A

MINI PROJECT REPORT

ON

**FILE MANAGEMENT SYSTEM in LINUX Using SHELL SCRIPT**

Submitted in partial fulfillment of the requirements

of the degree of

**Bachelor of Engineering**

**In**

**Computer Engineering**

By

(Roll No. 26)(Himani Joshi)

(Roll No. 46)(Prachi Nimase)

(Roll No. 42)(Devvrat Mungekar)

**Under the guidance of**

**Prof. Hasib Shaikh**



**Department of Computer Engineering**

K.C. College of Engineering and Management Studies And

Research, Thane (E)

Mumbai University

2018-2019

**ACKNOWLEDGEMENT**

No project is ever complete without the guidance of those expert who have already traded this past before and hence become master of it and as a result, our leader. So we would like to take this opportunity to take all those individuals who have helped us in visualizing this project.

We express our deep gratitude to our mini project guide Prof. Hasib Shaikh for providing timely assistant to our query and guidance that she gave owing to her experience in this field for past many years. She has indeed been a lighthouse for us in this journey.

We extend our sincerity appreciation to all our Professors from K.C. College of Engineering & Management Studies & Research for their valuable suggestions during the designing of the project. Their contributions have been valuable in so many ways that we find it difficult to acknowledge of them individual.

We also express our gratitude to our HOD Prof. Mandar Ganjapurkar for extending his help directly and indirectly through various channels in our project work.

Thanking You.

**CERTIFICATE**

This is to certify that the report on mini project entitled,

**PROJECT TITLE**

**FILE MANAGEMENT SYSTEM in LINUX Using SHELL SCRIPT**

**Project Members**

(ROLL NO. 26) (Himani Joshi)

(ROLL NO. 46) (Prachi Nimase)

(ROLL NO. 42) (Devvrat Mungekar)

As a partial fulfillment for Bachelor of Computer Engineering

Degree course of Mumbai University for Academic year 2018-2019

**INTERNAL GUIDE**

Prof. Hasib Shaikh

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

INTERNAL EXAMINER EXTERNAL EXAMINER

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

HOD PRINCIPAL

(Prof. Mandar Ganjapurkar) (Dr. Vijay Kumar N. Pawar)

**TABLE OF CONTENTS**

**Chapter No Name of Chapter Page No**

1. INTRODUCTION 01
2. PROGRAM CODE 02
3. OUTPUT SCREENSHOTS 03
4. APPLICATIONS 04
5. REFERENCES 05

**INTRODUCTION:**

A shell script is a text [file](https://whatis.techtarget.com/definition/file) that contains a sequence of commands for a [UNIX](https://searchdatacenter.techtarget.com/definition/Unix)-based [operating system](https://whatis.techtarget.com/definition/operating-system-OS). It's called a shell script because it combines into a "script" in a single file a sequence of commands that would otherwise have to be presented to the system from a keyboard one at a time. The [shell](https://searchdatacenter.techtarget.com/definition/shell) is the operating system's command interpreter and the set of commands you use to communicate with the system. A shell script is usually created for command sequences for which a user has a repeated need. You initiate the sequence of commands in the shell script by simply entering the name of the shell script on a command line.

In the [DOS](https://searchsecurity.techtarget.com/definition/DOS) operating system, a shell script is called a [batch file](https://searchwindowsserver.techtarget.com/definition/batch-file). In IBM's mainframe VM operating systems, it's called an EXEC.

A **shell script** is a [computer program](https://en.wikipedia.org/wiki/Computer_program) designed to be run by the [Unix shell](https://en.wikipedia.org/wiki/Unix_shell), a [command-line interpreter](https://en.wikipedia.org/wiki/Command-line_interpreter).[[1]](https://en.wikipedia.org/wiki/Shell_script#cite_note-1) The various dialects of shell scripts are considered to be [scripting languages](https://en.wikipedia.org/wiki/Scripting_language). Typical operations performed by shell scripts include file manipulation, program execution, and printing text. A script which sets up the environment, runs the program, and does any necessary cleanup, logging, etc. is called a **wrapper**.

A shell script can provide a convenient variation of a system command where special environment settings, command options, or post-processing apply automatically, but in a way that allows the new script to still act as a fully normal Unix command.

**Program Code:**

#! /usr/bin/bash

function file\_management() {

while :

do

clear

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

echo

echo " MAIN MENU "

echo

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

echo "1.Create a new file."

echo "2.Add contents in the file."

echo "3.Re-write the file."

echo "4.Rename the file."

echo "5.Copy the contents of the file."

echo "6.Read the contents of the file."

echo "7.Delete the file."

echo "8.Display the files in the directory."

echo "9.Quit!"

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

read -p "Enter your choice : " ch

case $ch in

1) read -p "Enter the new file name with the extension : " file\_name

touch $file\_name #creating new file

echo "File created!"

echo "Press a key. . ." ; read

;;

2) read -p "Enter the file name that you want to open : " file\_name

if [ -e $file\_name ] #if file exist..

then

if [ -w $file\_name ] #if file has write permission..

then

echo "Press 'Ctrl+D' to save and close the file."

cat >> $file\_name

else

echo "$file\_name doesn't have a write permission!"

read -p "Do you want to access the file? (y / n) : " question

if [ $question = "y" ]

then

read -sp "Enter the password : " passw

if [ "$passw" -eq "$password" ]

then

chmod +w $file\_name

sleep 1

echo "Enter the contents you want to enter."

echo "Press 'Ctrl+D' to save and close the file."

cat >> $file\_name

fi

fi

fi

fi

echo "Press a key. . ." ; read

;;

3) read -p "Enter the file name that you want to open : " file\_name

if [ -e $file\_name ]

then

if [ -w $file\_name ]

then

echo "Press 'Ctrl+D' to save and close the file."

cat > $file\_name #Overwriting the file

else

echo "$file\_name doesn't have a write permission!"

read -p "Do you want to access the file? (y / n) : " question

if [ $question = "y" ]

then

read -sp "Enter the password : " passw

if [ "$passw" -eq "$password" ]

then

chmod +w $file\_name

sleep 1

echo "Enter the contents you want to enter."

echo "Press 'Ctrl+D' to save and close the file."

cat > $file\_name

fi

fi

fi

fi

echo "Press a key. . ." ; read

;;

4) read -p "Enter the file name that you want to rename : " file\_name

if [ -e $file\_name ]

then

read -p "Enter the new file name you want : " new\_file

mv $file\_name $new\_file

echo "File renamed successfully . . . . . . "

else

echo "$file\_name not found!"

fi

echo "Press a key . . ." ; read

;;

5) read -p "Enter File name to copy the contents from : " file\_name\_1

read -p "Enter File name in which you want to copy data : " file\_name\_2

if [ -e $file\_name ] #if [ -f $f1 ]

then

cp $file\_name\_1 $file\_name\_2

echo "File copied..."

else

echo "$file\_name\_2 does not exist!"

read -p "Do you want to create a new file? Yes(y)/No(n) : " question

if [ $question = "y" ]

then

read -p "Enter the new file name with the extension : " file\_name

touch $file\_name

echo "File created!"

cp $file\_name\_1 $file\_name

echo "File copied..."

fi

fi

echo "Press a key . . ." ; read

;;

6) read -p "Enter the file name you wish to read : " file\_name

if [ -e $file\_name ]

then

if [ -r $file\_name ]

then

#IFS = "Internal field generator" used by shell to determine how to do word splitting

#Give space after '=' as we're assigning IFS to the space

while IFS= read -r line

#Another way to give the space

#while IFS='' read -r line

do

echo "$line"

done < $file\_name

else

echo "$file\_name doesn't have read permission!"

read -p "Do you want to access the file? (y / n) : " question

if [ $question = "y" ]

then

read -sp "Enter the password : " passw

if [ "$passw" -eq "$password" ]

then

chmod +r $file\_name

echo "Reading the contents . . . "

sleep 1

while IFS= read -r line

#Another way to give the space

#while IFS='' read -r line

do

echo "$line"

done < $file\_name

fi

fi

fi

else

echo "$file\_name does not exist!"

fi

echo "Press a key . . ." ; read

;;

7) read -p "Enter the file name you want to delete : " file\_name

if [ -e $file\_name ]

then

if [ -r $file\_name ]

then

if [ -s $file\_name ] #If file is empty or not?

then

read -p "$file\_name contains some data in it. Do you wish to delete? Yes(y)/No(n) : " question

if [ $question = "y" ]

then

rm -i $file\_name

else

echo "$file\_name not deleted!"

fi

else

rm -i $file\_name

fi

else

echo "$file\_name doesn't have a read permission!"

fi

fi

echo "Press a key . . ." ; read

;;

8) read -p "Do you want the detailed information of the file? (y / n) : " question

if [ $question = "y" ]

then

echo

echo "Reading the files in the `pwd` . . ."

sleep 1 #10 secs delay

echo #leaving a line

ls -al

else

echo

echo "Reading the files in the `pwd` . . ."

sleep 1 #10 secs delay

echo #leaving a line

ls

fi

echo "Press a key . . ." ; read

;;

9) read -p "Do you want to exit? (y/n) : " question

if [ $question = "y" ]

then

developer

sleep 10

exit 0

fi

echo "Press a key . . ." ; read

;;

esac

done

}

function developer() {

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

echo

echo " Developed by "

echo " 'Himani Joshi' 'Prachi Nimase' 'Devvrat Mungekar' "

echo

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

}

username=Devvrat

password=123

clear

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

echo

echo " Welcome to Linus File Management System "

echo

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

echo

echo " It is the secured File Management System developed for user friendliness "

echo

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

echo

echo " Please provide the Username and Password to SIGN IN "

echo

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

sleep 1

read -p "Enter the username : " user\_name

read -sp "Enter the password to access : " pass\_word

echo

if [ "$user\_name" = "$username" ]

then

if [ "$pass\_word" -eq "$password" ]

then

echo "Logged in!"

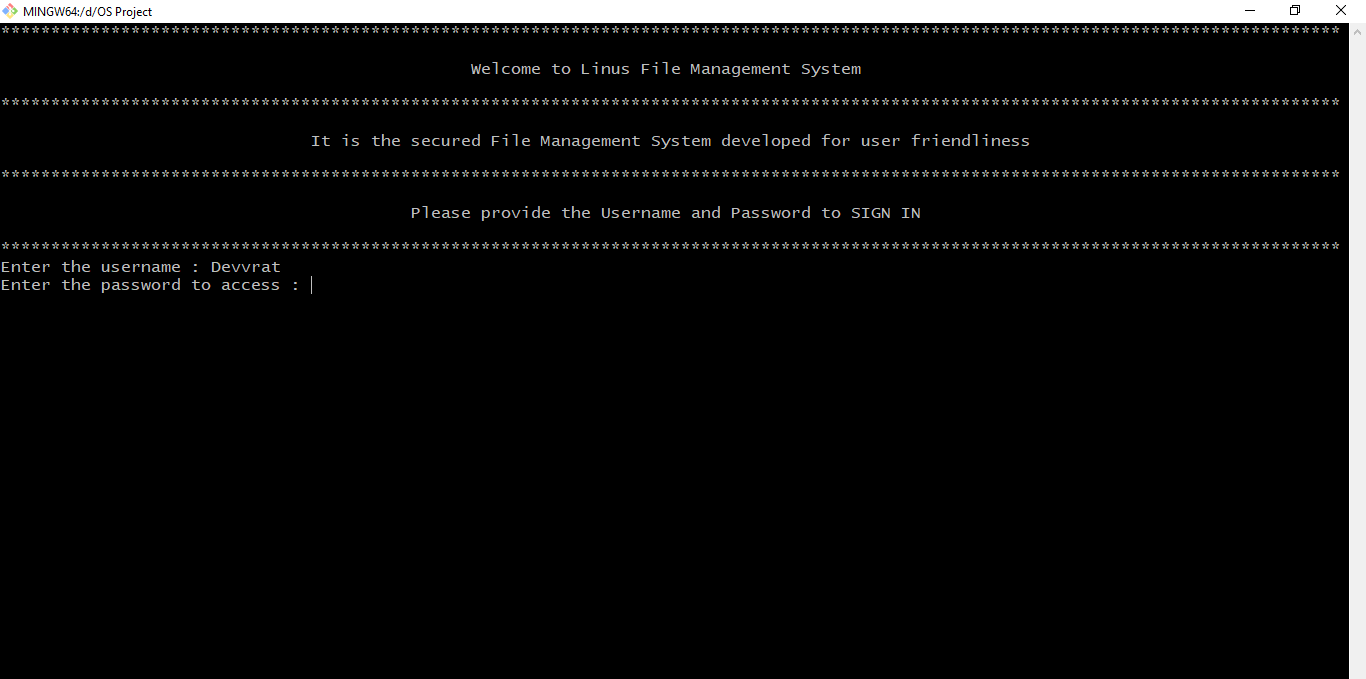
sleep 1

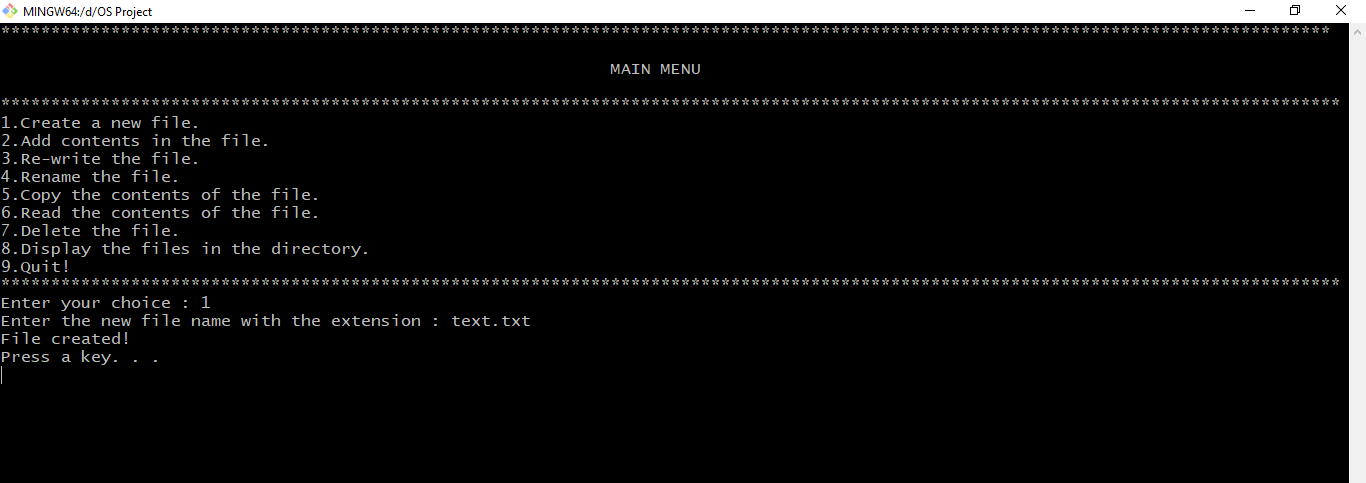
file\_management

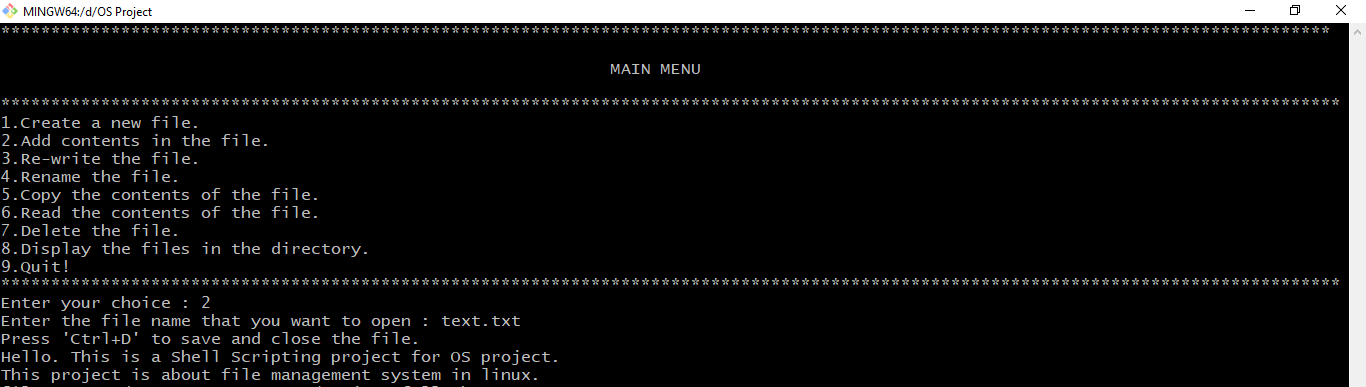
fi

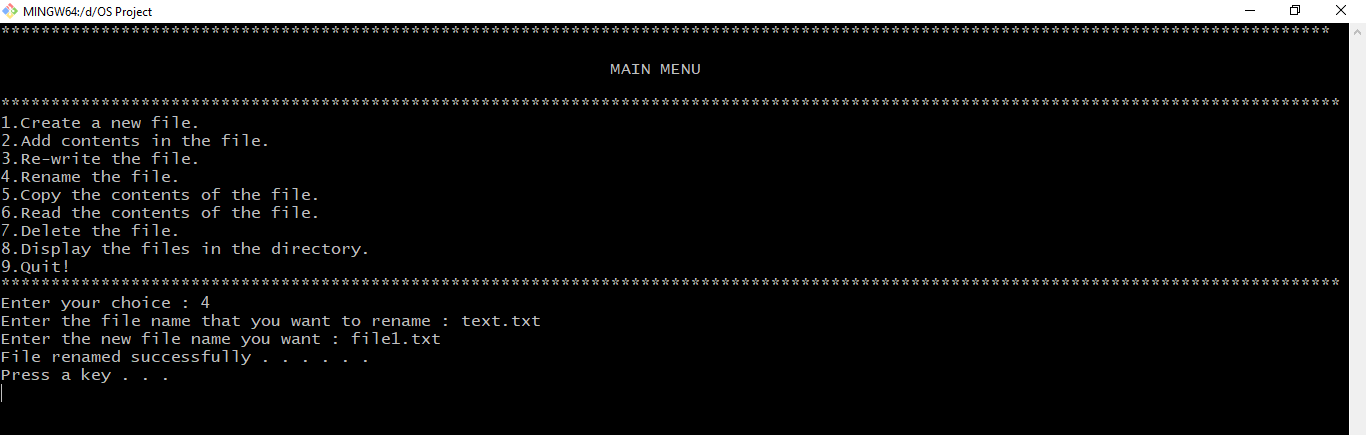
fi

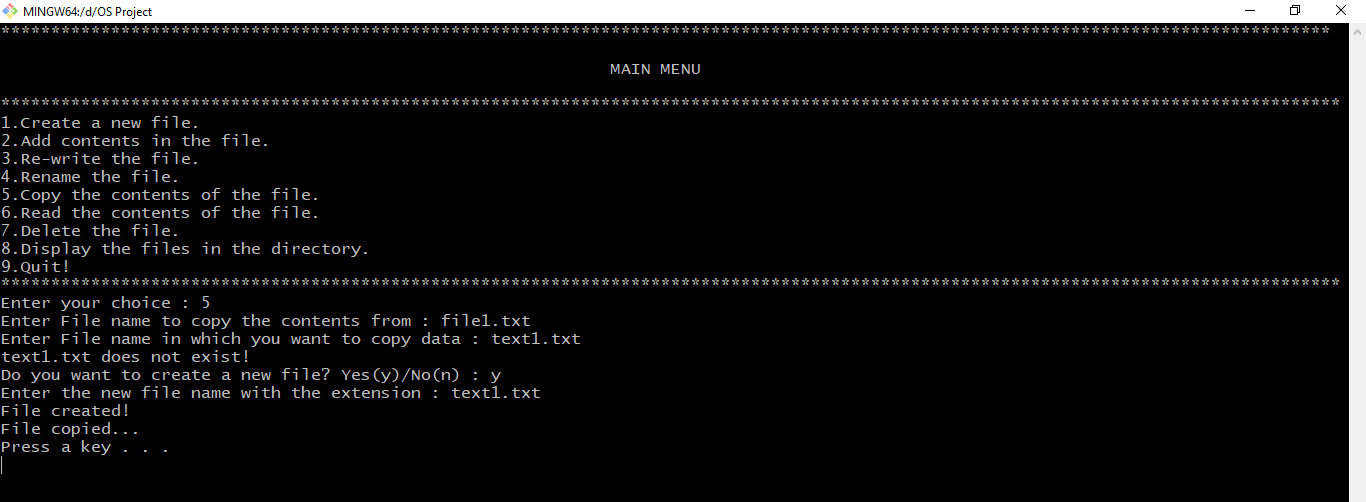
**Output Screenshots:**

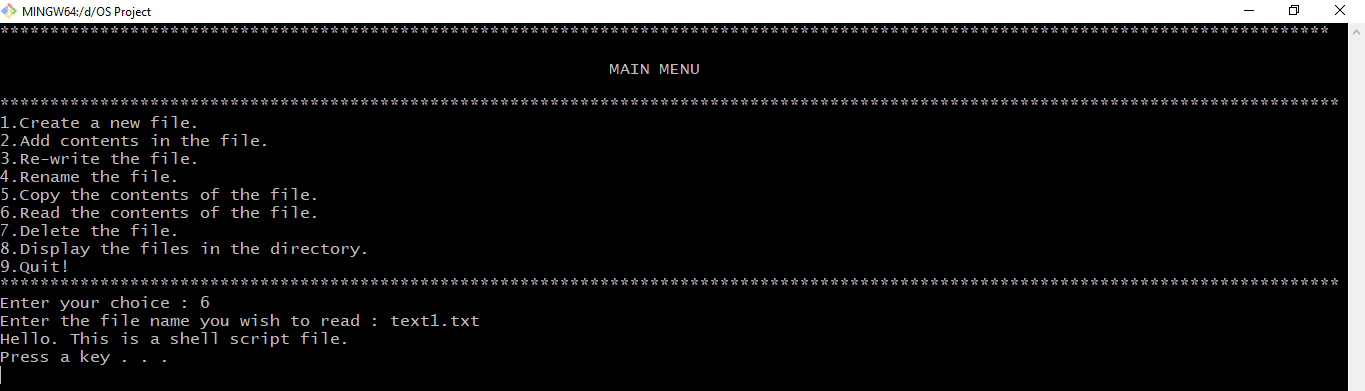


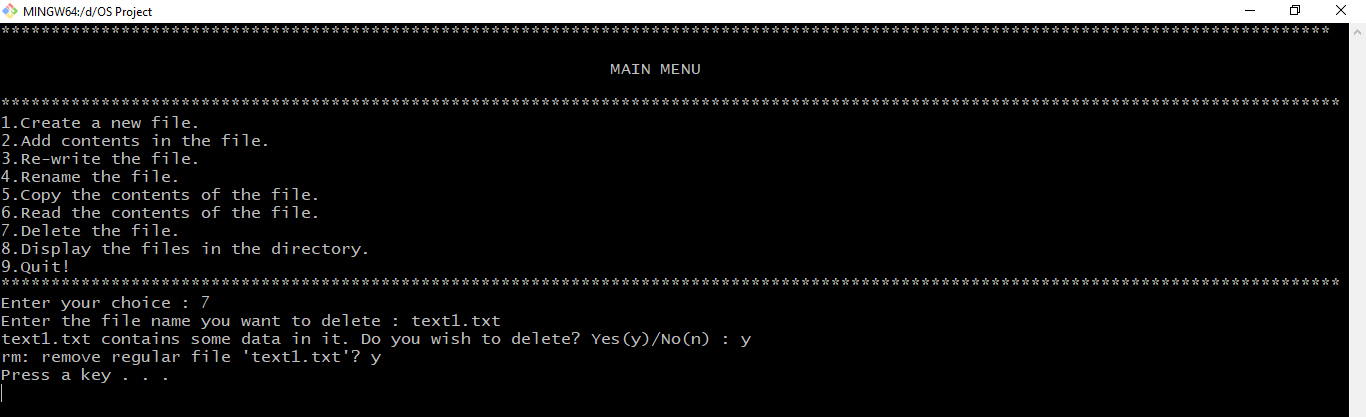


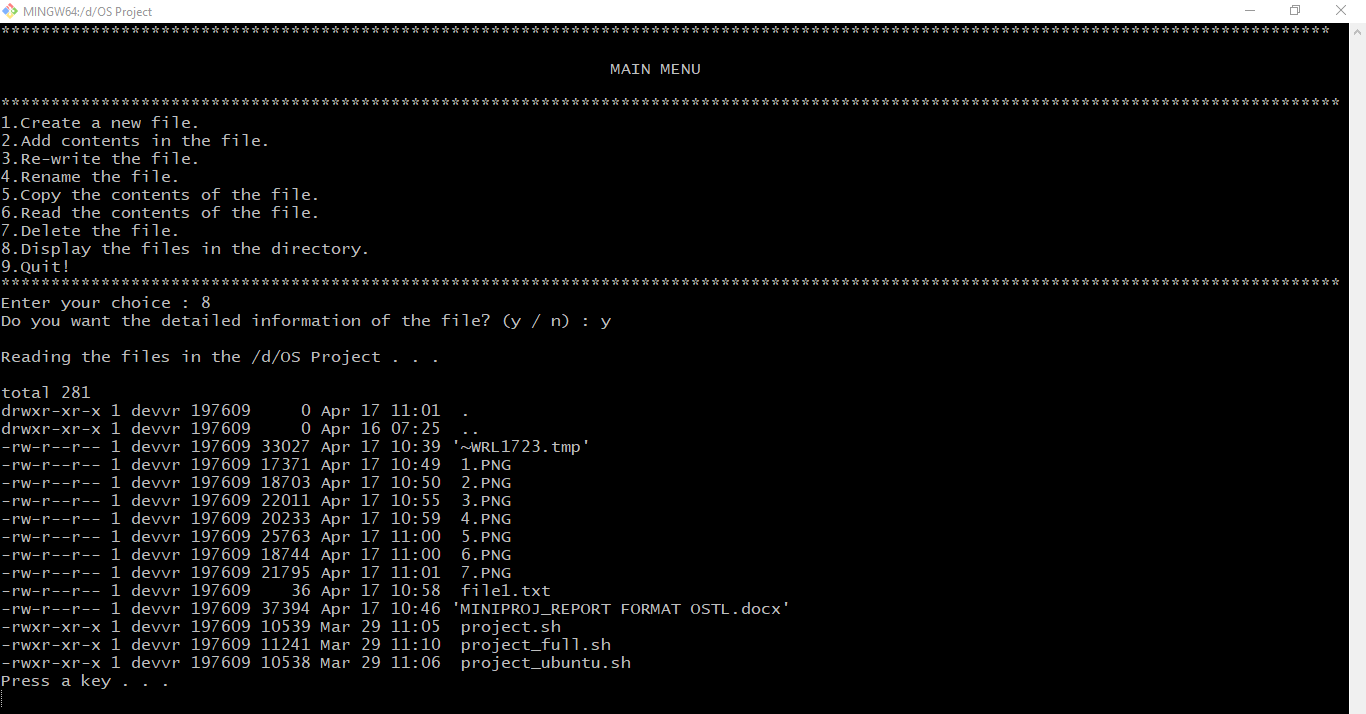














**Applications:**

1. You can write a script to initialize something at boot time of the system. so you don't need to do manually.
2. You can write a script which installs per-requisite and build the code with user input to enable/disable some features.
3. To kill or start multiple applications together.
4. To observe large database of files and find some patterns out of it.
5. Combine lengthy and repetitive sequences of commands into a single, simple command.
6. Generalize a sequence of operations on one set of data, into a procedure that can be applied to any similar set of data.
7. Create new commands using combinations of utilities in ways the original authors never thought of.
8. Simple shell scripts might be written as shell aliases, but the script can be made available to all users and all processes. Shell aliases apply only to the current shell.
9. Wrap programs over which you have no control inside an environment that you can control.
10. Create customized datasets on the fly, and call applications (e.g. matlab, sas, idl, gnuplot) to work on them, or create customized application commands/procedures.
11. Rapid prototyping (but avoid letting prototypes become production)

**Reference:**

https://www.youtube.com/playlist?list=PLS1QulWo1RIYmaxcEqw5JhK3b-6rgdWO\_

https://www.dartmouth.edu/~rc/classes/ksh/why.html