Operating Systems 1: Introduction to C and File system.

In UNIX there are no different "logical drives" such as the "A:" and "C:" in Windows, instead, there is a single file system that starts in the *root* directory ('/'). Each hard drive, pendrive, etc. exists in a specific path (directory tree) originating from *root*. This peculiarity is not accidental, and is part of the internal structure and general philosophy of UNIX, which is a very file-oriented system.
UNIX uses files to interact with the rest of the world. Not only our source code and our programs are stored in files that we can edit, compile, etc., but also the programs we use to edit, compile, etc. are, in most cases, executable files. Even the Shell commands that we use when we interact with the Operating System are actually programs (stored in files). The Shell itself is actually a program external to the operating system, which only translates our commands to interact with the system core (executing programs). This way we could build our own commands and our own interpreter and use the system as we wish.

In this practical activity, we will get to know the programming language C on Linux and approach the basic functions of file management. In addition, through some Shell commands, we will discover the Linux Operating System and, in this particular practical activity, some basic concepts of the Linux file system.

Suggested Shell commands: `man`, `ls`, `which`, `cat`, `chmod`, `export`...
Suggested C functions: `printf`, `fopen`, `fclose`, `fputs`, `fprintf`...
Includes (header files): `<stdio.h>`

*** Note: Find out how these commands and C functions will help you in the development of this practical activity.**

1) Design and Develop a **program** in C language that, given a number of columns (m) and a number of rows (n), shows the two-dimensional matrix on the screen [nxm]. Each element of the matrix corresponds to the value of the operation [i * j], where i is the row number, and j is the column number of the given element.
Where is the `gcc` executable that you use to compile your code?
How does the system find that file without you specifying the location?

2) Expand your C program so that it obtains the following parameters through the command line (as input parameters): number of rows, number of columns, file name (optional).
If the user enters a file name, the resulting matrix should be saved in a file with the given name. Otherwise, the result of the matrix will be shown on the terminal (on the screen).
Example:

```
./myexecutable number_of_rows number_of_columns [file_name]
```

Determine the size of the results file when the matrix dimension is 50x20.

Change the permissions of the results file so that only your user has write permission. Has the file size changed?

3) Define a basic **Makefile** file to help you with the compilation tasks.
The file should facilitate changing the compilation options, and should delete all the objects and compiled files through the clean command.