Title: Unix Shell

Objective: In this project we will build a unix shell from scratch which will provide a command line interpreter and support commands like Remove all but n, List Directory, Command History, Reverse Search a Command, and Execute, Sort by Type.

Description:

1. **Remove all but n <file1> [file2] [file3] …**
    - This call will be used to remove all the files from within a directory other than the number of files mentioned in the command line argument.
    - The user can enter any number of files as the command line arguments.
    - The function will trace the entire directory. Note that this API will delete all the files in the path mentioned with the file name.
    - E.g **rmallexn**(remove all except n) dir/file1 dir2/file2. This API call will delete all the files in dir1 except file1 and all the files in dir2 except file2. Only condition is that both dir1 and dir2 should be a subdirectory of the directory the API is called in.
    - The API will fetch all the files from the directory mentioned with the fileName.
    - The API will then delete all the files except the one passed in the arguments.
    - The API will continue until all the arguments are accessed.
    - Proper error conditions will be handled
    - Example command line API:
        - **rmallexn <file1> [file2] [file3] …**

2. **List Directory**
    - It is used to list all the files and directories in a given directory.
    - The list command will accept a variety of parameters.
    - The parameters involves:
        - -l : List the files and directories in the path mentioned as the argument.
        - -a: List all the files including the hidden files, symbolic links.
        - -tree: Show the hierarchy of all the files and directory inside the current directory.
        - -color: Display the files in different colors as per their file type
            - Green: executable files
            - Blue: Directory
            - Sky Blue: Symbolic links
            - Red: Hard links
    - The files will be fetched from the directory and based on the parameters passed, a switch case will be used to handle the parameters specifically.
    - Example call to the API
        - **listDir [-l] [-a] [-tree] [-color] <directory>**

3. **Command History**
    - It will be used to look back at the commands executed by the Unix shell entered by the user.

- This will be implemented using a doubly linked list **(Necessary for feature #4).**
- Every time a command is executed on the shell, the command will be added to the end of the doubly linked list.
- The doubly linked list will have a limit of 200 commands. Once 200 commands are filled up, the head of the doubly linked list will be removed.
- When the user requests for command history, the doubly linked list will be iterated from head to tail and each node in the doubly linked list will be displayed on the command line as a result of the history command with history command added at the tail of the doubly linked list.
- Example command:
    - **cmd_history**

4. Reverse search a command and execute it

- This command is used to search in the history list and execute the command based on the substring match.
- Once the user wishes to execute a command, the user will type **rev_search <command>**.
- Based on every substring entered, the program will reverse search the doubly linked list and stop at a point where the substring matches with any entry getting traversed from the tail.
- As the user keeps on adding the literals to the substring, it will stay at one particular node till the time the substring is still matching.
- Once the substring does not match the node, it will start moving towards the head in search of other nodes.
- The unique feature here is if the user enters a wrong substring and wishes to correct himself, in Unix, we have to start the reverse search again as there is no reversal. However, as we are using a doubly linked list, if the user deletes the substring, we will start the search again from the tail.
- With this, in one command, we will be able to execute the command in history without starting all over again.
- If no command matches, an appropriate error message will be displayed.
- Example command:
    - **rev_search <command>**

5. **Sort by type**

- Matches multiple files in a directory having similar pattern or type.
- Moves these matched files into a separate directory.
- This command is used to combine multiple files of similar pattern or type and it will add those files into a one separate sub directory.
- Example Command:
    - **sortbytype -t <file_type> <directory name>**
    - **sortbytype -n <file_name> <directory name>**

6. General Attributes Support (>, <, 2>, |, &)

- These general attributes are used either to redirect the result to a file or combining multiple commands at the same time using '|' operator or to run any process or command in the background.
- While accepting commands from the command line, there will be handlers each for **'>', '<', '2>', '|' and '&'**.
- If suppose the command goes **cmd_history > file.txt | sortbytype**
- Every argument will be treated separately.
- The argument would be considered as one command until we witness any one of the above attributes. Commands before and after those attributes will be treated as one command and once the command returns, the output will be displayed based on the order in which the attributes were placed in the command.

Releases:

V1: Command history, Dockerfile.
V2: Remove all but n, List Directory
V3: Reverse search and execute, Sort by type, General Attribute Support

Special Instructions:

- The Code will be maintained in the Git Repository,
- Github Repository Link:- https://github.com/aishwaryalonarkar/Rust-Unix-Shell
- A dockerfile to build the project inside the container will be maintained.
- Everyone will contribute to the code using their own github id's.
    - Aishwarya Lonarkar
    - Abhijeet Sonar
    - Omkar Wagle
    - Avranil Basu
- Guarantees safe execution of inputs (valid/invalid).
    - Use error handlers for best practice.
    - No panic!() errors are expected unless handled properly.
- We will disclose any use of libraries/crates in every release.