Assignment 1.2: Basic Linux/Unix shell

Working of the program -

1) Internal Commands -

Internal commands are those which are interpreted by the shell program itself, without requiring a different program.

1) External Commands -

These external commands have individual programs. To handle these external commands, the shell should typically create a new process, using the fork() system call and within each process and the execl() family system call to run the individual program.

NOTE: TEST CASES ARE WRITTEN INDIVIDUALLY FOR EVERY COMMAND IN THIS WRITE UP.

Internal Commands -

1) cd - System call used -chdir()

- **Options -**

  -L - force symbolic links to be followed: resolve symbolic links in DIR after processing instances of `..'

  --help - Display the help and exits

- **Errors -**

  Error 1 - If the directory is not present error will be printed that "No such file or directory"

Error 2 - If option is not present then error will be printed "No such option or directory or file"

- **Assumptions -**
  - Our home directory is the directory in which the program file is present.

- **Test Cases -**
  Please type these test cases in the same order or you may not find the directory or file.
  1) cd --help
  2) cd ..
  3) cd -L Question2
  4) cd ..
  5) cd Question2
  6) cd ..
  7) cd
  8) cd file **- No directory found ERROR**

cd .. : goes to the previous directory
cd : goes to the home directory
cd [directory] : goes to the given directory

2) echo -

- **Options -**

  -n - do not output the trailing newline
  -E - disable interpretation of backslash escapes (default)

- **Errors -**
  Error - If no argument is passed to echo error will be printed that "Argument required"

- **Assumptions -**
  - Don’t write the string to be printed in double quotes “” as double quotes will also be printed.

- **Test Cases -**
  1) echo hello world
  2) echo -n hello world
  3) echo -E hello\t \n world \\
  4) echo Hii how are you? What you doing?
  5) echo **- Argument Required ERROR**

3) history -

- **Options -**

  -c clear the history list by deleting all of the entries
  -d delete the history entry at position OFFSET.

- **Errors -**

  Error 1 **-** If Option is not present, then error will be printed that “Option not present”
  Error 2 **-** If in **-d option** the offset is not present, then error will be Printed that “Offset Entered is Wrong”.

- **Assumptions -**
  - The history list can store 1000 commands.

- **Test Cases -**

  Please type the commands in the following order

  1) echo hello world
     echo -n hello hello
     echo -E hii\n \\
     cd ..
     cd
     history
  2) history -d 3
  3) history
  4) history -c
  5) history
  6) history -p **- OPTION NOT PRESENT ERROR**

4) pwd - System call used - getcwd()

- **Options -**

  -L print the value of $PWD if it names the current working directory
  -P print the physical directory, without any symbolic links
  --help Display the help and exits

- **Errors -**

  Error 1 - If Option is not present, then error will be printed that "Option not found"

- **Test Cases -**

  1) pwd
  2) cd ..
     pwd -L

3) cd Question2
   pwd -P
4) pwd --help

5) exit - System call used - exit()

- **Options -**

  exit [n] - Exits the shell with a status of N.
  --help - Display the help and exits

- **Test Cases -**

  1) **exit**
  2) **exit 3**

External Commands -

1) ls - System call used - scandir()

- **Options -**

  -a - do not ignore entries starting with **‘.’**
  -m - fill width with a comma separated list of entries
  -1 - list one file per line.

- **Errors -**
  Error 1 - arguments are not enough.
  Error 2 - There is no such file or directory as the path given as argument

- **Assumptions -**
  -no other option is used

- **Test Cases -**

  FOLLOW THIS ORDER FOR BETTER RESULTS

  1) ls
  2) ls -a
  3) ls -m
  4) cd ..
  5) ls -1
  6) ls Question2

2) cat - **function used fgets()**

- **Options -**

  -n **-** number all output lines
  -E **-** display $ at end of each line

- **Errors -**

  All the errors types present in function fopen() has been managed using library <errno.h>
  Error 1 - file don't exists.
  Error 2 - User doesn't have access to the file.

- **Assumptions -**
  -Only two given options are there, no other option should be used.

- **Test Cases -**

    1) cat Sample_cat1.txt Sample_cat2.txt
    2) cat -n Sample_cat1.txt Sample_cat2.txt
    3) cat -E Sample_cat2.txt
    4) cat A.txt **-No such file exists ERROR**

3) date -

- **Options -**

    -u **-** print or set Coordinated Universal Time (UTC)
    --help **-** Display the help and exits

- **Errors -**

    Error 1 **-** If Option is not present, then error will be printed that "No such Option"

- **Assumptions -**

    -I am using Default format for Date and Time

- **Test Cases -**

    1) date
    2) date -u
    3) date --help
    4) date -p **-NO SUCH OPTION ERROR**

4) rm - System call used - remove()

- **Options -**

  -i **-** prompt before every removal (asks whether you want to remove the file)
  --help **-** Display the help and exits

- **Errors -**

  All the errors types present in function remove() has been managed using library <errno.h>
  Error 1 - file don't exists.
  Error 2 - Given path is a directory.
  Error 3 - User doesn't have access to the file.

- **Assumptions -**

  -Only two given options are there, no other option should be used.

- **Test Cases -**

  First create files-

  1) mkdir file1 file2 file3
     mkdir file1/file4
  2) rm file2 file3
  3) rm -i file1/file4
  4) rm --help
  5) rm file5 **-No such file Error**

5) mkdir - **System call used - mkdir()**

- **Options -**

  -v **-** prints a message for each created directory
  --help **-** Display the help and exits

- **Errors -**

  All the errors types present in function mkdir() has been managed using library <errno.h>
  Error 1 - The named file exists.
  Error 2 - The parent directory resides on a read-only file system.
  Error 3 - A component of the path prefix is not a directory.

- **Assumptions -**
  -Only two given options are there, no other option should be used.

- **Test Cases -**

  mkdir file1 file2 file3
  mkdir -v file1/file4
  mkdir --help
  mkdir file1 **-File Already Exists ERROR**