**FAST- National university of Computer and Emerging Sciences**

**CS 205: Operating Systems**

**[Unix Shell]**

**Project Proposal**

**Supervised**:

Ms.NausheenShoaib

**Group Members:**

[Izaan Sohail, 18k-0162]

[Maria Aliasghar, 18k-0161]

[Nimra Butt, 18k-1149]

1. **Objective**:

This project is based on making a mini-shell which works like the terminal on ubuntu. This shell responds to all codes and commands which has functions working in the background to perform specific tasks. Additionally we have added 15 custom commands which include commands that reduce the task done by two(2) or three(3) commands to one(1) commands, entertaining games and other usefull commands.

1. **Project Description:**

* The Mini Shell is basically based on 3 main functions:
* **DisplayPrompt**: Displays the starting Prompt: user@currect dir>
* **ExecFunction**: Preparation of a receiver input.
* **GarbageCollector**: Memory Release
* **Additionally there are 15 Custom functions we made:**
* calculator: This is a simple calculator for addition subtraction and division.
* cvrt\_code: This takes a string input and outputs equivalent barcode.
* fpermission: This command displays information about the permissions a file is given.
* Print\_Table: This command is used to display a multiplation table of the required number.
* weather: This command displays the weather forcasting and moon phases of a perticular place and a particular date.
* choo\_choo: This is a fun command , try it out.
* current\_time: This command displays the current time in hours , minutes and seconds.
* guessing\_game: Try this simple number guessing game in which you have 3 guesses.
* cmp\_files: This command takes file names as input and decides whether the files are identical or not.
* mem\_occ: This command displays the details of memory occupied by each running process.
* quotes: This displays inspirational quotes.
* cvt\_pdf: command converts a .doc file to .pdf file format.
* encrypt: This command is used to encrypt a string.
* MP4TMP3: This command is used to convert a mp4 video to mp3.
* todo\_list: This is a user friendly todo list in which you can add your tasks and mark them done.
* cus\_help: Manual for all custom commands.

1. Code:

#include <stdio.h>

#include <sys/types.h>

#include <pwd.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/wait.h>

#include <string.h>

#include <sys/mman.h>

#include <errno.h>

#define INPUT\_SIZE 510 //length of the maximum string for the user

#define CUTTING\_WORD " \n" //For dividing string into single words

#define ENDING\_WORD "done" //Program end word

#define RESET 0

void DisplayPrompt(); //Display Prompt : user@current dir>

char\*\* execFunction(char \*input,char \*\*argv,int \*sizeOfArray, int \*cmdLength); //Preparation of user input

void garbageCollector(char\*\* argv, int size); //Memory Release

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int \*numOfCmd;

int \*cmdLength;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main(){

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

numOfCmd = mmap(NULL, sizeof \*numOfCmd, PROT\_READ | PROT\_WRITE, MAP\_SHARED | MAP\_ANONYMOUS, -1, RESET);

cmdLength = mmap(NULL, sizeof \*cmdLength, PROT\_READ | PROT\_WRITE, MAP\_SHARED | MAP\_ANONYMOUS, -1, RESET);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

(\*numOfCmd)=RESET;

(\*cmdLength)=RESET;

int sizeOfArray=RESET;

char input[510]=""; //A String Array containing the input

printf("\n\n\n\n\n\n\*\*\*\*\*\*\*WELCOME TO MY SHELL\*\*\*\*\*\*\*");

printf("\n\n\n\*\*\*\*DEVELOPED BY : ");

printf("\n\*\*\*K18-0161");

printf("\n\*\*\*K18-0162");

printf("\n\*\*\*K18-1149\n\n");

DisplayPrompt(); //Display Prompt : user@current dir>

pid\_t id; //pid\_t use for process identifier

char \*\*argv; //A string array containing the program name and command arguments

while(strcmp(input,ENDING\_WORD)!=RESET)

{

if(fgets(input,INPUT\_SIZE,stdin)==RESET)

printf(" ");

//do nothing...count time regular

argv=execFunction(input,argv,&sizeOfArray,cmdLength);

if(strcmp("cd",argv[RESET])==RESET)

{

struct passwd \*pwd;

char\* path=argv[1];

if(path==NULL)

{

pwd=getpwuid(getuid());

path=pwd->pw\_dir;

}

if(path[0]=='/')

(path)=++(path);

errno=chdir(path);

DisplayPrompt();

if(errno!=RESET)

printf("error changing directory");

}

else

{

id=fork();

if(id<RESET)

{

printf("fork failed");

exit(RESET);

}

else if(id==RESET) {

(\*numOfCmd)++;

execvp(argv[RESET],argv);

garbageCollector(argv,sizeOfArray);

if(strcmp(input,ENDING\_WORD)!=RESET)

printf("command not found\n");

exit(1);

}else {

wait(&id);

if(strcmp(input,ENDING\_WORD) != RESET)

{

DisplayPrompt();

} else {

printf("Num of cmd: %d\n", \*numOfCmd);

printf("cmd length: %d\n", \*cmdLength-4);

printf("Bye \n");

garbageCollector(argv,sizeOfArray);

}

}

}

}

}

void garbageCollector(char\*\* argv,int size)

{

int i=RESET;

for (i = RESET; i < size; ++i) {

free(argv[i]);

}

free(argv);

argv=NULL;

}

char\*\* execFunction(char \*input,char \*\*argv,int \*sizeOfArray,int \*cmdLength)

{

int i=RESET,counter=RESET;

char inputCopy[INPUT\_SIZE];

strcpy(inputCopy,input);

char\* ptr= strtok(input,CUTTING\_WORD);

while(ptr!=NULL)

{

ptr=strtok(NULL,CUTTING\_WORD);

counter++;

}

argv = (char\*\*)malloc((counter+1)\*(sizeof(char\*)));

if(argv==NULL)

{

printf("error allocated");

exit(RESET);

}

char\* ptrCopy= strtok(inputCopy,CUTTING\_WORD);

while(ptrCopy!=NULL)

{

if (i==RESET)

(\*cmdLength)+=strlen(ptrCopy);

argv[i]=(char\*)malloc((sizeof(char)+1)\*strlen(ptrCopy));

if(argv[i]==NULL)

{

printf("error allocated");

for (int j = i-1; j >-1 ; j--) {

free(argv[j]);

}

free(argv);

exit(RESET);

}

strcpy(argv[i],ptrCopy);

argv[i][strlen(ptrCopy)]='\0';

ptrCopy=strtok(NULL,CUTTING\_WORD );

i++;

}

argv[counter]=NULL;

(\*sizeOfArray)=counter;

return argv;

}

void DisplayPrompt()

{

//-------------------show the path-----------------------------

long size;

char \*buf;

char \*ptr;

size = pathconf(".", \_PC\_PATH\_MAX);

if ((buf = (char \*)malloc((size\_t)size)) != NULL)

ptr = getcwd(buf, (size\_t)size);

//----------show the user name root------------------------

struct passwd \*getpwuid(uid\_t uid);

struct passwd \*p;

uid\_t uid=0;

if ((p = getpwuid(uid)) == NULL)

perror("getpwuid() error");

else {

printf("%s@%s>", p->pw\_name, ptr);

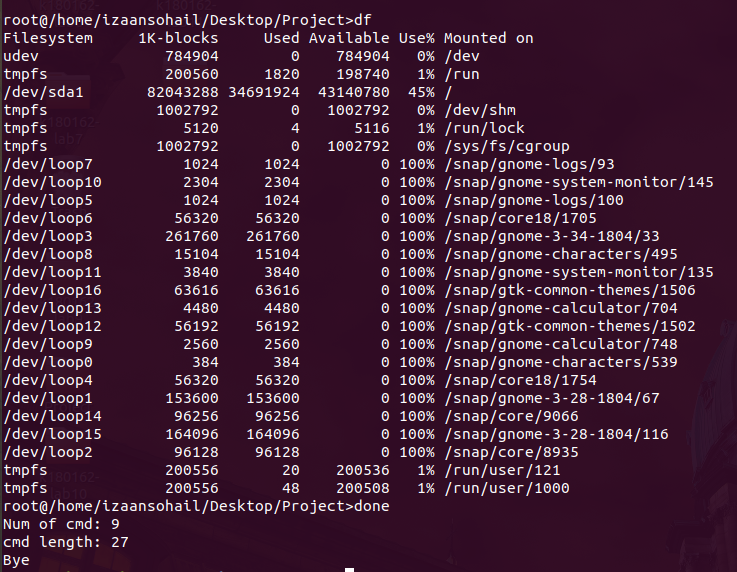
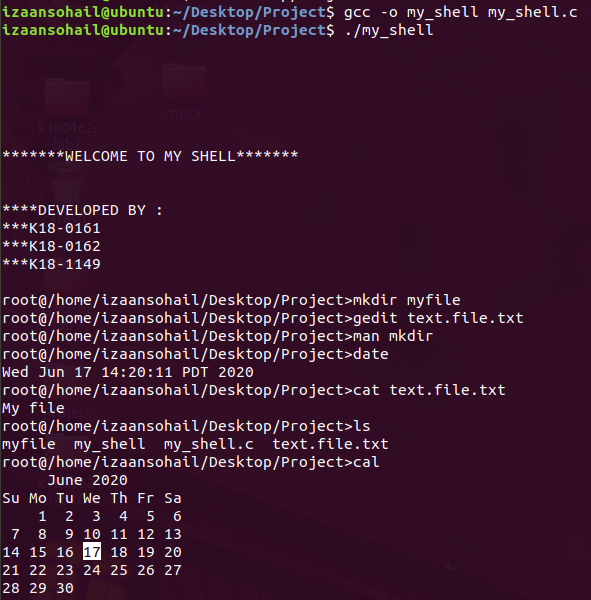
}

free(buf);

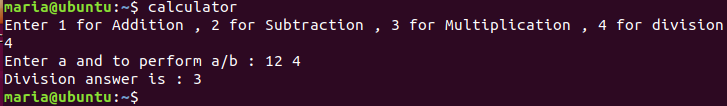
}

1. **Screenshots:**

* Unix-Shell:



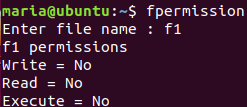
* Custom Commands:
* Calculator:



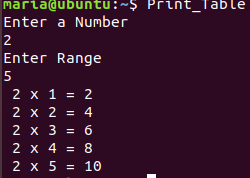
* Cvrt\_code:



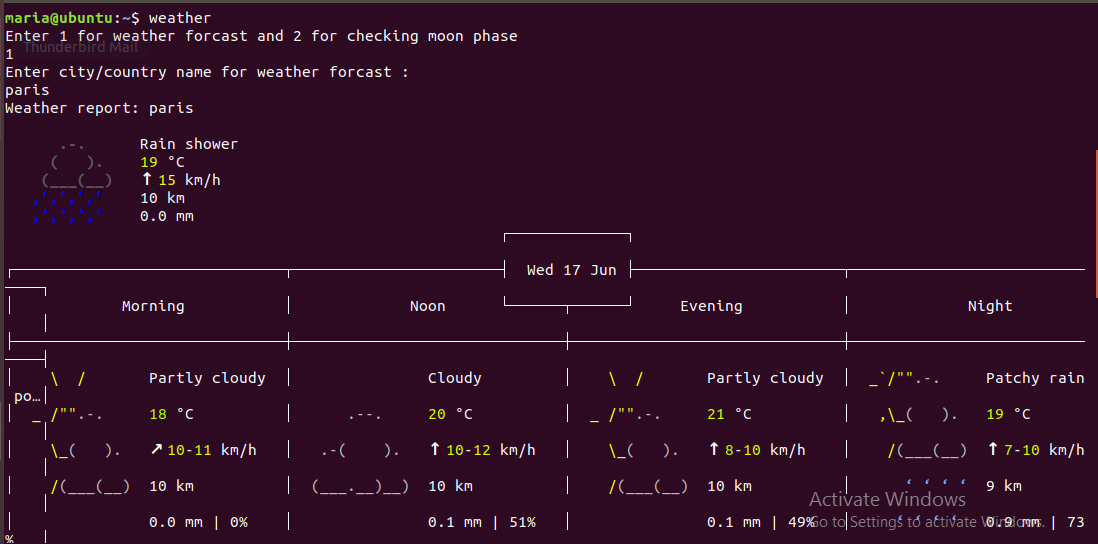
* Fpermission:



* Print\_Table:



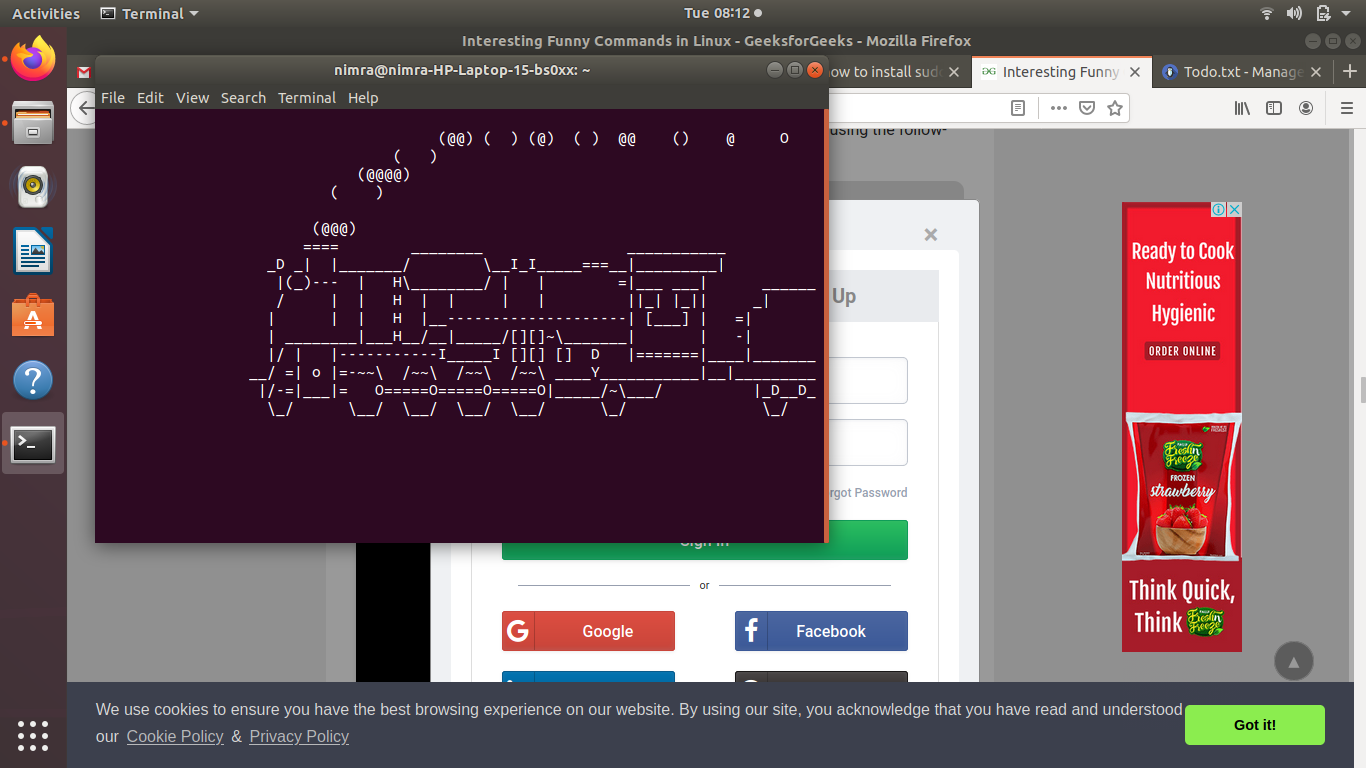
* Weather:



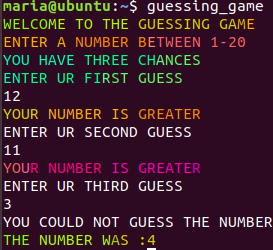
* Current\_time:

ct.png

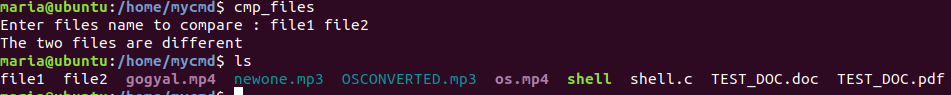
* Choo\_choo:



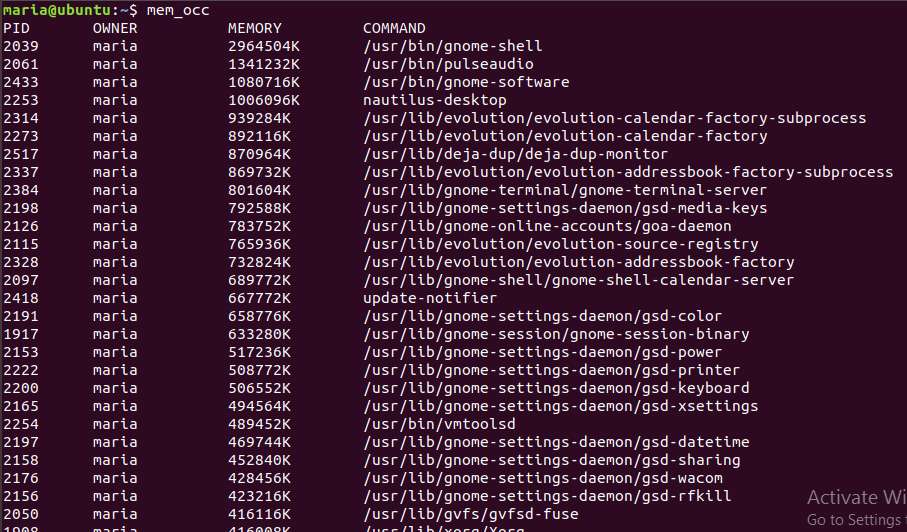
* Guessing\_game:



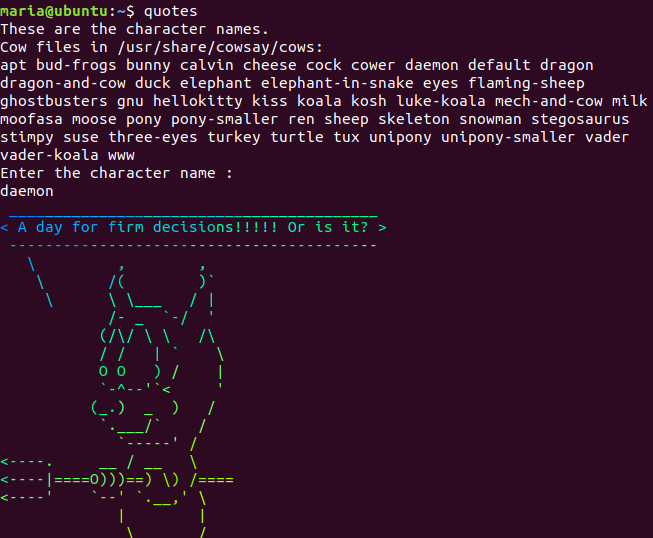
* Cmp\_files:



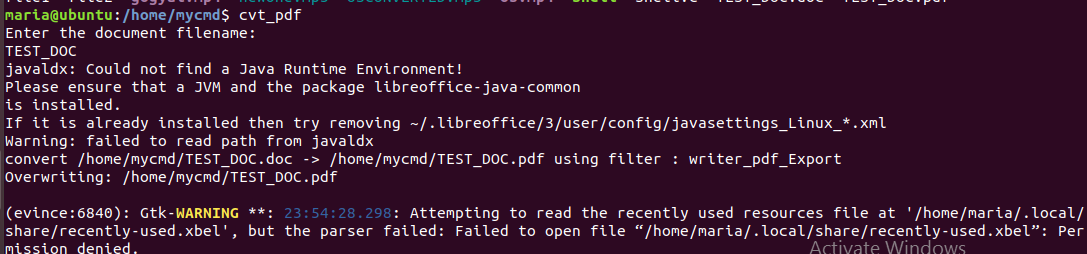
* Men\_occ:

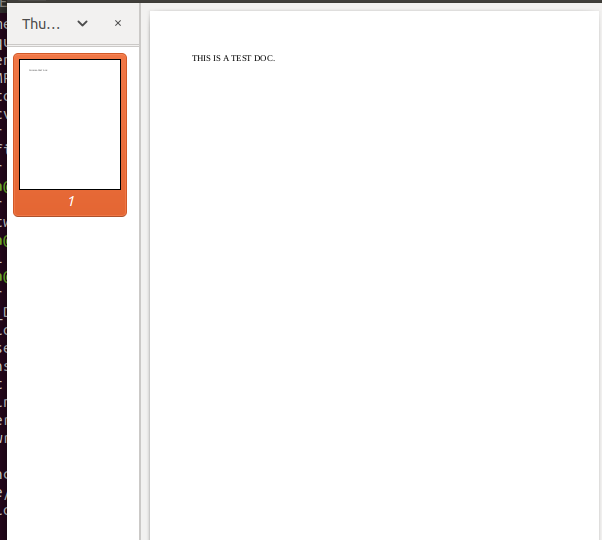


* Quotes:

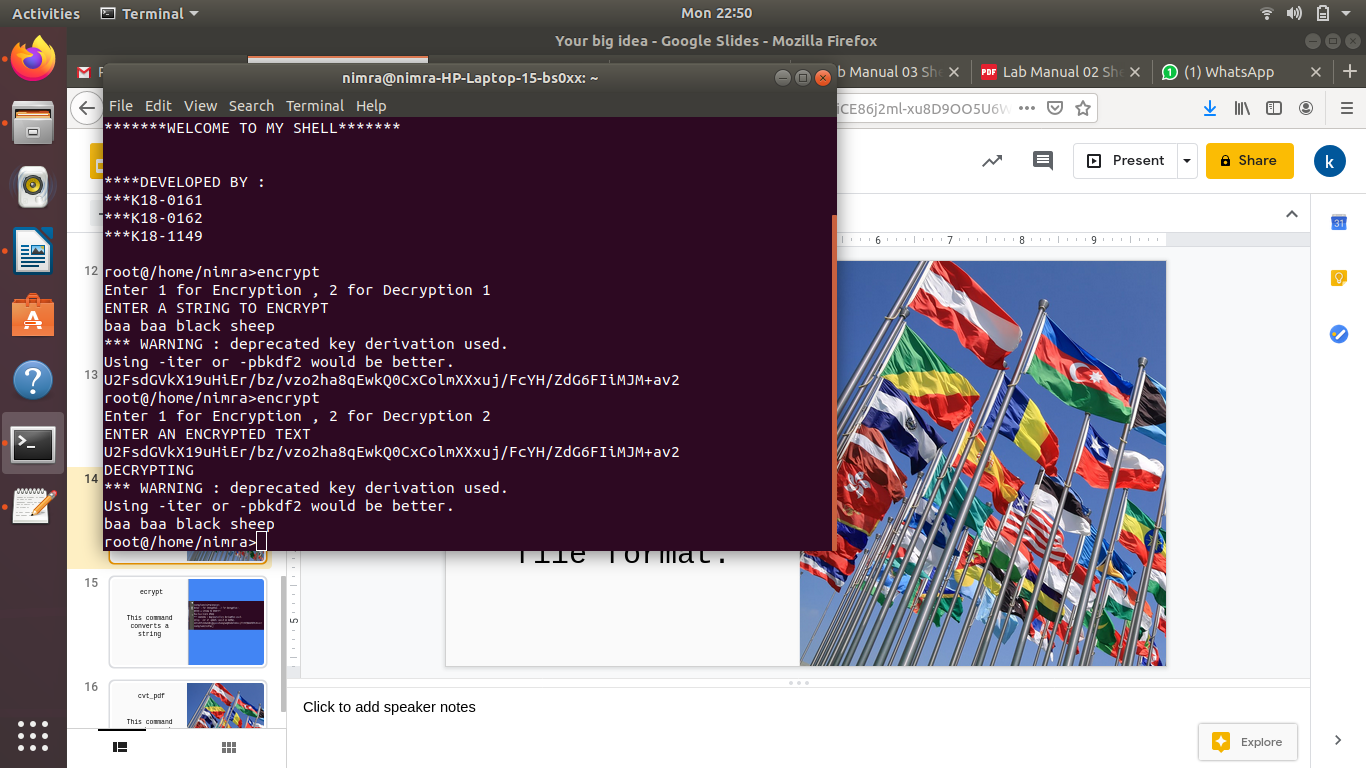


* Cvt\_pdf;:

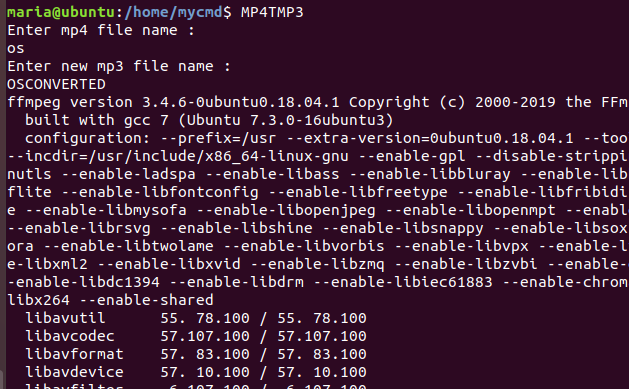




* Encrypt:

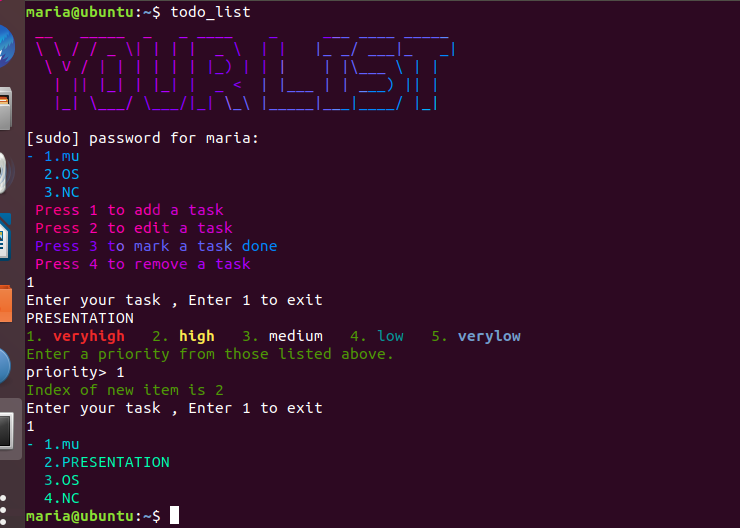


* MP4TMP3:

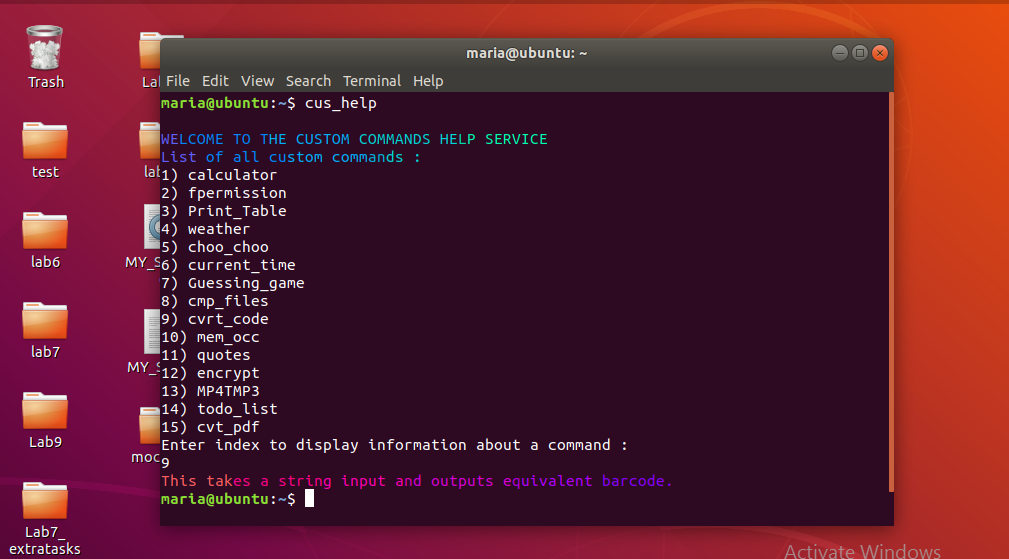




* Todo\_List:



* Cus\_help:



1. **Conclusion:**

This shell still has a lot of scope of improvement and many more commands can be increased which can be really complex but can reduce the workload and complexity of the user but this shell is a mini shell which covers almost all basic mini shell commands.