivendell
Gondolin
Lothlorien
Angband
Gondor
Moria
Shire
$ sort -n < numbers1.txt > numbers2.txt
$ sort -d < words.txt > words2.txt
```

copies numbers 1 to numbers 2 but sorts the numbers inside

copies words 1 to words 2 but puts words in alphabetical order inside

13. The following command allows you to calculate, with a good approximation, something about the processes currently being managed by the operating system. What?

```
$ ps -A | wc -l
```

ps show a list of all running processes, -A makes it show for all users
| is pipe symbol to feed left side as input on right side, wc -l counts the number of lines

14. What is the result of the last command in this sequence? Explain why.

```
$ emacs &
$ emacs &
$ emacs &
$ ps -A | grep emacs
 3577 ttys001    0:00.02 /usr/local/bin/emacs
 3578 ttys001    0:00.02 /usr/local/bin/emacs
 3579 ttys001    0:00.02 /usr/local/bin/emacs
 3583 ttys001    0:00.00 grep emacs
$ kill -9 3577 3578 3579
```