**Assignment-2 Question-2.2**

**Writeup (roll no-2019125)**

**Description:**

The program is designed to act as a terminal/shell with some 10 commands implemented along with 2 options for each command.

The commands along with the options implemented are as follows:

The options work as in real command line(for reference of how to use and other parameters needed).

Internal Commands (managed by the program itself):

1. cd (-L,~,..,.etc)
2. pwd(-L,-P)
3. echo(-E,-n)
4. history(-c,-d)
5. exit (no options possible)

External Commands (managed by forking external processes and system calls):

1. ls(-a,-f)
2. mkdir(-v,-Z)
3. rm(-I,-f)
4. date(-u, -R)
5. cat(-E,-n)

**Working of exec family commands:**

In this assignment I used the execvp() call of the exec family of system calls.this is used with the syntax as follows: **execvp (const char \*file, char \*const argv[]);** when it is called inside a child process it actually replaces the current process’s code file and executes the executable file passed in the first parameter shown above so in my code I have passed the absolute address of the needed executable files need for a particular external command to take place(NOTE:the absolute address is not hard coded and will work on any system as it automatically generates the address strings with reference to the location of storage of the program itself)

**Error handling and assumptions:**

**NOTE**: the **major error handled** is that the program can be run on any machine it will automatically handle the directories just u need to put the executable needed for the external command in the same location as that of the program.

The program automatically gets the user name and manages the directories accordingly taking its location in the system as the reference point.

The program automatically directs to home/<user’s> directory when cd -L is called with no further input.

If you try to navigate to a directory which does not exist it automatically shows the appropriate error.

History is maintained for 1000 entries and can hold first 100 character of each command you enter in a new line.

Error handled when you clear the history using the history-c command the counting is also reset, while when u delete a particular history content using the history -d <lineno> the counting is not reset.

NOTE: all commands also work even if the option is not specified and only the required parameter is passed.

rm -i command prompts you to confirm before actually deleting a file. While rm-f forcefully deletes the file.

cat command is designed to read any document in a string format irrespective of the length of document while cat -n and cat -E also can read and display files of any length.

NOTE: whenever user enters any invalid command appropriate error is displayed.

when ever cd is called after the directory reached after the command’s execution is displayed for user’s convenience.

If user type two or more commands who do not require any parameters then they are executed in the order in which they are entered that means there is no need to enter the new command in a new line. Thus, handles multiple commands in a line as it is managed by an actual terminal.

If we use cat for a totally empty file then not even a single line is printed for no reason.

When using a command which requires at least parameters like rm and mkdir the terminal will wait for you to enter the second parameter that means it will work when it will get some workable number of parameters.

When you delete a particular entry from the history using history -d <lineno> command that line is deleted and line displays “this was deleted” text in place of actual history at that very line for users convenience .

When you try to create a directory with a particular name and a file already exists with the exact same name then appropriate error is shown is user readable manner.

Similar is the case with when you try to use rm for a file that does not exist then also appropriate error is flashed on the screen in user readable manner.

**Test cases to test functioning:**

* navigate to any of the directory using cd command as you would do in actual shell.Eg.” cd ..” or “cd -L”
* You can use ls wherever you want to list the contents of the current directory Eg.” Ls”
* You can use history command at any point of time to display the history of commands since the program has been running. Eg. “history” or “history -c” or “history -d 0”
* You can use echo wherever you want the input afterwards the echo will be printed as it is in the next line. Eg. “echo hello echo is working fine” or “echo -E this is also working fine” or “echo -n no trailing next line is printed”
* Exit command will simply end the terminal program and exit you displaying appropriate text .Eg “exit”
* You may navigate to any directory using and use pwd to get the present working directory. Eg. “pwd”
* You can navigate to a location holding a readable file and then use the cat command for the file with let say file name as <filename.txt>.Eg “cat filename.txt” or “cat -E filename.txt” or “cat -n filename.txt”
* Wherever you are in the program you can anytime type “date” to get the date printed in the next line along with the current time.Eg “date” or “date -u” or “date -R”
* Mkdir can be used to create a directory in your present working directory by using it as. Eg. “mkdir newfile” or “mkdir -v newfile” or “mkdir -Z newfile”
* Rm can be used to delete a particular file as follows.Eg.”rm deletethis.o” or “rm -i deletethis.txt” or “rm -f deletethis.txt”