PROJECT STATEMENT

DEVELOPMENT OF A TEXT ADVENTURE GAME TO LEARN THE USE OF A TERMINAL

1 *Introduction*

A text adventure is a type of adventure game where the entire interface can be text-only.

This project's aim is to develop a game in C language, using mostly system calls when using the Operating System's (OS) tools. On another hand, the objective of the development of said project is to put into practice the theoretical knowledge acquired in the subject and to develop students' ability to program using the programming interface offered by the Linux OS, mainly system calls, trying to use C library functions as little as possible.

The development of this project has a weight of 40% (30% + 10%) of the grade for the course. The project must be developed by teams of 4 to 5 students, and these teams must guarantee that all its members participate in all types of tasks to be carried out (design, development, testing and documentation).

2 Project's previewed plan

01/03/21	Project development	Project desing	Project design
08/03/21	Project development	Project progress demonstration	*Non-school day*
15/03/21	Project development	Friday schedulle Project design	*Non-school day*
22/03/21	Project development	Project progress demonstration	Project desing
29/03/21	*Friday schedulle* *Project development*	Project desing	*Spring holidays*
05/04/21	*Spring holidays*		
12/04/21	Wednesday, April 14		
19/04/21	Project development	Project progress demonstration	Project desing
26/04/21	Project development	Project desing	Project desing
03/05/21	Project development	Project writing	Project writing
10/05/21	Project delivery week		

3 Description of the project

Teams must develop a limited shell (**a computer program which exposes an operating system's services to a human user or other program) aimed to learn how to use Linux's basic commands.**

that will give the player the option to move through a series of directories. When entering these directories the player will receive some indications and thus, little by little, learn the use of some operating system commands.

As an example, teams can use the Terminus game, its final report and the structure of the its file tree. However, they have to meet the following requirements:

3.1 The project from the point of view of the player

Players have to learn how to:

1. **Change the current directory – (cd)**
2. **Read the contents of a directory (just the name or more information) – (ls -l)**
3. **Look the location – (pwd)**
4. **Read from file or stdin and print in stdout – (cat)**
5. **Create a new file – (touch)**
6. **Move a file – (mv)**
7. **Copy a file – (cp)**
8. **Search a file's contents – (grep)**
9. **Read the manual file of a command – (man)**
10. **Get information about available commands – (help)**

Players may have restricted access to some files and directories at certain times. Not all paths have to be always accessible.

3.2 The project from the point of view of the programmer

The minimum requirements that the project code must meet are the following (6pts):

- **Input/Output and file system management**
  - System calls
    - open
    - read
    - write
    - close
    - link
    - unlink
    - stat
    - readdir
    - lseek
  - C library functions
    - opendir
    - closedir

- **Process management**
  - System calls
    - fork
    - wait (not waitpid nor waitid)
  - C library functions
    - execlp
    - execvp

- You don't need to create all the commands in the users' list. Just the ones needed to use all the required system calls and library functions. For the rest, you can use the ones on Bash. However, the more you make the greater your score will be.
- Concatenate commands with pipes
- Error control
  - All errors that could occur during execution should be handled and users should be informed about why this error occurred. Test in bash a command that may cause an error and see how the user is informed, for example:
    - $ ls añsldk
      ls: cannot access añsldk: No such file or directory

Additional options (to raise the score up)

- Add the option of executing commands concurrently (&).
- Add the option of copying more than one file at the same time with threads.
  - Add showing the percentage already copied of each file.
- Add a history to the shell so that it is possible to see the last commands used.
- Add the option to protect a file with a password. The password shouldn't be shown on screen (*******).
- Report any other option that you have come up with.

4 *Final delirable*

Each team must deliver before **midnight of the 14th may** a .zip or .tar file containing:

- All the **code** of the project properly ordered in **directories**.
- A **.pdf** document reporting:
  - The description/design of the full project. Add the game's file tree and a short explanation of what happens at each place.
  - Each program's specification document (find example in eGela, project tab)
  - Each program's verification document (find example in eGela, project tab)
  - Specification and verification documents must show who has taken care of each task.
  - An explanation of how each part of the project relates to the concepts treated in theoretical classes.
  - An overall explanation of how the tasks have been distributed.
  - A section with the conclusions of each teammate (write them apart, without looking at the conclusions of the others and put them together only at the end).