Objective:

Build a functional terminal based file explorer, albeit with a restricted feature set.

Prerequisites:

1. Basic usage and architectural know-how of file explorer features.
2. Preliminaries such as C/C++ code compilation, execution & debugging.
3. Familiarity with version-control system (git)
4. Usage of Makefile for compilation.

Specification:

Your file explorer is supposed to work in two modes. The application should start in normal mode, which is the default mode and used to explore the current directory and navigate around in the filesystem.

The root of your application should be the directory where the application was started.

The last line of the display screen is to be used as status bar - to be used in normal and command-line modes.

1. Normal Mode:

1. Read and display list of files and directories in the current folder-
    a. Your file explorer should show each file in the directory (one entry per line). The following attributes should be visible for each file
        i. File Name
        ii. File size (Human readable format similar to ls -lh)
        iii. Ownership (User & Group) & Permissions
        iv. Last modified
    b. Your file explorer is supposed to handle scrolling (vertical overflow) in case the directory has a lot of files.
    c. The file explorer should also show the entries "." & ".." for current and parent directory respectively.
    d. User should be able to navigate up & down the file list using corresponding arrow keys.

2. Open files & directories-
    a. When enter is pressed-
        i. *Directory*- Clear the screen and Navigate into the directory and show the files & directories inside it as specified in point 1.
        ii. *Files*- Your file explorer should be able to open files using the corresponding default application. You may use `xdg-open` for this
            - Bonus points if done without xdg-open. (Hint: Check default application list for linux)

3. Traversal-
    a. Go Back- On pressing left arrow key the file explorer should go back to the previously visited directory.
    b. Go Forward- On pressing right arrow key the file explorer should go forward to the next directory (Similar to forward/back feature of web pages)
    c. Up one level- On pressing backspace go up one level.
    d. Home - On pressing 'h' key, go to the home folder (the folder where application was started).

2. Command Mode:

The application should enter the command mode whenever the ":" (colon) key is pressed. Upon entering the command mode the user should be able to enter different commands. All commands should appear in a bottom status bar

1. Copy ':copy <source_file(s)> <destination_directory>'
   Move ':move <source_file(s)> <destination_directory>'
   Rename ':rename <old_filename> <new_filename>'
    a. Ex- `:copy foo.txt bar.txt baz.mp4 ~/foobar`
       Ex- `:move foo.txt bar.txt baz.mp4 ~/foobar`
       `Ex-:rename foo.txt bar.txt`
    b. Assume that the destination directory exists and you have write permissions.
    c. Copying / Moving directories should also be implemented.
    d. Ensure that file permissions and the ownership is intact.

2. Create file ':create_file <file_name> <destination_path>'
   Create directory ':create_dir <dir_name> <destination_path>'

    a. `Ex- :create_file foo.txt ~/foobar`
       `Ex- :create_file foo.txt .`
       `Ex- :create_dir folder_name ~/foobar`
       `Create the file/directory in current working directory if '.' is specified as the destination path.`

3. Delete file: `':delete_file <file_path>'`
   Delete directory: ':delete_dir <directory_path>'
   The file/directory path will be relative to the root from where the application is started.

4. Goto ':goto <location>'
   a. `Ex- :goto <directory_path>`
   b. Absolute path of directory w.r.t. application root will be given.
   c. In case of '/' your application should traverse to root directory of your application.

5. Search a file or folder given full filename.
   a. Command `':search <filename>'`
   b. Search for the given filename under the current directory recursively.
   c. Output would be the absolute paths which should be shown in the normal mode.
      On pressing back key you should go to the previous folder.

6. Snapshotting the filesystem and dump into a file
   a. Command '`:snapshot <folder> <dumpfile>`'
   b. Given a base directory this command should recursively crawl the directory and store the output in dumpfile.
   c. Output format should be similar to ls -R

7. On pressing 'ESC' key the application should go to Normal Mode.

Bonus-

- **Recycle bin-** Implement commands:
  - `':delete_file <filename>'` (permanent delete if current folder is trash folder)
  - `':restore <filename>'` (to be executed in trash folder)
  - `You are free to implement recycle bin in your own way.`

References:
[1] Canonical vs. Non-canonical Terminal modes: link.
[2] Beginning Linux Programming (Chapter 1 for how to compile your C/C++ source files, running the compiled and linked programs, using manual pages for reference, and debugging your programs in Linux. Chapters 5 for canonical mode. Chapter 9 for Makefile compilation.)
[3] Escape sequences (How to obtain fine control over the terminal): http://asciitable.com/ansi-escape-sequences.php
[4] Spawning processes and fork-exec: https://en.wikipedia.org/wiki/Fork_(system_call) https://en.wikipedia.org/wiki/Fork%E2%80%93exec