Part I) Implementation of commands for file manipulation

Implement the following commands through the system calls (see http://manpages.ubuntu.com/manpages/). These commands must be implemented in C (for Linux) and shall be called through a command line interpreter (Bash shell). Any error message must be presented to the *stderr* descriptor. The following commands should be implemented in C and must not use any existing applications for file manipulation.

a) **show** filename – This command should present in the console the contents of the specified filename. If the file does not exist (in the current directory), the command should warn the user of that;
b) **count** filename – This command counts the number of lines within a file. The line number is defined as the number of '\n' characters in the contents. If the filename does not exist, the user should receive a warning;
c) **erase** filename – This command erases the filename (from the current directory). In case the filename does not exist, and only in that case, the user should be warned;
d) **inform** filename – This command informs the user, by showing the information in the console, about the given filename, like its size, file type, (normal, diretory, link, etc.), i-node, owner, creation and last modification date;
e) **append** filename1 filename2 – This command appends the contents of filename2 at the end of filename1. In case any of the filenames does not exist, the user must be warned;
f) **list** [path] – This command lists all the files and directories of the specified path. If the path is not given, it shows the contents of the current directory. Additionally, simple files and directories should be distinguished through a textual letter.

Part 2) Implementation of a Command Line Interpreter (4 values)

In order to replace the usual shell interpreter, Bash shell, with a new custom interpreter, an application whose function is to read a string from the console, and then execute this string as a command and its arguments in the system. The program should display the "%" symbol as an indication that it is ready to read a new user command.

The program must execute the command through generic process execution primitives regarding the functionality of the *system* (3) function, but **without making use of it**. Each command should give rise to a new process. Additionally, you may consider that the execution of the interpreter should be suspended until the specified command is complete. The interpreter should always indicate whether the command has successfully or unsuccessfully been terminated by its error / termination code. The program must allow several commands to be executed sequentially, that is, one after the other, until the user indicates the special command "ends" that ends this application.

```
$ ./interpreter
% list /home/user/Desktop
...
Finished command list with code: 0
% erase /home/user/Desktop/file
...
Finished command erase with code: 0
% ends
$
```