Project Report

Project Title	Shell Implementation
Course code	CS 218
Course title	Operating Systems
Department	Computer Science
Section	4C
Group Members	▪ Sara Jamal (19K-0207) ▪ Naz Panjwani (19K-1256) ▪ Musbah Sohail (19K-1510)

❖ What is a Shell?

A shell basically acts as an interface between the user and the operating system. It is called a shell because it is the outer layer of the operating system. It executes programs based on the input provided by the user. Generally, operating system shells use either a command-line interface (CLI) or graphical user interface (GUI). We used the command line interface(CLI).

❖ Shell working mechanism:

A shell accepts instructions or commands fed by user in user understandable language and translate it to binary language which a computer can easily understand. So in short a Shell is a language translator between the user and the operating system.

❖ Types of Shell:

- Bourne shell (sh)
- Korn shell (ksh)
- Bourne Again shell (bash)
- POSIX shell (sh)

We worked on Bourne Again shell (bash)

❖ Objective

The main objective of our project was to create/implement our own customized shell and modify it according to our own convenience. Furthermore, we wanted to automate frequently performed operations by executing them through simple, short commands, in addition to the already built-in commands, which would otherwise require long calculations or a sequence of commands to be run.

❖ Project description

To achieve our objective, we built our own shell which first takes input commands from the user, taking care of any pipes or extra white spaces given in the command, parses them and executes them accordingly. As for the commands we have modified the already built in commands such as help and along that we have made our own custom commands like showip, showdir etc. Following is the complete list of all the commands we have made with their brief description.

❖ Commands list

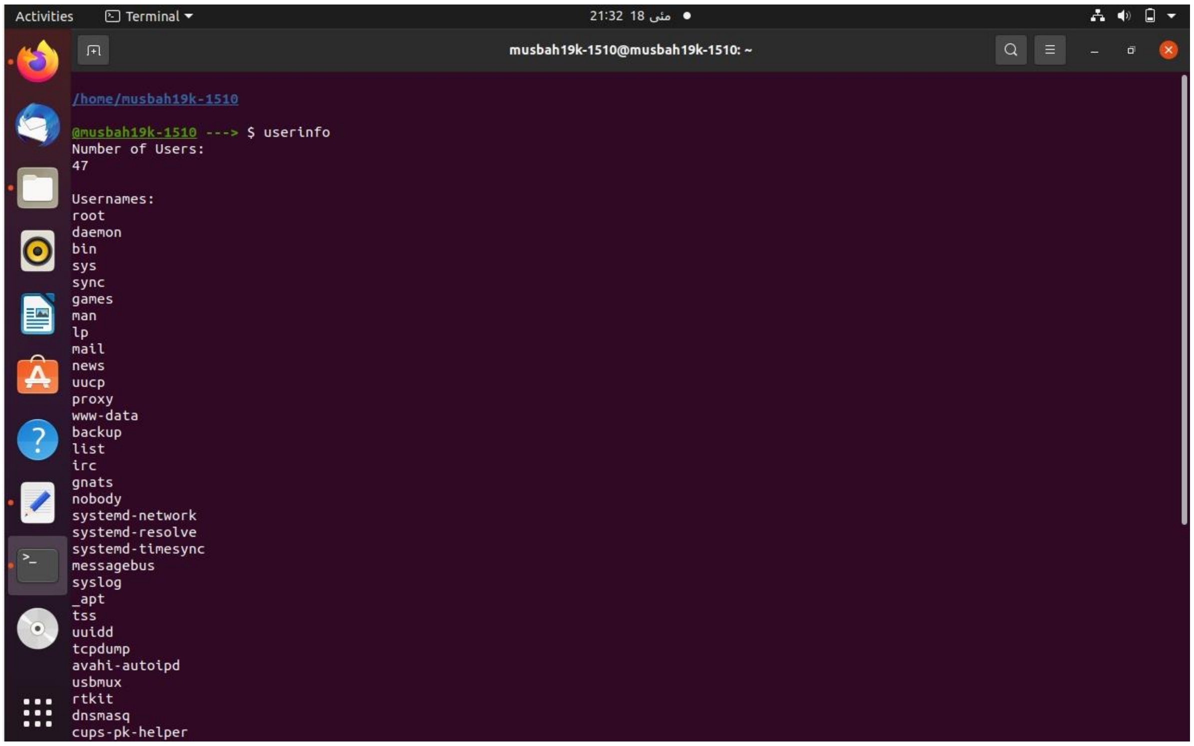
Commands	**Description**
Exit	Exits the respective shell
Topram	Displays the files/application which are used most of the RAM
Help	Displays certain instructions which would help the user in using commands in the shell
Hello	Prints hello with the user name
Showip	Shows IP address
Add	First asks for two numbers to be added and then displays their sum

Multiply	First asks for two numbers to be multiplied and then displays their product
Subtract	First asks for two numbers to be subtracted and then displays their difference
Divide	First asks for two numbers to be divided and then displays their quotient
Upc	Displays files which are starting with a upper case letter
Lc	Displays files which are starting with a lower case letter
Showdir	Displays the current directory
Run	Whatever website we write after this commands starts running in the browser
Meminfo	Displays information about the memory
Update	Does any updation needed and removes any unnecessary things
Modulo	First asks for the dividend and the divisor and then displays their modulus
Userinfo	Displays all users information
Cd	Changes the current directory
Osinfo	Displays information about our operating system
Factorial	Gives the factorial of the number provided by the user

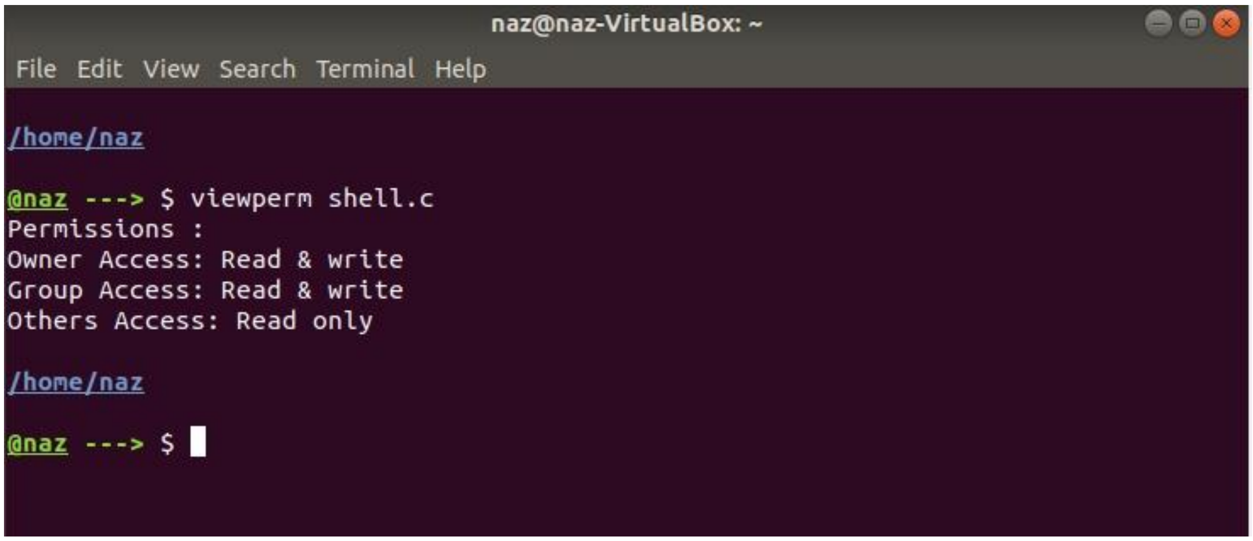
Sc	Sorts a certain file and creates its copy
Compile	Compiles the program written after this command
Viewperm	Displays permission of a file in words
Ext	Reads extension of a file and displays what type of file it is
Compare	Compare any two files, numbers or sentences and tells if they are equal or not
Search	Searches if a word is present in the sentence/paragraph given by the user
Wordfreq	This will find the frequency of a given word

❖ **Code results**

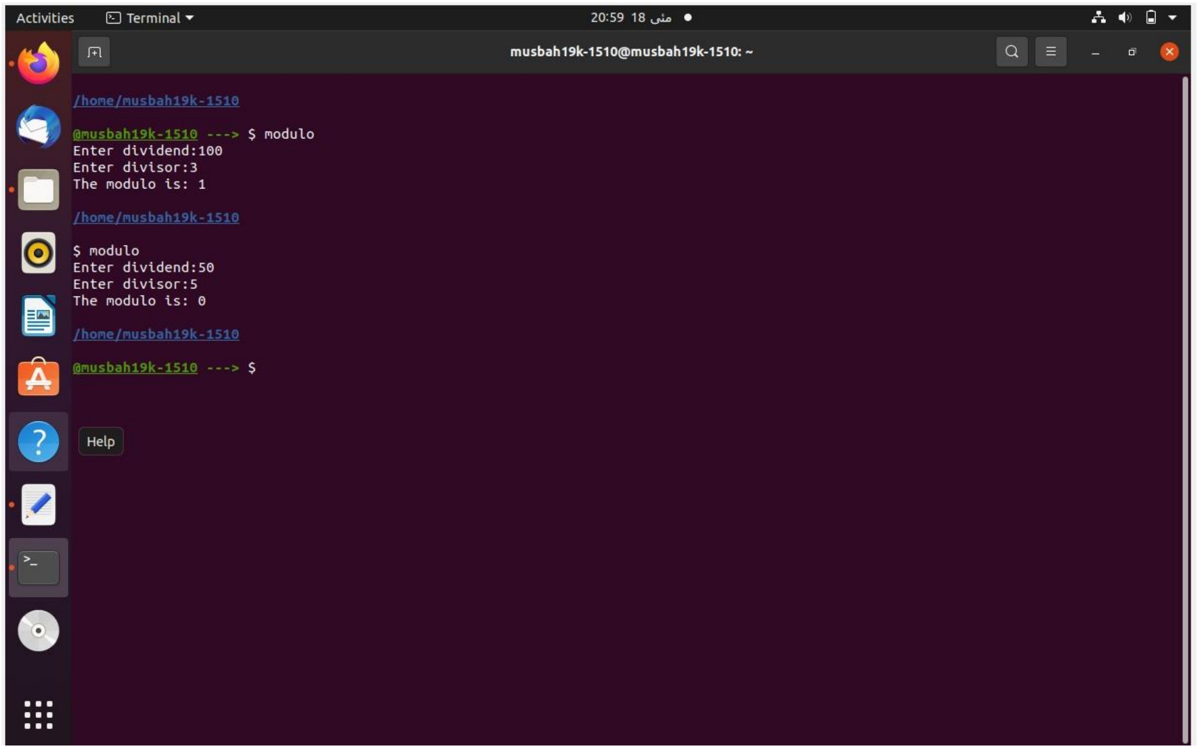
Following are the results of some of our customized commands:



```
/home/musbah19k-1510

@musbah19k-1510 ---> $ userinfo
Number of Users:
47

Usernames:
root
daemon
bin
sys
sync
games
man
lp
mail
news
uucp
proxy
www-data
backup
list
irc
gnats
nobody
systemd-network
systemd-resolve
systemd-timesync
messagebus
syslog
_apt
tss
uuidd
tcpdump
avahi-autoipd
usbmux
rtkit
dnsmasq
cups-pk-helper
```

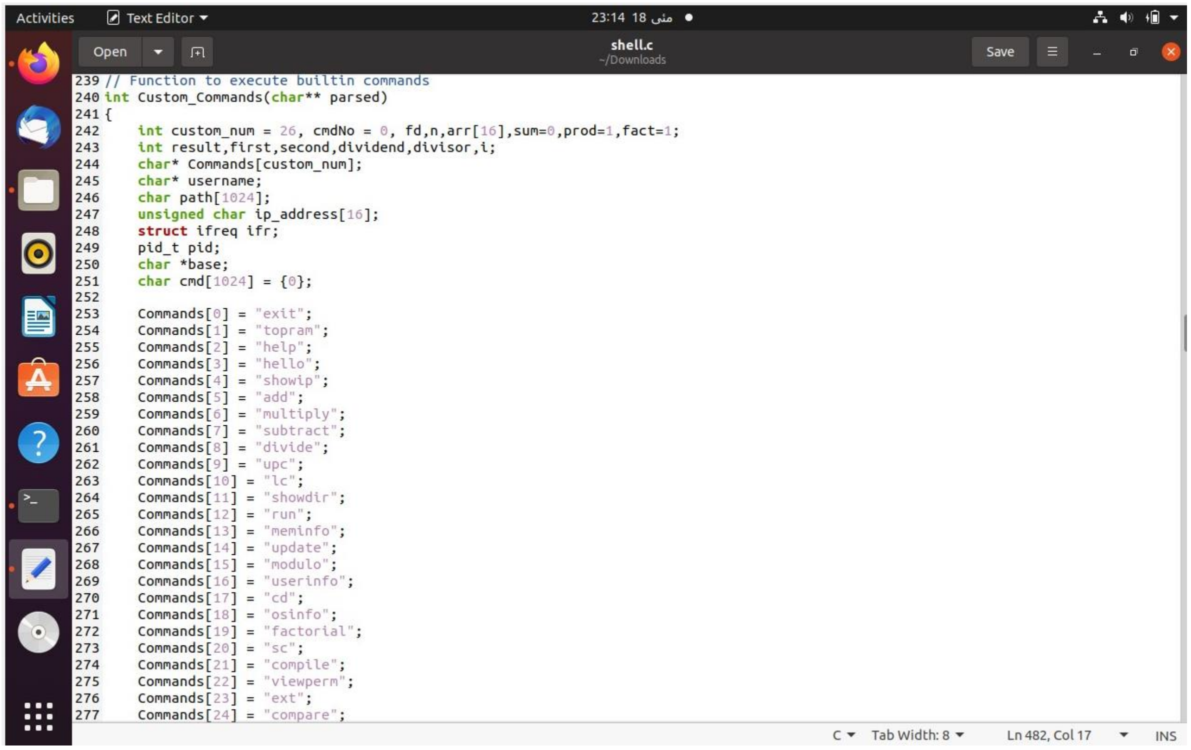


```
/home/naz

@naz ---> $ viewperm shell.c
Permissions :
Owner Access: Read & write
Group Access: Read & write
Others Access: Read only

/home/naz

@naz ---> $ 
```

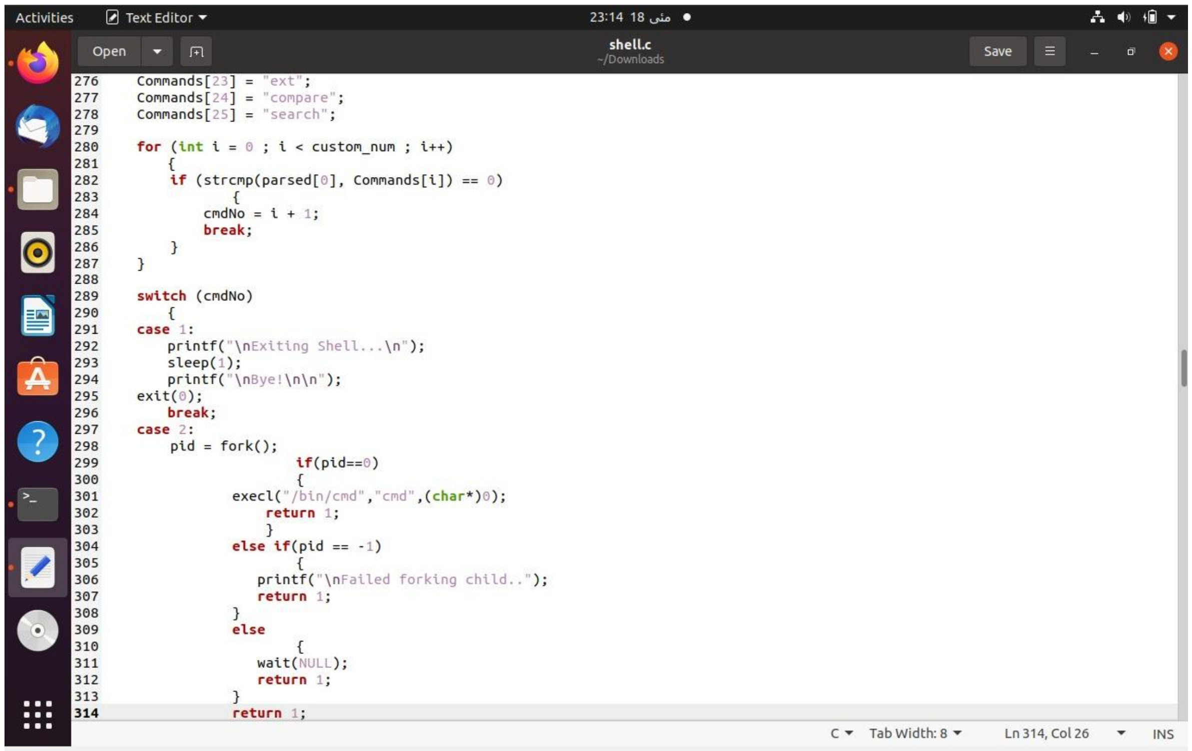


```
/home/musbah19k-1510

@musbah19k-1510 ---> $ modulo
Enter dividend:100
Enter divisor:3
The modulo is: 1

/home/musbah19k-1510

$ modulo
Enter dividend:50
Enter divisor:5
The modulo is: 0

/home/musbah19k-1510

@musbah19k-1510 ---> $
```

In the Custom_Commands function we have customized the commands using the case structure and through system calls we have executed these commands by bash scripts.



```c
// Function to execute builtin commands
int Custom_Commands(char** parsed)
{
    int custom_num = 26, cmdNo = 0, fd,n,arr[16],sum=0,prod=1,fact=1;
    int result,first,second,dividend,divisor,i;
    char* Commands[custom_num];
    char* username;
    char path[1024];
    unsigned char ip_address[16];
    struct ifreq ifr;
    pid_t pid;
    char *base;
    char cmd[1024] = {0};

    Commands[0] = "exit";
    Commands[1] = "topram";
    Commands[2] = "help";
    Commands[3] = "hello";
    Commands[4] = "showip";
    Commands[5] = "add";
    Commands[6] = "multiply";
    Commands[7] = "subtract";
    Commands[8] = "divide";
    Commands[9] = "upc";
    Commands[10] = "lc";
    Commands[11] = "showdir";
    Commands[12] = "run";
    Commands[13] = "meminfo";
    Commands[14] = "update";
    Commands[15] = "modulo";
    Commands[16] = "userinfo";
    Commands[17] = "cd";
    Commands[18] = "osinfo";
    Commands[19] = "factorial";
    Commands[20] = "sc";
    Commands[21] = "compile";
    Commands[22] = "viewperm";
    Commands[23] = "ext";
    Commands[24] = "compare";
```



```c
    Commands[23] = "ext";
    Commands[24] = "compare";
    Commands[25] = "search";

    for (int i = 0 ; i < custom_num ; i++)
        {
        if (strcmp(parsed[0], Commands[i]) == 0)
                {
            cmdNo = i + 1;
            break;
        }
    }

    switch (cmdNo)
        {
    case 1:
        printf("\nExiting Shell...\n");
        sleep(1);
        printf("\nBye!\n\n");
    exit(0);
        break;
    case 2:
         pid = fork();
                        if(pid==0)
                        {
                execl("/bin/cmd","cmd",(char*)0);
                    return 1;
                    }
                else if(pid == -1)
                        {
                   printf("\nFailed forking child..");
                   return 1;
                }
                else
                        {
                   wait(NULL);
                   return 1;
                }
                return 1;
```

❖ Conclusion

Therefore, our own new implementation of the Linux shell is easy to use and user friendly as well. Most importantly, the custom commands help the user to get their work done just within a few or in some cases one command.

❖ Acknowledgement

We would like to thank our theory teacher Miss Tania Iram and our lab instructor Miss Rabia Ansari for their sincere efforts and guidance throughout the course and for providing us with the best of materials and resources. We are grateful for studying from both of you. Thank you!