DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROJECT REPORT

Course Code: **10B17CI571**

Course Name: **Operating Systems Lab**

Power X Shell

A Command Line Interface Project in C Language

SUBMITTED BY:

ISHITA GUPTA (211472), NANDINI SAINI (211355)

Date: **12-05-2023**

SUBMITTED TO:

DR. MONIKA BHARTI JINDAL | ASSISTANT PROFESSOR (SG)

1 TABLE OF CONTENTS

2 ACKNOWLEDGEMENT

- We would like to express my gratitude to Jaypee University of Information Technology, Solan for providing me with the opportunity to pursue this project as a part of my academic curriculum. I am also grateful to the resources and facilities provided by the institution that have been instrumental in completing my project.

- I am also grateful to Dr. Monika Bharti Jindal, our professor for providing guidance and support through the course of the semester. Their feedback and suggestions were valuable in refining my approach and sharpening my problem-solving skills.

- I hope that this acknowledgement accurately reflects my gratitude towards the individuals and resources that have contributed to the successful completion of my project.

- Sincerely,

  Ishita Gupta (211472), Nandini Saini (211355)

  Batch: CS-48

3 INTRODUCTION

- The Power X Shell is a command-line interface program written in C language that emulates the functionality of a shell in Linux. This project aims to create a simple, easy-to-use shell that can execute basic commands such as changing the directory, listing files, and running other programs.
- The project is designed to read input commands from the user, tokenize them, and execute the appropriate actions based on the commands given. The program implements several basic shell functionalities like change directory (cd), list directory contents (dir), execute external commands (like ping, ipconfig), and keep command history. The program can also copy files from one location to another.

4 PROBLEM STATEMENT

- The Power X Shell aims to provide a simple and intuitive interface for users to execute basic commands in Linux. The project seeks to address the issue of users having to use the terminal to access the command-line interface.

5 AIMS AND OBJECTIVE

- The main aim of the project is to create a user-friendly shell program that emulates the functionality of the Linux command-line interface. The specific objectives of the project are:
- To provide a simple and intuitive interface for users to execute basic commands.
- To allow users to change the current working directory using the 'cd' command.
- To enable users to list the contents of the current directory using the 'dir' command.
- To provide users with the ability to exit the shell program.

6 TECHNOLOGY STACK

- The Power X Shell project was developed using the C programming language.
- The following libraries and functions were used:

- stdio.h - for input/output operations
- string.h - for string manipulation functions
- stdlib.h - for dynamic memory allocation functions
- sys/wait.h - for waitpid() function
- unistd.h - for getcwd() function

7 SIGNIFICANCE OF THE PROJECT

- Learning and Understanding: Developing a shell emulator in C provides an opportunity to deepen your understanding of the Linux operating system and how shells work. It allows you to explore the intricacies of process management, input/output redirection, command execution, and other fundamental concepts.
- Customization and Extension: Developing your own shell emulator allows you to customize and extend its functionality to suit your specific needs. You can add new features, modify existing ones, or integrate external libraries to enhance the shell's capabilities.

8 PROJECT FEATURES

- Customized command line interface (CLI) for Linux OS written in C language.
- Support for basic commands like ls, cd, pwd, and exit.
- User-friendly interface with clear and concise command prompt messages.
- Ability to handle input from the user and tokenize it into separate arguments.
- Robust error handling with proper error messages displayed to the user.
- Continuous running of the shell until the user types the exit command.
- Ability to display the current working directory using the pwd command.
- It also displays Ip confirmation and ping information.
- Customized command dir added to display the current working directory.
- Implementation of cd command to change the current working directory and display it to the user.

- Code structure with proper comments for easy understanding and maintenance.
- Use of dynamic memory allocation for handling input and arguments.
- Minimal use of external libraries or dependencies to ensure faster execution and lower resource consumption.
- Overall, the project is aimed at providing a powerful and efficient shell interface for Linux users.

9 CODE SNIPPETS - OUTPUT

```
PS E:\Projects\os> dir


    Directory: E:\Projects\os


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        06-05-2023  01.36 AM                .vscode
-a----        12-05-2023  02.06 AM              0 file.txt
-a----        12-05-2023  01.21 AM              5 file1.txt
-a----        12-05-2023  11.32 AM             45 file2.txt
-a----        12-05-2023  11.30 AM           6882 main.c
-a----        12-05-2023  02.32 AM          52998 main.exe
-a----        12-05-2023  11.06 AM         502506 Project-Report-LRC.pdf
-a----        12-05-2023  10.40 AM          58419 Project-Report-Power-X-Shell.docx


PS E:\Projects\os> cd E:\Projects
PS E:\Projects> dir


    Directory: E:\Projects
```

```
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        13-03-2023  11.52 PM                1
d-----        11-05-2023  10.26 AM                daa project
d-----        12-05-2023  11.05 AM                dsl project
d-----        27-11-2022  05.12 PM                Guess the number
d-----        06-12-2022  04.00 PM                harry potter cloak
d-----        10-05-2023  11.37 AM                Mental healh Website
d-----        12-05-2023  11.06 AM                os
d-----        07-12-2022  09.48 AM                paint-main
d-----        23-04-2023  11.30 PM                Quiz Game
d-----        25-04-2023  01.30 AM                quiz website
d-----        11-03-2023  09.25 PM                Wine shots
d-----        03-12-2022  11.43 PM                Youtube video downloader
-a----        11-05-2023  10.46 AM        3035378 daa project (zip file).zip
-a----        10-05-2023  10.03 PM            131 webdev1.code-workspace


PS E:\Projects> ping

Usage: ping [-t] [-a] [-n count] [-l size] [-f] [-i TTL] [-v TOS]
            [-r count] [-s count] [[-j host-list] | [-k host-list]]
            [-w timeout] [-R] [-S srcaddr] [-c compartment] [-p]
            [-4] [-6] target_name
```

```
Options:
    -t             Ping the specified host until stopped.
                   To see statistics and continue - type Control-Break;
                   To stop - type Control-C.
    -a             Resolve addresses to hostnames.
    -n count       Number of echo requests to send.
    -l size        Send buffer size.
    -f             Set Don't Fragment flag in packet (IPv4-only).
    -i TTL         Time To Live.
    -v TOS         Type Of Service (IPv4-only. This setting has been deprecated
                   and has no effect on the type of service field in the IP
                   Header).
    -r count       Record route for count hops (IPv4-only).
    -s count       Timestamp for count hops (IPv4-only).
    -j host-list   Loose source route along host-list (IPv4-only).
    -k host-list   Strict source route along host-list (IPv4-only).
    -w timeout     Timeout in milliseconds to wait for each reply.
    -R             Use routing header to test reverse route also (IPv6-only).
                   Per RFC 5095 the use of this routing header has been
                   deprecated. Some systems may drop echo requests if
                   this header is used.
    -S srcaddr     Source address to use.
    -c compartment Routing compartment identifier.
    -p             Ping a Hyper-V Network Virtualization provider address.
    -4             Force using IPv4.
    -6             Force using IPv6.
```

```
PS E:\Projects> ipconfig

Windows IP Configuration


Ethernet adapter Ethernet:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Ethernet adapter Ethernet 2:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::c943:5413:a8a5:19a5%5
   IPv4 Address. . . . . . . . . . . : 192.168.56.1
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

Wireless LAN adapter Local Area Connection* 1:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 2:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Wi-Fi:
```

```
   Connection-specific DNS Suffix  . : juitw.org
   Link-local IPv6 Address . . . . . : fe80::b43a:ac8d:4f50:82ae%7
   IPv4 Address. . . . . . . . . . . : 172.16.110.153
   Subnet Mask . . . . . . . . . . . : 255.255.224.0
   Default Gateway . . . . . . . . . : 172.16.96.1

Ethernet adapter Bluetooth Network Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
PS E:\Projects> md newProject


    Directory: E:\Projects
```

```
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        12-05-2023  11.33 AM                newProject

   1 try { . "c:\Users\Admin\AppData\Local\Programs\Microsoft VS Code\resources\app\out\vs\workbench\contrib\terminal\bro...
   2 cd "e:\Projects\os\" ; if ($?) { gcc main.c -o main } ; if ($?) { .\main }
   3 cd "e:\Projects\os\" ; if ($?) { gcc main.c -o main } ; if ($?) { .\main }
   4 cd "e:\Projects\os\" ; if ($?) { gcc main.c -o main } ; if ($?) { .\main }
   5 clear
   6 cd "e:\Projects\os\" ; if ($?) { gcc main.c -o main } ; if ($?) { .\main }
   7 dir
   8 cls
   9 cd "e:\Projects\os\" ; if ($?) { gcc main.c -o main } ; if ($?) { .\main }
  10 dir
  11 cd E:\Projects
  12 dir
  13 ping
  14 ipconfig
  15 md newProject
  16 copy file1.txt file2.txt E:\Projects\os
```

10 CODE FLOW AND EXPLANATION

The Power X Shell program is structured as follows:

1. Initial Setup and Main Program Loop:

The program then enters a while loop that keeps on asking the user for input until the user types "exit" to exit the program. Within this loop, the mentioned tasks are performed.

2. Reading Input from The User:

The program reads input from the user using the fgets() function and stores it in a character array called line. The program then creates a deep copy of the input string using strdup() and stores it in a character pointer called cinput.

3. Tokenizing the Input String:

The cinput string is then tokenized into words using the strtok() function. The individual words are then stored in an array of character pointers called savedTokens.

4. Checking for the "exit" Command:

The program checks if the user has entered the "exit" command by comparing the first token in the savedTokens array with the exitSmallString and exitBigString strings. If the user has entered the "exit" command, the program exits.

5. Checking for the "dir" Command:

The program checks if the user has entered the "dir" command by comparing the first token in the savedTokens array with the dirString string. If the user has entered the "dir"

command, the program calls the getcwd() function to get the current working directory and displays it to the user.

6. Checking for the "cd" Command:

The program checks if the user has entered the "cd" command by comparing the first token in the savedTokens array with the cdString string. If the user has entered the "cd" command, the program calls the chdir() function to change the current working directory to the directory specified in the second token of the savedTokens array.

7. Checking for the "md" Command:

The program then defines a function `mdFunction()` which is used to create a new directory. However, this function is not used in the current implementation.

8. The "clear" Command:

The main function of the program starts by calling the `clear()` function, which clears the console window before printing the initial message. It then prints a welcome message and starts an infinite loop to read commands from the user.

9. Creating the Command Prompt for Power-X-Shell:

The program then prompts the user to enter a command by printing the command prompt `Power X Shell>`. The user input is read using the `fgets()` function and is stored in the `line` character array.

10. History Feature to Display All The Previously Entered Commands By The User:

The command entered by the user is then stored in the `commandHistory` array, which is used to keep track of the commands entered by the user. The program also counts the number of commands entered using the `commandCount` variable.

The program then checks if the first token of the input command is "history" using `strcmp()` function. If it is, the program displays the command history to the user by iterating through the `commandHistory` array and printing the commands one by one. The program then continues with the next iteration of the loop.

11. Checking for the "ping" Command:

If the first token is "ping", the program checks if the user has entered enough arguments. If the user has entered less than two arguments, an error message is displayed. If the user has entered two or more arguments, the program constructs a `ping` command using the `sprintf()` function, which is then executed using the `system()` function.

13. Checking for the "pipe" Command:

This implementation checks if the user has entered a pipe command (command1 | command2) and executes both commands using a pipe. If the user enters a single command, it is executed normally using the `system`() function. The `execute_pipe`()

14. Checking for the " ipconfig " Command:

If the first token is "ipconfig", the program executes the `ipconfig` command using the `system()` function.

15. Checking for the " Copy " Command:

If the first token is "copy", the program checks if the user has entered enough arguments. If the user has entered less than three arguments, an error message is displayed. If the user has entered three or more arguments, the program gets the source file names and the destination directory and copies each file to the destination directory.

16. Printing the Tokens:

Finally, the program prints out the tokens in the savedTokens array for testing purposes.

11 GITHUB LINK OF THE PROJECT

- https://github.com/nandini-2407/Power-X-Shell
- https://github.com/Ishita1604/Power-X-Shell

12 REFERENCES

- Writing Your Own Shell LG #111 (linuxgazette.net)
- https://onestepcode.com/writing-basic-shell/