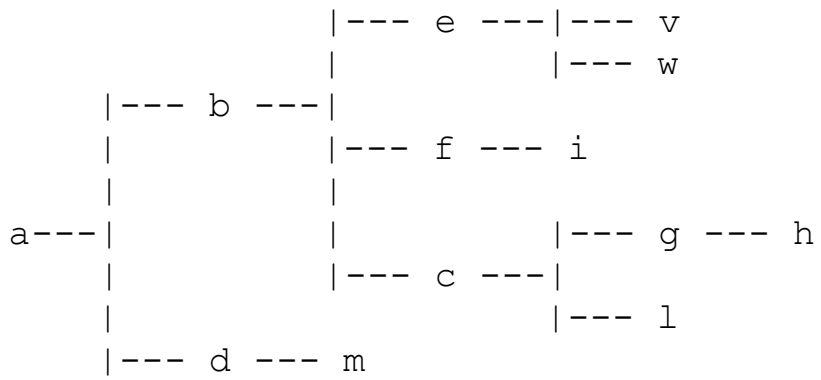
System Programming

Laboratory #1: **UNIX shell**

Laboratories are mandatory. Each student must submit the work following the rules published on the web portal. Any text produced must be exported into a (single) PDF file and the code (if required) a single file for each script or program. All Please remember that for each laboratory the final mark can get from -2.0 (not submitted or with severe flaws) up to +0.5 points (excellent work). If all 4 laboratories are not submitted 8 points are lost.

The deadline for the first laboratory is 08/11/2020 24:00.

Exercise 1 – dealing with files and folders
Build the following directory tree (and track all instructions used to do so):



modify permissions of directories in order that:

1. everybody can read and write in a;
2. nobody can create subdirectories of d;
3. others can not see inside c, but can create brothers of g;
4. others can enter in g, but not in h;
5. group can not see g (visualize what is inside g), but can enter in h;
6. others have no rights on b, while group has all permissions;
7. owner can read i, but has no other rights on i;

Create a file xyz in folder "a" (using touch command) and modify permissions in order to let only the owner read and write it. What happens if another user tries to read the file? What happens if another user tries to delete the file?

Exercise 2 – Basic bash

Inside the previous directory tree, try to solve the following operations (only one instruction per operation is allowed, no graphical user interface!):

1. Create an empty file named test.txt in a.
2. Duplicate the test.txt and name the duplicate test_dup.txt.
3. Edit the test.txt file with some poetry of William Blake.
4. Move the text.txt file into d.

5. Rename text.txt into w_blake_poetry.txt.
6. Duplicate w_blake_poetry.txt into c.
7. Remove folder d.
8. Print the full content of the a directory (including all subdirs, at any level) and save it into a file named a_dir.txt, placed in folder b.
9. Generate a new file in folder e with a copy of the first 15 lines of the test.txt file.
10. Generate a new file in folder f with a copy of the last 15 lines of the test.txt file.

Exercise 3 – Dealing with medias

Move the folder structure of Exercise 1 to an external USB key using the command line

Exercise 4 – Finding files

- Find all files in the system named "core"; try to avoid visualizing errors on the screen (put errors in a log file);
- Find all the files whose name contains the string "conf" and show its size;
- Find all files of the bin user, which are in the /usr directory (and all its subdirectories) and whose permissions are: –r-xr-xr-x;
- Find all files of yours that have been seen from less than a week.

Exercise 5 – Find

Find all the directories of your home whose name ends with the string "backup" (create some of them to check it) and move them in a backup directory.

Exercise 6 – Find

Find all files in the system that are greater than 2Mbyte and compress them into a single archive.tar.gz file. Then do it again with all file in a specific folder (create one in your home directory) and the move them into another folder (still in your home folder) called big_files.

Hint: Look for tar command and resort to the –exec option of the find command.

Exercise 7 – Scripts

Write a script able to read two parameters from the command line. The two parameters represent the names of two directories. Copy all the files and folders from the first directory to the second.

Exercise 8 – Scripts

Verify that the PATH variable contains the /usr/local/bin directory and print a message with the result of the verification.

Exercise 9 – Scripts

Read from **stdin** a set of strings, stopping when the string is "END". Copy and number each line on **stdout**. Then create a second script able to copy and number each line into a file, whose name is provided by command line.

Exercise 10 – Scripts

Create Y empty files called [xxx1, xxx2, xxx3, ... , xxxY], where xxx and Y are asked to the user by command line parameters. Show the list of the previously created file names both in ascending and descending order. Save the list in a file called "list.txt".

Exercise 11 – Scripts

Write a script that given two numbers as parameters from the command line, visualizes a rectangle whose dimensions are the two parameters.

Example:

```
$> ./Create_Rectangle.sh 5 10

→ +---------+
  |         |
  |         |
  |         |
  +---------+
```

Exercise 12 – Scripts and C

Write a bash script that resort to the program developed in LAB 0 to draw a tree and wrap it in a bash script that checks for all command line parameters before running the program and reports if the program terminated with some error.

Exercise 13 – Scripts

Write a script that given two parameters from the command line, the first as a directory name and the second as the size in bytes of a file, visualizes all ordinary files in the specified directory that can be read (read permission enabled) and with a size greater than the specified one. Please also verify that the given parameters are correct and display errors accordingly.

Exercise 14 – Scripts

Write a script that is able to create the following folder tree:

```
destination --- project_name ---|--- branches
                                |
                                |--- tags
                                |
                                |--- trunk
```

where the destination directory and the **project_name** are parameters of the script. Let the destination be either an absolute or a relative path. The script has to stop and notify any issue raised during the creation steps. Try it several times choosing destinations either writable or read-only.

Exercise 15 – Scripts

Write a script that is able to receive and to store N (less than 3) arguments from command line, then print out all of them at once. Number of received arguments (i.e., N) should be stored in the script in a way that it is accessible from terminal as well.

Note: N is not a defined number.

Exercise 16 – Scripts & C

Write a C program that is able to revert the content of a file and it can write it back into a different file. Then, write a script accept two folders as command line parameters, that reads the content of the first folder and for all non-text file, copy them into the second folder, while for all text files runs the C program in order to save a reverted copy of the file into the second folder.
Try to make the script recursive, repeating the operations for all files in subfolders (keeping the structure while copying).