Linux Shell

Name: Kunal Sharma
Roll No: 2021331
Section: B
Branch: CSD

How my shell work

- The shell operates in a while(1) endless loop. The user input is first taken as a string using fgets, parsed using strtok, and then stored in a 2d array referred to as command. The shell prompt is then printed.
- The programme than examine the command; if it is null, it simply print another prompt;if it is a legitimate internal command, it execute it internally; if it is a valid external command, it uses fork(),execvp()and wait(); Otherwise,it informs the user that the command is invalid.

Assumptions

- Command length will not exceed 500 characters
- It does not support two options at once for any commands

Shell Handles the following commands

- Internal commands - cd, echo, pwd and exit
- External commands - ls, cat, date, rm and mkdir

Options for Shell Commands

- cd
  - \- goes one directory back
  - ~ goes to home directory
- echo
  - -n do not append a newline
  - * works like ls command
- pwd
  - -L Prints the symbolic path
  - -P Prints the actual path

- exit

- ls
  - -a do not ignore the entry starting with .
  - -l Shows list in long list format.
- cat
  - -e get $ at the end of each line
  - -n to display line numbers
- date
  - -u get coordinated universal time(UTC)

    - -R output date and time in RFC 5322 format
- rm
    - -i prompt before every removal
    - -ri removes a directory
- mkdir
    - -v prints a message for each created directory
    - -p add directory including its sub directory

Shell Implementation by threads

- Pthread_create():
    - It is a function used to create a new thread with attributes within a process.
    - Its functioning is the same as fork() system call.

- Pthread_ join():
    - This function waits for a thread to terminate, detaches the thread then returns the threads exit status.

- Its functioning is the same as the wait() system call.

- system():

  - It executes the system linux based commands by taking in standard input as argument.
  - Its functioning is the same as the execvp() system call.

➢ To access the commands using pthreads type &t before the command you want to use.

Error Handling

- cd

  - If more than 1 argument is passed it prints the error “Too many arguments”.
  - If the given directory does not exist. It prints the error “No such directory”.
  - If an invalid option is given it prints the error “Invalid command”.

- echo
    - It does not require error handling. Any input is handled natively.
- pwd
    - If more than 1 argument is passed it prints the error “Too many arguments”
    - If the argument passed is wrong it prints the available commands.
- exit
    - It uses exit() to exit the current session
- ls
    - If more than one argument is supplied, the message “Too many arguments” is printed.
    - If the argument passed is wrong it prints the available commands.
- cat
    - If more than 1 argument is passed it prints the error “Too many arguments”

    - If the name of file is not present in the directory it prints the error “No such file or directory”

- date
    - If the argument passed is wrong it prints the available commands.
    - If the argument passed is wrong it prints the available commands.

- rm
    - If the argument passed is wrong it prints the available commands.
    - If the argument passed is wrong it prints the available commands.

- mkdir
    - If the argument passed is wrong it prints the available commands.
    - If the argument passed is wrong it prints the available commands.

➢ If any command is used that is not one of those listed above the error “Invalid command” is printed.

Test cases

Test case for shell implementation using system calls:

$ cd-> Home directory
$ cd desktop -> To go to desktop
$ cd desktop -> error “No such Directory”
$ pwd -> To check the present directory
$ ls -l -> Shows everything on the desktop
$ ls -d -> error “Shows available commands”
$ echo * -> works same as ls
$ date -u anything -> error “Too many arguments”
$ date -u -> shows date in UTC
$ date -R -> shows date in RFC 5322 format
$ mkdir -v filename filename1 -> error “Too many arguments”
$ mkdir -v filename -> makes a directory
$ lst -> error “Please enter a valid command”
$ mkdir -p filename/filename1 -> make a directory in filename

$ rm -i file.txt -> ask y/n before removing the file.txt
$ cd filename -> goes into filename directory
$ cd filename -> error “No such directory”
$ pwd -> To check the present directory
$ rm -ri filename1 -> Remove the directory named filename
$ cd - -> To go to desktop
$ pwd -> To check directory
$ cat -n file.txt -> Prints the file contents with line number
$ cat -e file.txt -> Prints the file contents with $ at the end of line
$ rm -i file.txt -> ask y/n before removing the file.txt
$ rm -i file.txt -> error “No such file or directory”
$ rm -ri filename -> removes the directory named filename
$ echo -n “Bye Bye!!”
$ exit -> To exit the current session

Test case for shell implementation using pthreads:

$ cd-> Home directory
$ cd desktop -> To go to desktop
$ cd desktop -> error “No such Directory”
$ pwd -> To check the present directory
$ &t ls -l -> Shows everything on the desktop
$ &t ls -d -> error “Shows available commands”
$ echo * -> works same as ls
$ &t date -u anything -> error “Too many arguments”
$ &t date -u -> shows date in UTC
$ &t date -R -> shows date in RFC 5322 format
$ &t mkdir -v filename filename1 -> error “Too many arguments”
$ &t mkdir -v filename -> makes a directory
$ &t lst -> error “Invalid command”
$ &t mkdir -p filename/filename1 -> make a directory in filename

$ &t rm -i file.txt -> ask y/n before removing the file.txt
$ cd filename -> goes into filename directory
$ cd filename -> error “No such directory”
$ pwd -> To check the present directory
$ &t rm -ri filename1 -> Remove the directory named filename
$ cd - -> To go to desktop
$ &t cat -n file.txt -> Prints the file contents with line number
$ cat -e file.txt -> Prints the file contents with $ at the end of line
$ &t rm -i file.txt -> ask y/n before removing the file.txt
$ &t rm -i file.txt -> error “No such file or directory”
$ &t rm -ri filename -> removes the directory named filename
$ echo -n “Bye Bye!!”
$ exit -> To exit the current session