Operating System - Assignment 1

Exercise 1.2

(Ayush Misra,2019301)

Basic Linux/Unix Shell

Steps to keep in mind:
1)Take string input for the command.
2)Parse the string into individual words for easier computation .
3)Check for command and then its option
4)Execute external commands using fork() and execl().
5)Printing the necessary output.

INTERNAL COMMANDS:

1. **cd (options: '..' , '~')**
   It changes the working directory.

   Implementation in C using: **chdir()**
   int chdir(const char *path);
   chdir() changes the current working directory of the calling process
   to the directory specified in path.

   $ cd ..
   Changes the directory to the previous directory.

   $ cd ~
   Changes the directory to the home directory.

2. **pwd (options: '-L' , '-P')**
   It prints the path of the working directory.

   Implementation in C using : **getcwd()**

char *getcwd(char **buf*, size_t *size*);
The getcwd() function copies an absolute pathname of the current working directory to the array pointed to by *buf*, which is of length *size*.

$ pwd -L
Prints the symbolic path.

$pwd -P
Prints the actual path.

3. history(options: ‘-c’ , [number])

Used to view the previously executed command

Implementation in C using: simple 2d array to store string commands.

$ history 5
To show the limited number of commands that executed previously

$ history -c
The whole history can be removed using **history -c** option.
(Using memset on the array in C)

4. echo(options: ‘-n’ , ‘-e’)

Used to display line of text/string that are passed as an argument

Implementation in C using:Reading the command string character by character.

$ echo -n
This option is used to omit echoing trailing newline .

$echo -e
Enables the interpretation of backslash escapes

5. exit(options: ‘-n’ , ‘-e’)

Terminates the shell

Implementation in C using: **exit(0)**

<u>EXTERNAL COMMANDS</u>

For external commands my shell creates a new process using the fork() system call and executes the external .c file corresponding to that command with the help of execvp() system call to run the command externally.

int execvp (const char *file, char *const argv[]);
file: points to the file name associated with the file being executed.
argv: is a null terminated array of character pointers.

1. **ls(option: '-a' , '-1')**
   ls *with no option list files and directories in bare format*

   Implementation in C using: **readdir()**
    struct dirent *readdir(DIR *dirp);
   The readdir() function returns a pointer to a dirent structure
    representing the next directory entry in the directory stream pointed
    to by dirp. It returns NULL on reaching the end of the directory
    stream or if an error occurred.

   $ ls -a
   Prints all files including the ones starting with dot

   $ls -1
   Prints all files and directories in a long list format.

2. **mkdir(option: '-v' , '-p')**
   Creates directory in the working directory

Implementation in C using:**mkdir()**
int mkdir(const char **path*, mode_t *mode*);
The *mkdir*() function shall create a new directory with name *path*. The file permission bits of the new directory shall be initialized from *mode***.**

$ mkdir -v
It displays a message for every directory created

$mkdir -p
A flag which enables the command to create parent directories as necessary. If the directories exist, no error is specified.

3. **rm(option: '-i' , '-d')**
   Removes files/directories from the working directory

   Implementation in C using: **remove()**

   $ rm -i
   Confirms from user for the deletion to begin

   $ rm -d
   Used for the deletion of empty directories
   (Implementation in C by **rmdir())**

4. **cat(option: '-n' , '-s')**
   It reads data from the file and gives their content as output

   Implementation in C using: **fopen()- to open the file**
   fgets()-to read the file line by line

   $ cat -s
   suppress repeated empty lines in output

   $ cat -n
   To view contents of a file preceding with line numbers**.**

5. **date(option: ‘-u’ , ‘%+d’)**
   Displays the current date and time of the system

   Implementation in C using: **C library time.h**

   $ rm -u
   Displays GMT time

   $ rm +%d
   Displays the number of day of the month

<u>CORNER CASES HANDLED(Error Handling)</u>

1) Multiple inputs handled for mkdir,cat and rm
2) Invalid command and option error handled
3) File/Directory not found error handled
4) Fork failing.