Riccardo Cappuzzo
S191436

System Programming

LAB 01

The majority of exercises is written as executable scripts, with the exception of the find exercises which are standard text files and are reported here as well.

Exercise 1

```
mkdir a
mkdir a/b a/d
mkdir a/b/e a/b/f a/b/c
mkdir a/b/f/i
mkdir a/b/c/g
mkdir a/b/c/g/h

chmod a+rwx a/ # part 1
chmod a-w a/d # part 2
chmod o-r a/b/c; chmod o+w a/b/c # part 3
chmod o+x a/b/c/g/; chmod o-x a/b/c/g/h/ # part 4
chmod g-r a/b/c/g/; chmod g+x a/b/c/g/ # part 5
chmod o-rwx a/b/; chmod u+rwx a/b/ # part 6
chmod u+r a/b/f/i/; chmod u-wx a/b/f/i/; # part 7

touch a/xyz; chmod go-rwx a/xyz # part 8
```

A script version is present in ex1.sh.
What happens if another user tries to read the file? What happens if another user tries to delete the file?
In both cases the result is “Permission denied”, and the user isn't able to access the file in any way. The only user who can access the file is root.

Exercise 2

```
umask 000
touch x
umask 066
touch y
```

The first mask makes every newly created file readable and writable from everyone, since 666-000 = 666. This is applied to test file x.
The second mask allows the owner to read and write, while everyone else has no permission (read or write) on the file, since 666-066 = 600. This is applied to test file y.

Exercise 3

```
find / -name "core" 2> errors.log
```
The command above searches from root (“/”) all the files containing “core” in their name, redirecting the error messages (including all the “permission denied” ones) to the output file errors.log.

```
find / -name "conf" -printf "%p %s\n"
```
The command above searches from root (“/”) all the files with name “conf” and then prints on stdout theit path followed by their size.

```
find /usr -perm /u=rx,g=rx,o=rx
find /usr -perm /a=rx
```
The commands above search folder “/usr” for all the files with read+write permissions for user, group and others. They are equivalent.

```
find /home/username -atime -7
```
The command above searches folder “/home/username” for all the files not accessed for at least 7 days. Username should be replaced with the correct user.

Exercise 4

This exercise can be executed by running the script ex4.sh.

The sorting commands are the following:
```
ls xxx* | sort -V
ls xxx* | sort -r -V
```
The commands above list all the files with a name starting with “xxx” and sort it. The “-V” parameter is needed in order to sort numbers within the text.

```
ls xxx* | sort -V > list.txt
```
This command performs the standard sort but redirects the output to file list.txt.

Exercise 5

```
find /home/username/ -path "*backup" | xargs mv -t folder_backups
```

The command above navigates the user's home directory and looks for all the directories ending with backup. Each directory is then moved to a directory (which must already be present) called folder_backups. In case the destination directory doesn't exist, an error occurs.

Exercise 6

```
find / -size +2M -or -name "a.*o"
```

The command above searches from the root folder for every file with size larger than 2MB or name starting with “a” and ending with “o”. In this version errors are shown on stderr. In order to avoid showing them it is possible to use the following command, which redirects the output to /dev/null:

```
find / -size +2M -or -name "a.*o” 2> /dev/null
```

Exercise 7

The script is contained in the file ex7.sh.

Exercise 8

The script is contained in the file ex8.sh.
The script employs the grep command in order to check whether the $PATH variable actually contains the /usr/local/bin and then prints a confirmation message.

Exercise 9

The script is contained in the file ex9.sh.

Exercise 10

The script is contained in the file ex10.sh.
I print the first line, then the lines containing empty spaces and eventually the last line. Since the shell collapses white spaces I first create a string which contains "." characters as placeholders and then I replace them with the "tr"command in order to have actual spaces.

Exercise 11

The script is contained in the file ex11.sh.
Error checking is performed by using "if" conditions on the two parameters. The first if checks whether parameter 1 is actually a directory, the second checks whether parameter 2 is a number and the third whether the number is positive. If all these checks are correct the search operation is performed. If a check isn't satisfied, the program exits.

Exercise 12

The script is contained in the file ex12.sh.
For every command I check whether the return value is 0 or not. If it is not 0 this means that an error when creating the folder occurred. Such errors may be due to a folder being non existent or to permissions denied (e.g. when trying to write inside a folder without the write permission).
I also redirect the stderr to /dev/null in order to have only the error message I specify.