



Universidad Nacional Autónoma De
México

Facultad de Ingeniería

Proteco Generación 44

Proyecto

Linux

Alquicira Peña Luis Enrique

González Frías Ana Paula

22 Abril 2023

0.1 Introduction

En este proyecto se va a mostrar así como demostrar nuestros conocimientos como prebecarios; conocimientos que fueron adquiridos a lo largo de las dos semanas del curso de Linux.

Recordando que Linux es un sistema operativo de código abierto y gratuito, basado en Unix, creado por un programados finlandés, Linux Torvalds, conocido como desarrollador principal del Kernel de Linux.

Nuestro primer contacto como prebecarios en este curso fue su instalación nativa del sistema, rescatando que lo que se va a mostrar fue programado en terminal de LinuxMint y Fedora 37.

Ahora bien en Linux, vine el uso del trabajo con archivos, los cuales van a ser organizados en una estructura jerárquica de directorios, con el directorio raíz representado por `/`. Los archivos pueden ser de diferentes tipos, incluyendo archivos regulares, directorios, enlaces simbólicos; para verlos así como para crear, eliminar, mover o copiarlos, se utilizan comandos como `ls`, `touch`, `rm`, `mv` y `cp`; comandos esenciales para manejar y organizar los archivos en el sistema operativo Linux.

Otro tema a retomar son los comandos los cuales nos van a ser bastante útiles como herramientas de línea de comandos utilizadas para realizar diversas tareas, como manejar archivos y directorios, administrar procesos, configurar el sistema, hacer descargas, etc. Técnicamente este proyecto su una clara muestra de su uso; este uso y explicación se va a dar más a fondo cuando vayamos explicando cada parte de nuestros script.

No hay mejor forma que demostrar el conocimiento que mostrando o realizando tus logros, y nuestros scripts o sea nuestros archivos `.txt` y `.sh` que contienen series de comandos que se van a ejecutar según la secuencia de nuestros codigos, así como su utilidad. Cabe destacar que estos scripts los programamos en un lenguaje de programación llamado shell scripting, como programadores o como a los usuarios a los que se los mostraremos les permitirá automatizar tareas y realizar operaciones complejas en el sistema operativo.

Como último punto para este proyecto se nos asignó un tiempo considerable para realizarlo, se nos complicó a veces pero creemos que se logró concluir bien como lo van a poder observar a las siguientes explicaciones con capturas.

0.2 Desarrollo

En esta parte de este documento escrito se mostrará el desarrollo de nuestro programa, el cual simula una terminal de trabajo. La cual permitirá a los usuarios trabajar con archivos, hacer uso de comandos y permitir la consulta de información en el sistema, reproducir música y jugar ahorcado. A continuación, se mostrará su funcionamiento paso por paso:

0) Terminal:

De ella se desatan todos estos sub-codigos(scripts) ya que en ella como se puede observar en el codigo primero se van a cargar los 6 scripts adicionales usando el comando "source"; acto seguido se pide al usuario su nombre, nombre del cual el sistema verificara su existencia con el comando "id", si este ya existe se le preguntara al usuario si desea crear un nuevo usuario usando el comando "adduser" si el usuario responde "S" o "s".

Continuando con la explicación, explicaremos "tarea" esta es un arreglo donde se lee la entrada del usuario segun esta sea el "case" verificará la acción a realizar o sea la tarea, mientras el usuario ingrese comandos válidos, de lo contrario se mostrará el respectivo mensaje. Cabe destacar que es ciclo se repetirá hasta que se interrumpa su ejecución del script. para esto como una de las tareas era el que el programa no pudiera ser detenido ca traves de ctrl C o ctrl Z, se hace uso del comando trap INT TSTP haciendo que la unica forma de que este sea finalizado sea a travez del comando salir, incorporado como la ultima opción en nuestro switch.

```
#!/bin/bash
source ./creditos.sh
source ./darHyF.sh
source ./infosis.sh
source ./juego.sh
source ./mp3.sh
source ./buscar.sh
source ./ayuda.sh
trap '' INT TSTP

clear
echo "Bienvenido a tu terminal"
read -p "Por favor ingresa tu usuario: " nombre
if id $nombre &> /dev/null; then
    read -s -p "Ingresa tu contraseña: " contrasena
    echo # Salto de l nea para mayor legibilidad
    if su "$nombre" -c "echo 'Contraseña correcta'" &> /dev/null; then
        echo "Bienvenido , $nombre"
```

```

else
    echo "La contraseña es incorrecta para el usuario $nombre"
fi
else
    echo "El usuario $nombre no existe"
    read -p " Deseas    crearlo?[S/N]" respuesta

    if [ "$respuesta" == "S" -o "$respuesta" = "S" ];then
        sudo adduser $nombre
    fi
fi

```

```

while true; do
printf "[%s@%s] " "$(whoami)" "$(basename "$(pwd)")"
read -a tarea
case $tarea in
    creditos)
        creditos
        true
        ;;
    fecha)
        darHyF
        true
        ;;
    infosis)
        infosis
        true
        ;;
    ahorcado)
        juego
        true
        ;;
    mp3)
        mp3
        true
        ;;
    buscar)
        buscar
        true
        ;;
    ayuda)
        ayuda
        true
        ;;
    clear)

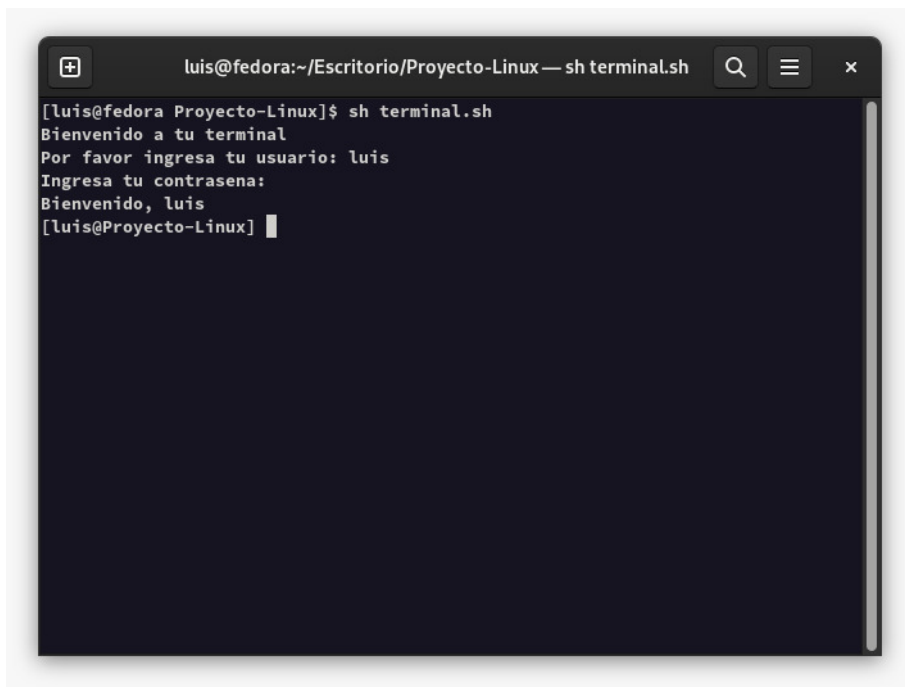
```

```

        clear
        true
        ;;
        salir)
        exit 0
        ;;

        *)
        echo Comando no valido
esac
done

```



a) Comando “ayuda”:

El comando de ayuda que se ejecuta a través de la función ayuda() , este comando proporciona información detallada sobre los comandos y programas que puede ejecutar nuestra terminal. El funcionamiento de este programa es bastante sencillo pues solo se hace uso de los comandos "echo" que permiten que en la terminal se imprima el texto deseado el cual en este caso será nuestro manual para hacer uso de la terminal.

```

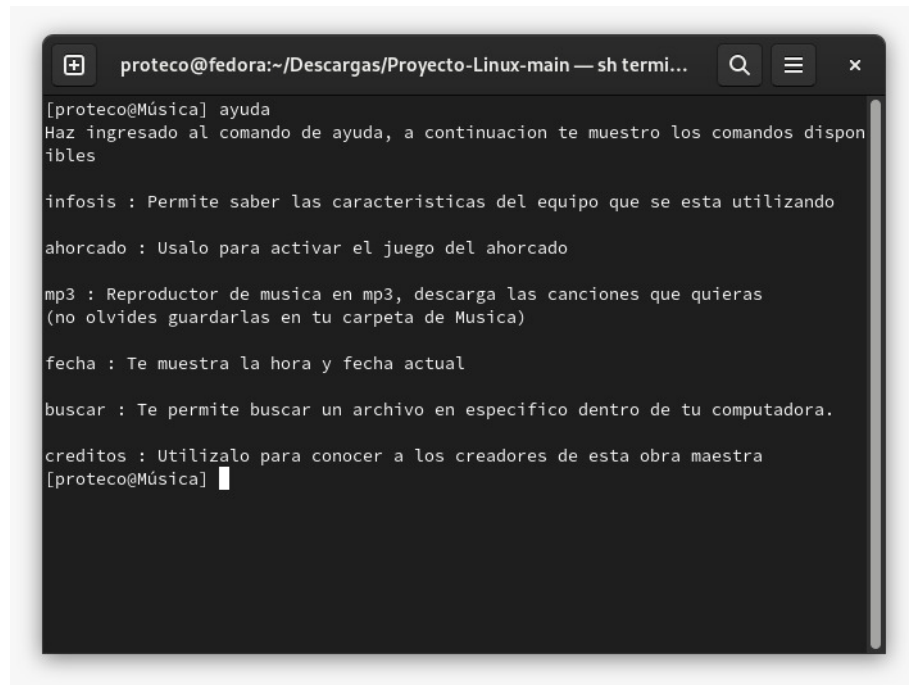
#!/bin/bash
function ayuda(){

```

```

echo Haz ingresado al comando de ayuda, a continuacion te muestro los comandos d
echo
echo infosis : Permite saber las características del equipo que se esta utilizando
echo
echo ahorcado : Usalo para activar el juego del ahorcado
echo
echo "mp3 : Reproductor de musica en mp3, descarga las canciones que quieras"
echo "(no olvides guardarlas en tu carpeta de Musica)"
echo
echo fecha : Te muestra la hora y fecha actual
echo
echo buscar : Te permite buscar un archivo en especifico dentro de tu computador
echo
echo credits : Utilizalo para conocer a los creadores de esta obra maestra
}

```



b) Comando “infosis”:

En este caso la función “infosis()”, que muestra información del sistema, como:

1. La Arquitectura del sistema, en esta se ejecuta el comando “uname -m”, el cual devuelve esta misma.
2. Versión de Sistema Operativo, el cual se ejecuta con “uname -r”.
3. Memoria RAM, dada por

```
"free -h | grep -i mem | awk '{print $2}'"
```

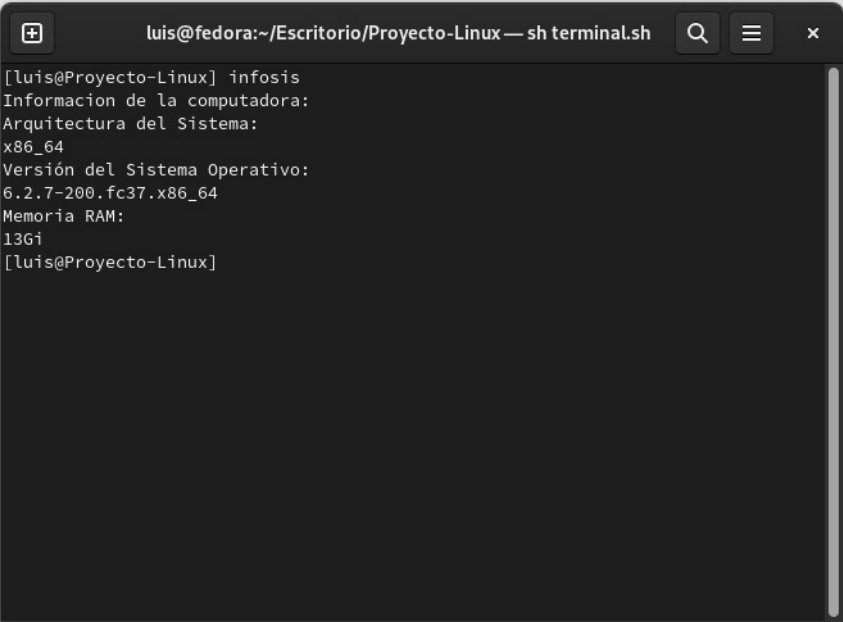
, comando que devolverá este mismo pero de una forma en la que el usuario la pueda entender; desglosando el comando "free -h" es lo que muestra información sobre el uso de la memoria, con "grep -i mem" se selecciona escoge la información de la memoria y con

```
"awk '{print $2}'"
```

se selecciona la columna que contiene el tamaño total de la memoria.

```
#!/bin/bash
```

```
function infosis() {  
    echo "Informacion de la computadora: "  
    echo "Arquitectura del Sistema:"  
    uname -m  
    echo "Versi n del Sistema Operativo:"  
    uname -r  
    echo "Memoria RAM:"  
    free -h | grep -i mem | awk '{print $2}'  
}
```



The screenshot shows a terminal window titled "luis@fedora:~/Escritorio/Proyecto-Linux — sh terminal.sh". The prompt is "[luis@Proyecto-Linux] infosis". The output of the function is as follows:

```
Informacion de la computadora:  
Arquitectura del Sistema:  
x86_64  
Versión del Sistema Operativo:  
6.2.7-200.fc37.x86_64  
Memoria RAM:  
13Gi  
[luis@Proyecto-Linux]
```

c)Comando da la fecha y la hora :

Nuestra función "darHyf()" básicamente muestra la fecha y la hora actual en un formato específico. Cuando se llama a esta función, se ejecuta el comando:

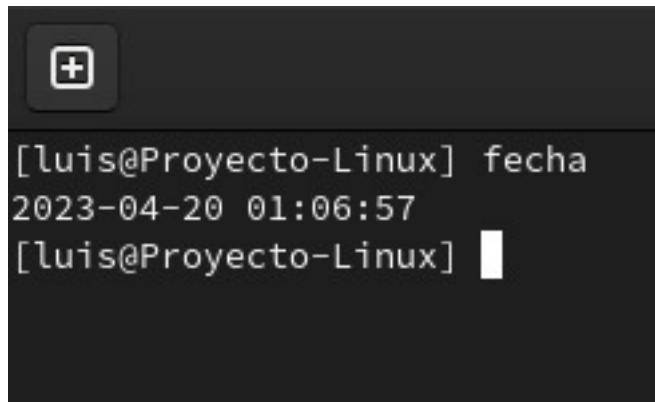
```
date '+%Y-%m-%d %H:%M:%S'
```

que devuelve la fecha y la hora actual en el siguiente formato:

Year—Mes—Dia Hora:Minutos:Segundos

```
#!/bin/bash
```

```
function darHyF() {  
    date '+%Y-%m-%d %H:%M:%S'  
}
```

A terminal window with a dark background. At the top left is a button with a white plus sign. The terminal shows the prompt [luis@Proyecto-Linux] followed by the command fecha. The output is 2023-04-20 01:06:57. Below that, the prompt [luis@Proyecto-Linux] is followed by a white cursor bar.

```
[luis@Proyecto-Linux] fecha  
2023-04-20 01:06:57  
[luis@Proyecto-Linux]
```

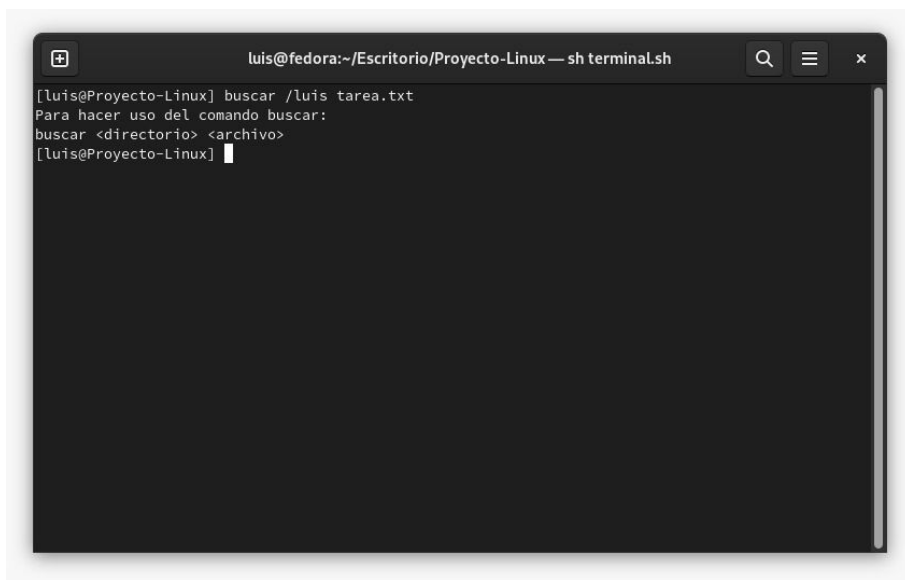
d) Comando que busca por un archivo en un directorio específico hasta recibir dos parámetros: La carpeta a buscar y el archivo que va a buscar:

Para este comando se manda a llamar a la función buscar(). Esta función utiliza el comando "find" para buscar un archivo con un nombre específico en un directorio y todos sus subdirectorios. La función toma dos argumentos: el primer argumento es el directorio donde se debe buscar el archivo y el segundo argumento es el nombre del archivo que se está buscando. Si se llama a la función sin argumentos o con un número incorrecto de argumentos, la función mostrará un mensaje explicando cómo usar correctamente el comando "buscar". Si se encuentra el archivo, la función mostrará la ruta completa del archivo en la salida estándar. Si no se encuentra el archivo, no se muestra nada en la salida estándar, pero se redirige la salida de error estándar a /dev/null para que no se muestre ningún mensaje de error.


```
#!/bin/bash

function buscar() {
    if [ $# -ne 2 ]; then
        echo "Para hacer uso del comando buscar:"
        echo "buscar <directorio> <archivo>"
        return
    fi
    find $1 -type f -name $2 2>/dev/null
}

```



e)Créditos de los programadores:

Creditos(), es solo la función que muestra nuestros respectivos créditos como equipo, cabe destacar que esto se muestra con formato. Este formato se mostrará siempre y cuando se cuente con "figlet".

```
#!/bin/bash
function creditos(){
    echo "*****"
    echo " * "
    echo " * "
    echo " * "
    echo " * $(figlet -c CREDITOS DEL MEJOR EQUIPO DE LINUX) "
    echo " * "
    echo " * $(figlet -c 'Prebes Gen.44') "
    echo " * $(figlet -c 'Elaborado por:') "
    echo " * "
    echo " * "
    echo " * $(figlet -c '*Alquicira Pe a Luis Enrique*') "
    echo " * $(figlet -c '*Gonzalez Frias Ana Paula*') "
    echo " * "
    echo " * "
    echo "*****"
}

```



f) Comando que despliega el ahorcado:

Este código implementa el juego del ahorcado en Bash. En resumen, el jugador debe adivinar una palabra secreta letra por letra antes de quedarse sin intentos. El juego utiliza varias funciones de Bash, como "echo" para imprimir mensajes en pantalla, "read" para leer la entrada del usuario, "sed" para manipular cadenas de texto, y "if" para tomar decisiones basadas en las condiciones establecidas. La estructura principal del juego es un ciclo "while" que se ejecuta hasta que el jugador gane o pierda. La función principal del código es juego(), que se encarga de llevar a cabo el juego. Inicialmente, muestra la interfaz gráfica del juego en la terminal, solicita al usuario que ingrese una letra y verifica si es correcta o no. Si la letra es correcta, se agrega a las letras adivinadas y se actualiza la palabra mostrada en la interfaz. Si la letra es incorrecta, se decrementa el número de intentos restantes. El juego continúa hasta que se adivina la palabra secreta o se agotan los intentos.

Es importante mencionar que el código utiliza varias técnicas de programación como la modularización y la validación de entradas para asegurar el correcto funcionamiento del juego y una buena experiencia de usuario.

```
#!/bin/bash
function juego(){
#variables para que funcione el juego
palabra_secreta="manzana"

letras_adivinadas=()

intentos_maximos=5
```

```

palabra_mostrada=$(echo "$palabra_secreta" | sed 's/./-/g')
intentos_restantes=$intentos_maximos

#-----
#Interfaz del juego
interfaz() {
    echo "Palabra: $palabra_mostrada"
    echo "Letras adivinadas: ${letras_adivinadas[*]}"
    echo "Intentos restantes: $intentos_restantes/$intentos_maximos"
}

verificar_victoria() {
    if [[ "$palabra_mostrada" == "$palabra_secreta" ]]; then
        echo "$palabra_secreta"
        echo "FELICIDADES GANASTE!"
        juego_terminado=true
    fi
}

#-----
#Loop y presentacion
echo "Bienvenido al juego del Ahorcado"
echo "Adivina la palabra secreta"

juego_terminado=false

while [ $juego_terminado = false ]; do
    clear
    interfaz

    echo -n "Ingresa una letra: "
    read letra

    if [[ ! "$letra" =~ ^[a-zA-Z]$ ]]; then
        echo "caracter no valido"
        continue
    fi

    if [[ "${letras_adivinadas[*]}" == *"$letra"* ]]; then
        echo "Letra repetida "
        continue
    fi

    if [[ "$palabra_secreta" == *"$letra"* ]]; then
        letras_adivinadas+=("$letra")
        palabra_mostrada=$(echo "$palabra_secreta" | sed "s/[^${letras_adivinadas[*]}]/-/g")
        verificar_victoria
    else
        intentos_restantes=$((intentos_restantes - 1))
    fi

    if [ "$intentos_restantes" -eq 0 ]; then
        echo "Perdiste :( "
        echo "La palabra secreta era: $palabra_secreta"
        juego_terminado=true
    fi
done
}

```



```
#!/bin/bash
function mp3(){
# Colores de terminal
nazul="\e[1;33m"
original="\e[0m"
nblanco="\e[1;37m"

bnegro="\e[1;40m"

# Es verifica si est instalada dicha dependencia (mpg123) si no lo est , instala
#
paquete=$(man mpg123 | grep mpg123) #man=manual de usuario/man se envia a grep
para buscar. busca info y almacena en paquete

if [ -z "$paquete" ]
then
echo -e "$nazul} No se encuentra el paquete mpg123 que ejecuta
el mp3; Desea instalarlo? s/n: ${nazul}"
read as
if [ "$as" = "s" ]
then
sudo apt install mpg123
else
echo -e "$nazul} El programa no tiene como ejecutarse :(${nazul}"
exit
terminal
fi
fi

#Codigo del reproductor
printf "$nblanco}"
echo -e "${bnegro} _____${bnegro}"
${bnegro}| ${bnegro}
${bnegro}| ${bnegro}
${bnegro}| Bienvenido al MP3 ${bnegro}
${bnegro}| ${bnegro}
${bnegro}| ${bnegro}
${bnegro}_____${bnegro}"
printf "${original}" # restaurar el color del texto al original de la terminal.
```

```

#Avisa y cambia el directorio actual del usuario
#a la carpeta "M sica" en su directorio de inicio.
#variable donde estara el directorio con los archivos mpg
echo -e "${nazul}Tus archivos mp3 deberan estar almacenados en
la carpeta M sica del usuario${namarillo}"
cd ~/M sica #cambia de directorio "" representa directorio de inicio

while [ "$Sac" != "0" ]
do
    printf "${nazul}"

    echo -e "${bnegro}
${bnegro}|      ${nblanco} Que quieres hacer?${nazul}      |${bnegro}
${bnegro}|
${bnegro}|      1) Reproducir todas las canciones      |${bnegro}
${bnegro}|      2) Reproducir una cancion                    |${bnegro}
${bnegro}|      0) Salir                                      |${bnegro}
${bnegro}|_____${bnegro}"
    printf "${nblanco}"

    read ac

    case $ac in
    0)
        echo ""
        echo -e "Chafaaa, no quieres endulzar tu odio con buena musica"
        printf "${original}"
        clear
        ;;
    1)
        printf "${namarillo}"
        echo -e "${bnegro}_____${bnegro}"
        echo -e "${bnegro}|      ${nblanco} En que modo?${nazul}      |${bnegro}"
        echo -e "${bnegro}|_____${bnegro}"
        echo -e "${bnegro}|      1) Lineal                      |${bnegro}"
        echo -e "${bnegro}|      2) shuffle                      |${bnegro}"
        echo -e "${bnegro}|_____${bnegro}"
        printf "${nblanco}"
        read op2
        case $op2 in
        1)
            clear
            printf "${namarillo}"
            echo -e "${bnegro}_____${bnegro}"
            echo -e "${bnegro}|      ${nblanco}ACCIONES${nazul}      |${bnegro}"
            echo -e "${bnegro}|_____${bnegro}"
            echo -e "${bnegro}|      S) Pausar/Reanudar                |${bnegro}"
            echo -e "${bnegro}|      F) Siguiente cancion              |${bnegro}"
            echo -e "${bnegro}|      D) Cancion anterior                |${bnegro}"
            echo -e "${bnegro}|      B) Reanudar cancion                |${bnegro}"
            echo -e "${bnegro}|      +) Subir volumen                  |${bnegro}"
            echo -e "${bnegro}|      -) Bajar volumen                  |${bnegro}"
            echo -e "${bnegro}|      U) Mute                          |${bnegro}"
            echo -e "${bnegro}|      L) Canciones disponibles          |${bnegro}"
            echo -e "${bnegro}|      Q) Parar el reproductor            |${bnegro}"
            echo -e "${bnegro}|_____${bnegro}"
            mpg123 -C -title -q *.mp3
            # -C: mp3 controlado por comandos
            en la l nea de comandos/controlan la reproducci n.
            # -title: muestra el t tulo de la canci n actual en reproducci n.
            # -q: en el reproductor de audio no se muestran mensajes extras.

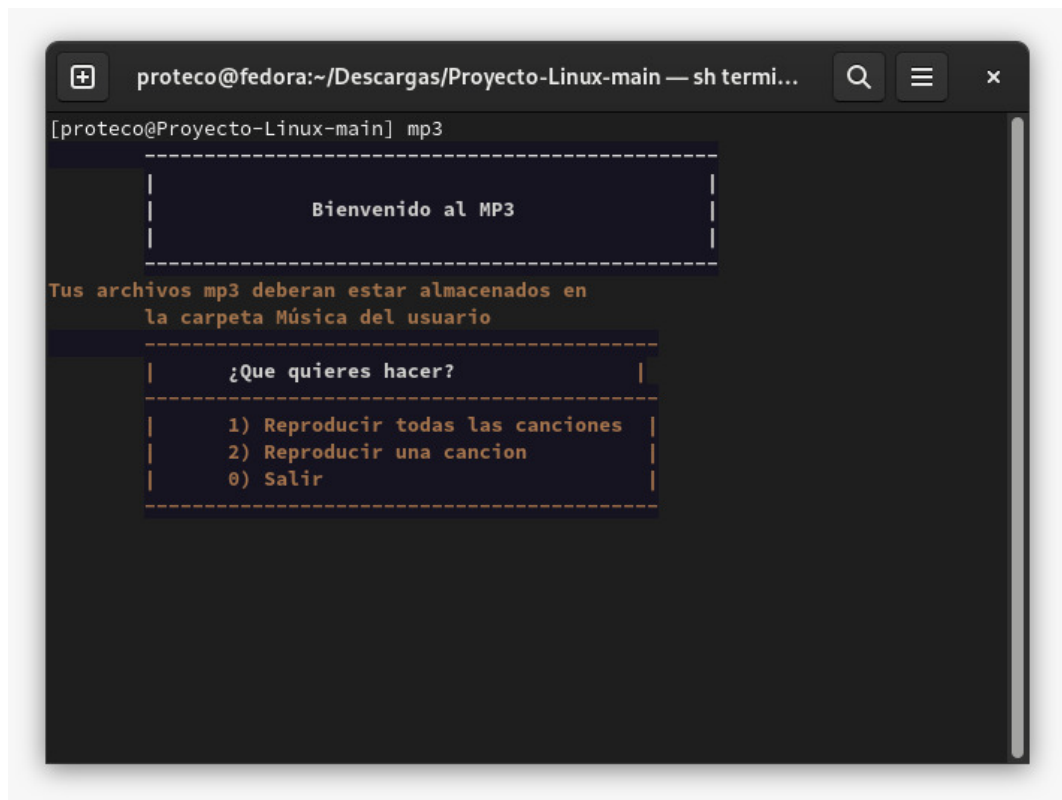
            clear
            ;;
        2)
            clear
            printf "${namarillo}"
            echo -e "${bnegro}_____${bnegro}"
            echo -e "${bnegro}|      ${nblanco}COMANDOS DISPONIBLES${nazul}      |${bnegro}"
            echo -e "${bnegro}|_____${bnegro}"
            echo -e "${bnegro}|      S) Pausar/Reanudar                |${bnegro}"
            echo -e "${bnegro}|      F) Siguiente                      |${bnegro}"

```

```

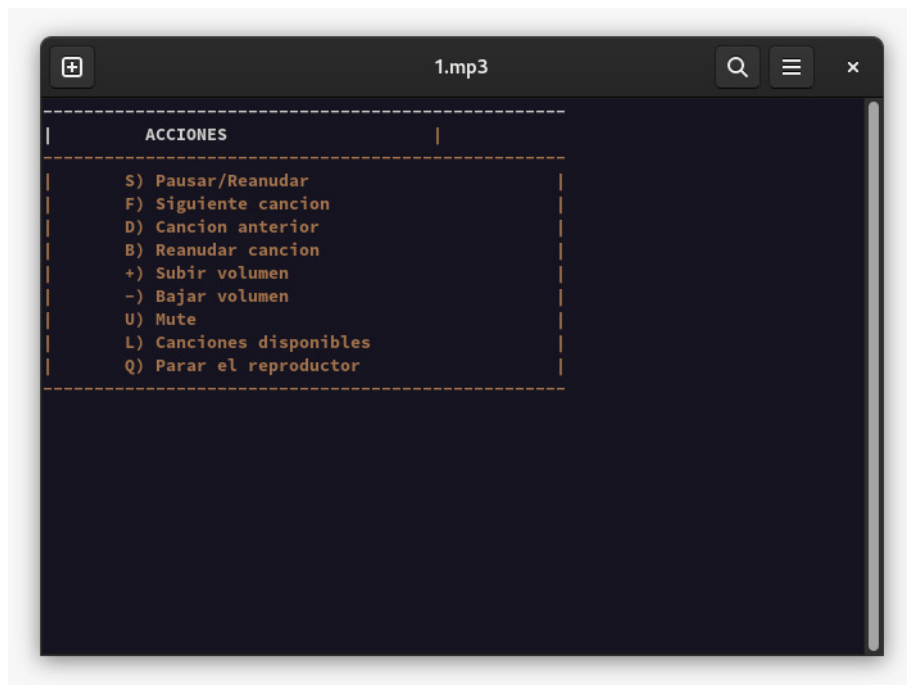
| ${bnegro}"                echo -e "${bnegro}|          D) Regresar
| ${bnegro}"                echo -e "${bnegro}|          B) Reanudar
| ${bnegro}"                echo -e "${bnegro}|          +) Subir volumen
| ${bnegro}"                echo -e "${bnegro}|          -) Bajar volumen
| ${bnegro}"                echo -e "${bnegro}|          U) Mute
| ${bnegro}"                echo -e "${bnegro}|          L) Canciones disponibles
| ${bnegro}"                echo -e "${bnegro}|          Q) Parar el reproductor
| ${bnegro}"                echo -e "${bnegro}-----${bnegro}"
mpg123 -C --title -q -z *.mp3      #-z:habilita la reproducci n en modo shuffle de los archivos MP3.
clear
ii
*)
clear
echo -e "No existe el modo"
ii
esac
ii
2)
printf "${namarillo}"
echo -e "${bnegro}-----${bnegro}"
echo -e "${bnegro}|          ${nblanco}CANCIONES DISPONIBLES${namarillo}"
| ${bnegro}"                echo -e "${bnegro}-----${bnegro}"
echo -e "${ls -l}"
echo -e "${bnegro}-----${bnegro}"
echo "Indica el nombre y formato del archivo de musica por ejemplo : Hotel.mp3:"
read cancion
(printf "${namarillo}"
echo -e "${bnegro}-----${bnegro}"
echo -e "${bnegro}|          ${nazul}COMANDOS DISPONIBLES${nazul}"
| ${bnegro}"                echo -e "${bnegro}-----${bnegro}"
| ${bnegro}"                echo -e "${bnegro}|          S) Pausar
| ${bnegro}"                echo -e "${bnegro}|          B) Reiniciar
| ${bnegro}"                echo -e "${bnegro}|          +) Subir volumen
| ${bnegro}"                echo -e "${bnegro}|          -) Bajar volumen
| ${bnegro}"                echo -e "${bnegro}|          U) MUTE
| ${bnegro}"                echo -e "${bnegro}|          Q) Parar el reproductor
| ${bnegro}"                echo -e "${bnegro}-----${bnegro}"
mpg123 -C --title -q $cancion && clear) || (clear && echo -e "Esa cancion no la topo :(")
#&&&: si el comando se ejecuta correctamente se realiza el clear
ii
*)
clear
echo -e "No es opcion valida"
ii
esac
done
}
}

```



A terminal window titled "proteco@fedora:~/Descargas/Proyecto-Linux-main — sh termi..." with search, menu, and close icons. The prompt is "[proteco@Proyecto-Linux-main] mp3". The script displays a dashed box with "Bienvenido al MP3". Below it, text in orange says "Tus archivos mp3 deberan estar almacenados en la carpeta Música del usuario". Another dashed box asks "¿Que quieres hacer?" with options: "1) Reproducir todas las canciones", "2) Reproducir una cancion", and "0) Salir".

```
proteco@fedora:~/Descargas/Proyecto-Linux-main — sh termi...
[proteco@Proyecto-Linux-main] mp3
-----
|                               |
|           Bienvenido al MP3   |
|                               |
|-----|
Tus archivos mp3 deberan estar almacenados en
la carpeta Música del usuario
-----
|           ¿Que quieres hacer?           |
|-----|
| 1) Reproducir todas las canciones |
| 2) Reproducir una cancion |
| 0) Salir |
|-----|
```



0.3 Conclusiones

Alquicira Peña Luis Enrique

Durante mi proyecto de Linux, logré cumplir los objetivos planteados por mis compañeros y por mí mismo. Gracias a este proyecto, pude poner en práctica los conocimientos adquiridos durante el curso de Linux, así como aquellos conocimientos que había obtenido durante mi carrera. Además, investigué y aprendí nuevos comandos de Shell Script para poder llevar a cabo las tareas requeridas en el proyecto. Aunque aún hay mucho por aprender sobre el sistema operativo Linux y Shell Script, este proyecto me permitió sentar unas bases sólidas para seguir aprendiendo y comprender mejor estos temas en el futuro. En resumen, este proyecto me brindó una experiencia valiosa y me permitió desarrollar habilidades útiles que puedo aplicar en otros proyectos y en mi carrera profesional.

González Frías Ana Paula

En conclusión para mí el desarrollar un programa que simula una terminal de trabajo, que permite interactuar con ella haciendo uso de archivos, comandos hasta llevar su documentación a latex fue medio estresante, aunque nos dividimos el trabajo, siento que teníamos altas y bajas, aunque mi buddy me ayudo demasiado. No obstante, fue bastante grato poder ver nuestros conocimientos aquí mostrados; creo que a grandes rasgos se lograron los objetivos. También quiero aprovechar para decir que sin lugar a duda este ha sido el curso que más me ha gustado, siento que todo lo que se realizó en este mismo me ayudo a ver mis errores y a probarme como persona.