Proyecto final GNU-LINUX

Diego Leonardo Castillo Martínez

Jimena Hernández García

abril 2023

PROTECO

Linux

Contents

Proyecto Linux
PROTECO

pro.jpeg

---

1. Introducción

El proyecto conciste en realizar una terminal que se pueda ejecutar dentro de la terminal original de linux que pueda contener todos los comandos internos y los programados por nosotros, nos permite tener un sistema de acceso que se tenga en el sistema operativo del anfitrión, se debe mostrar la ruta donde se encuentra, debe ser capaz de interpretar los comandos programados y la unica forma de salir es con el comando "salir", no es valido el Ctrl+C o Ctrl+Z, se debe impedir que estos cierren el programa, debe tener los siguientes comandos programados

- Ayuda: Te da una lista de los comandos disponibles y su descripción.
- Infosis: Permite mostrar la informacion del sistema,RAM, arquitectura, VersiónSO
- Fecha: Nos da la fecha y la hora
- Buscar: Busca un archivo en un directorio especifico
- Juego: juego programado
- mp3: reproductor de múscia con interfaz gráfica

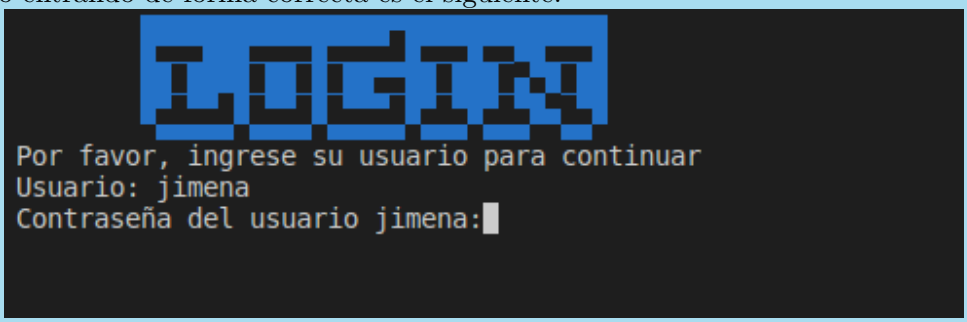
2. Objetivo:

Que el prebecario demuestre los conocimientos adquiridos durante el curso de Linux, así mismo, que ponga a prueba su capacidad de investigación, creatividad y análisis para poder cumplir con las especificaciones del proyecto.

3. Desarrollo

3.1. Login

En este comando lo que se hizo fue programar un titulo bonito, y despues hicimos que se pidiera el usuario pero el usuario se buca en el sistema si no se encuentra aparece una leyenda que dice "no existe en el sistema", si existe entonces entra al usuario y te pide su contraseña y tienes 3 intentos un ejemplo de su funcionamiento entrando de forma correcta es el siguiente:



Código

```
    #!/bin/bash
clear
rojo='\e[34m'
reset='\033[0m'
echo -e "        ${rojo}"
echo -e "         "
echo -e "        "
echo -e "         ${reset}"
echo "Por favor, ingrese su usuario para continuar"

read -p "Usuario: " usuario

if ! id -u "$usuario" >/dev/null 2>&1; then
    echo "El usuario '$usuario' no existe en el sistema"
    exit 1
fi

intentos=1
while [ $intentos -le 3 ]; do
    read -s -p "Contraseña del usuario $usuario:" contrasena

    if sudo -l -U "$usuario" -S >/dev/null 2>&1 <<< "$contrasena"; then
        . ./bienvenida.sh
        . ./terminal.sh
        exit 0
    else
        echo -e "\n\033[31m ERROR La contraseña es incorrecta\033[0m \n"
        intentos=$(expr $intentos + 1)
    fi
done

echo "Número máximo de intentos alcanzado. Adiós."
exit 1
```

3.2. terminal

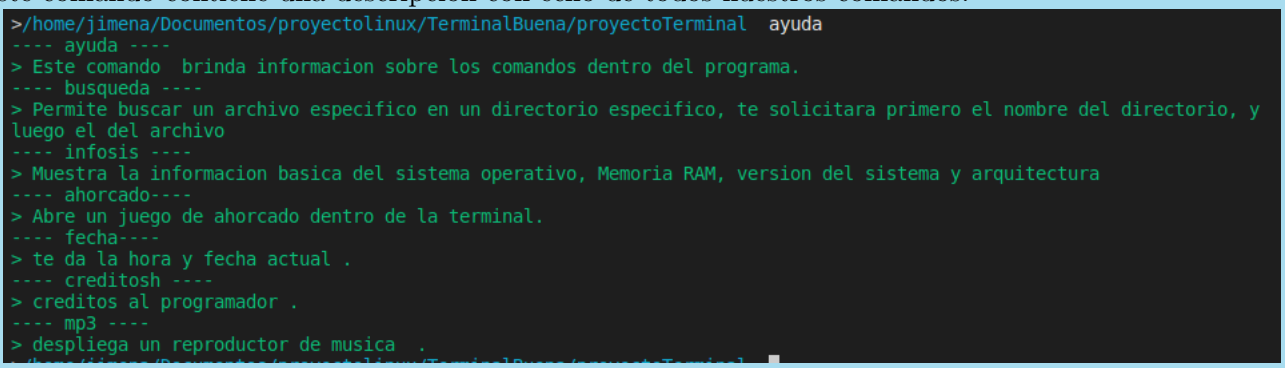
En este comando creamos una terminal y mandamos a llamar todos los comandos que programamos, hace las comparaciones para ver si los comandos estan escritos de manera correcta y los ejecuta, si el comando no existe entonces no manda ninguna orden, se especifica el comando de salir y es justo aqui donde se hacen las restriciones para evitar que la terminal se cierre con Ctrl-C y Ctrl-Z, solo se cierra con el comando salir y nos muestra un mensaje ”nos vemos pronto”.

Código

```
    #!/bin/bash
#terminal nueva
#bienvenida
trap '' INT
stty susp ^0
ruta=$(pwd)
comandos=("ayuda" "busqueda" "ahorcado" "infosis" "creditosh" "fecha" "mp3" )
comando=""
while true;
do
    terminalnueva="$(pwd)"
    printf ">""\e[0;36m$terminalnueva\e[0;37m "
    read -e -p " " comando
    for aux in "${comandos[@]}"
    do
        # Compara si la palabra dada es igual a un comando
        if [ "$comando" == "$aux" ]
        then
                . "$ruta/$comando.sh"
                comando=" "
                break

        fi
    done
    if [ "$comando" == "salir" ]; then
            comando=" "
            break
    fi
    $comando

done

echo -e "\e[34;1m  "
echo -e "\e[34;1m  "
echo -e "\e[34;1m  "
echo -e "\e[34;1m   "

```

3.3. Comando ayuda

Este comando contiene una descripcion con echo de todos nuestros comandos.



Código

```
#!/bin/bash
#ayuda
echo -e "\033[32m---- ayuda ----\n> Este comando  brinda informacion sobre los comandos dentro del programa.\033[0m"
echo -e "\033[32m---- busqueda ----\n> Permite buscar un archivo especifico en un directorio especifico, te solicitara primero el nombre del directorio, y luego el del archivo\033[0m"
echo -e "\033[32m---- infosis ----\n> Muestra la informacion basica del sistema operativo, Memoria RAM, version del sistema y arquitectura\033[0m"
echo -e "\033[32m---- ahorcado----\n> Abre un juego de ahorcado dentro de la terminal.\033[0m"
echo -e "\033[32m---- fecha----\n> te da la hora y fecha actual .\033[0m"
echo -e "\033[32m---- creditosh ----\n> creditos al programador .\033[0m"
echo -e "\033[32m---- mp3 ----\n> despliega un reproductor de musica  .\033[0m"

```

3.4. Comando infosis

En este comando se hizo una busqueda de las caracteristicas de ca l¿computadora extrayendo unicamente la RAM el UbuntuID, la arquitectura, el modelo, el numero de CPU(S), el modelo de la tarjeta grafica, el tipo de memoria.

```
>/home/jimena/Documentos/proyectolinux/TerminalBuena/proyectoTerminal  infosis
 el sistema operativo y su version es la siguiente
Ubuntu ID:
 22.04
la cpu que usted posee y su arquitectura es
Arquitectura:                    x86_64
Nombre del modelo:               AMD A6-6310 APU with AMD Radeon R4 Graphics
CPU(s):                          4
la modelo de gpu es:
[Radeon R4/R5 Graphics]
la ram instalada es la siguiente :
6.74184 GB
Correction Unknown Unknown
Memory DDR3 1600
Memory
```

Código

```
    #!/bin/bash
#infosis
echo -e "\e[0;35m el sistema operativo y su version es la siguiente \e[0m"
lsb_release -a 2>/dev/null | grep -i "Distributor ID\|Release" | awk '{print $3,$2}'
echo -e "\e[0;35mla cpu que usted posee y su arquitectura es\e[0m"
lscpu | grep -E '(Model name:|Arquitectura:)'
lscpu | grep -E '(Model name:|Nombre del modelo: )'
lscpu | grep -E '^CPU\(s\):|^Thread\(s\) per core:'
echo -e "\e[0;35mla modelo de gpu es:    \e[0m"
sudo lshw -C display | grep -i "producto" | awk '{print $3,$4,$5,$6,$7}'
echo -e "\e[0;35mla ram instalada es la siguiente :   \e[0m"
cat /proc/meminfo | grep "MemTotal" | awk '{print $2/1024/1024 " GB"}'
sudo dmidecode --type memory | awk '/Type:|Speed:/ {print $2}' | sed 'N;s/\n/ /;N;s/\n/ /'
```

3.5. Comando fecha

Este comando nos da la fecha y la hora actuales se progrmaa el numero de segundos y se formatea la fecha y la hora y depues se imprime.

```
>/home/jimena/Documentos/proyectolinux/TerminalBuena/proyectoTerminal  fecha
la fecha actual es 2023-04-22
la hora es 15:04:45
```

Código

```
    #!/bin/bash


epoch=$(printf '%(%s)T\n' -1)  # Obtener el número de segundos desde el 1 de enero de 1970
fecha=$(printf '%(%Y-%m-%d)T\n' "$epoch")  # Formatear la fecha y hora en un formato personalizado

```

```
echo "la fecha actual es $fecha"  # Imprimir la fecha y hora en la salida estándar
fecha=$(printf '%(%H:%M:%S)T\n' "$epoch")
echo "la hora es $fecha"
```

3.6. Comando búsqueda

En este comando se tiene que ingresar el directorio que vamos a buscar y el nombre del archivo empieza a buscar en el sistema en la ruta proporcionada, si no se encuentra nos da una leyenda.[el] archivo no fuen encontrado en la ruta"si es que lo encuentra .[el] archivo fue encontrado en la rutaDesea abrir el archivo(s/n)", si pulasas s se abre y si no lo puede abrir manda la leyenda de "no se abrio".

```
>/home/jimena/Documentos/proyectolinux/TerminalBuena/proyectoTerminal  busqueda
Ingrese el nombre del directorio que desea buscar:
 /home/descargas
Ingrese el nombre del archivo que desea buscar:
 redes.pdf
Buscando el directorio /home/descargas ...
El directorio /home/descargas no se encontró en el sistema.
```

Código

```
    #!/bin/bash

echo "Ingrese el nombre del directorio que desea buscar: "
read -e -p " " directorio
echo "Ingrese el nombre del archivo que desea buscar: "
read -e -p " " archivo
echo "Buscando el directorio $directorio ..."

# El comando "find" busca el directorio en todo el sistema, comenzando desde la raíz del sistema "/"
# El flag "-type d" especifica que se busquen solo directorios
# El flag "-name" especifica el nombre del directorio que se desea encontrar
resultados=$(find / -type d -name "$directorio" 2>/dev/null | head -n 1)

if [ -n "$resultados" ]
then
    echo "Se encontró el directorio $resultados   \n"
    cd "$resultados"
    archivo_encontrado=$(ls -R "$resultados" | grep "$archivo" | head -n 1)
    if [ -n "$archivo_encontrado" ]
    then
        echo "---------------- El archivo $archivo fue encontrado en  la ruta $resultados ----------- "
        echo "¿Desea abrir el archivo? (s/n)"
        read -e -p " " respuesta
        if [ "$respuesta" == "s" ]
        then
            cat "$archivo_encontrado"
        else
            echo "El archivo no se abrió."
        fi
```

```
    else
        echo "El archivo $archivo no fue encontrado en el directorio $resultados."
    fi
else
    echo "El directorio $directorio no se encontró en el sistema."
fi
```

3.7. Comando ahorcado

Código

```
    #!/bin/bash

# Lista de palabras
palabras=("perro" "gato" "elefante" "jirafa" "cebra" "pájaro" "león" "tigre" "oso" "conejo" "ratón"
↪  "ballena" "delfín" "tiburón" "serpiente" "lagarto" "araña" "mosquito" "mariposa" "gusano" "caracol"
↪  "abeja" "avispa" "pingüino" "koala" "puma" "hipopótamo" "rinoceronte" "gorila" "mono" )


```

```
# Seleccionar una palabra aleatoria
palabra=${palabras[$RANDOM % ${#palabras[@]} ]}
letras_por_adivinar=${#palabra}

# Inicializar variables
vidas=5
letras_correctas=()

function dibujar_ahorcado() {
  case $vidas in
    5)
       echo " _________     "
       echo " | /           "
       echo " |/            "
       echo " |             "
       echo " |             " ;;

    4)
       echo " _________     "
       echo " | /      |    "
       echo " |/       O    "
       echo " |             "
       echo " |             " ;;

    3)
       echo " _________     "
       echo " | /      |    "
       echo " |/       O    "
       echo " |        |    "
       echo " |          "   ;;

    2)
       echo " _________     "
       echo " | /      |    "
       echo " |/       O    "
       echo " |       /|\    "
       echo " |            " ;;

    1)
       echo " _________     "
       echo " | /      |    "
       echo " |/       O    "
       echo " |       /|\    "
       echo " |       / \   " ;;
  esac
}

# Función para imprimir la palabra con las letras adivinadas
function imprimir_palabra() {
  for letra in $(echo $palabra | grep -o .); do
    if [[ "${letras_correctas[*]}" == *"$letra"* ]]; then
```

```
      echo -n "$letra "
    else
      echo -n "_ "
    fi
  done
  echo ""
}

# Función para leer la letra del usuario
function leer_letra() {
  while true; do
    echo -n "Adivina una letra: "
    read letra
    if [[ ! "$letra" =~ ^[a-zA-Z]$ ]]; then
      echo "Por favor, introduce una letra."
    else
      break
    fi
  done
}


# Bucle principal
while true; do
  clear
  # Imprimir la palabra con las letras adivinadas
  dibujar_ahorcado
  imprimir_palabra
  longitud=${#palabra}
  # Leer la letra del usuario
  leer_letra
  # Comprobar si la letra está en la palabra
  acierto=false
  for (( i=0; i < $longitud; i++ )); do
    caracter=${palabra:i:1}
    if [[ "$caracter" == "$letra" ]]; then
        letras_correctas+=("$letra")
        letras_por_adivinar=$((--letras_por_adivinar))
        acierto=true
    fi
  done
  if [ "$acierto" == true ]; then
    echo "¡Correcto!"
  else
    vidas=$((vidas - 1))
    echo "Incorrecto. Te quedan $vidas vidas."
    echo "Presiona Enter para continuar..."
    read

  fi
  # Comprobar si el usuario ha ganado o perdido
```

```
if [[ "$letras_por_adivinar" == 0 ]]; then
    echo "¡Ganaste! La palabra era $palabra."
    echo "Presiona Enter para continuar..."
    read
    echo -e "\e[32m         "
    echo -e "        "
    echo -e "        "


    break
  elif [[ "$vidas" == 0 ]]; then



    echo -e "\e[31m                    "
    echo -e "\e[31m                    "
    echo -e "\e[31m                    "


    echo "¡Perdiste! La palabra era $palabra."
    echo "Presiona Enter para continuar..."
    read

    break
  fi
done

```

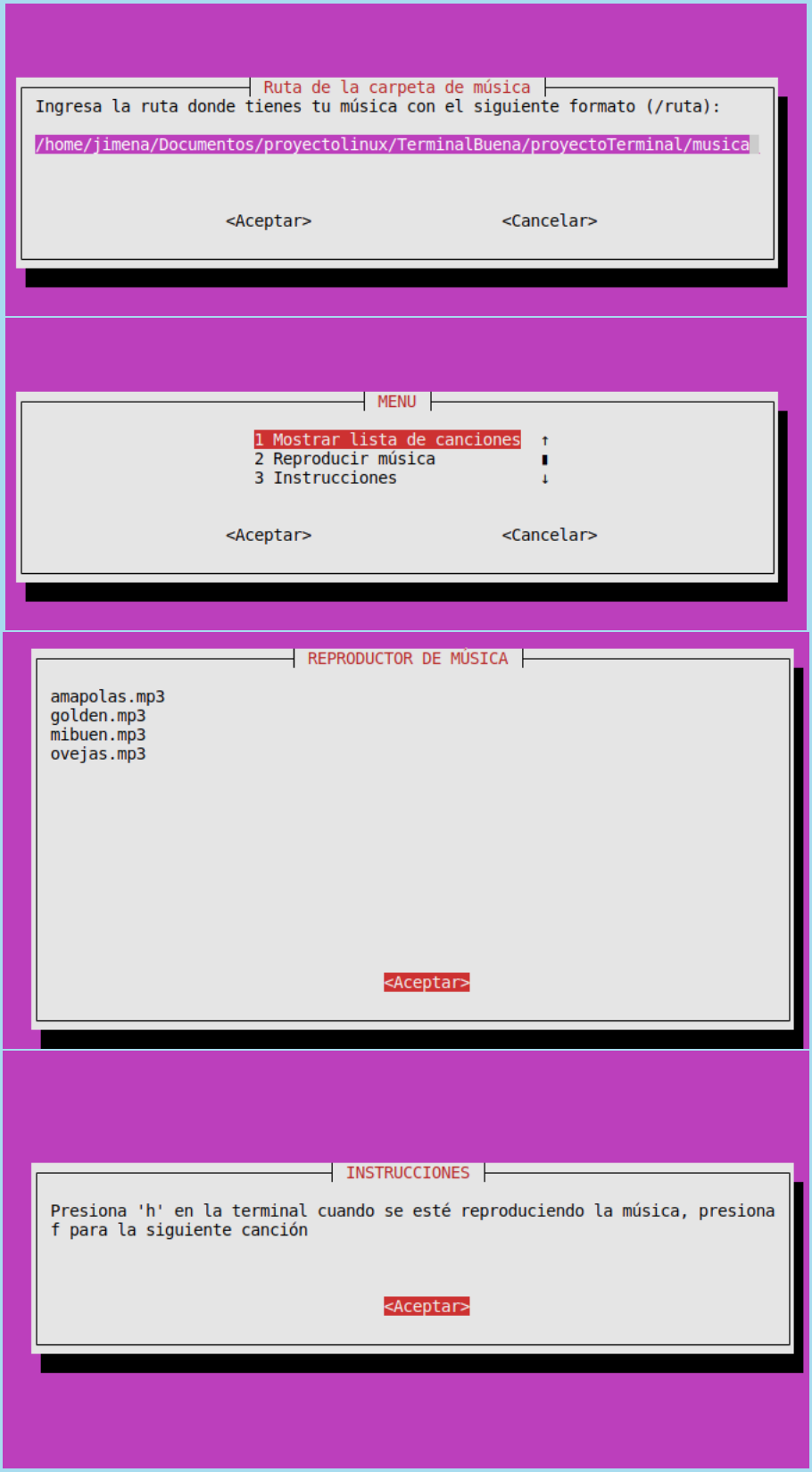
3.8. Cómando créditos

En este comando se utilizaron 2 herramientas una para las letras y la otra para el color lolcat y figlet.
Código

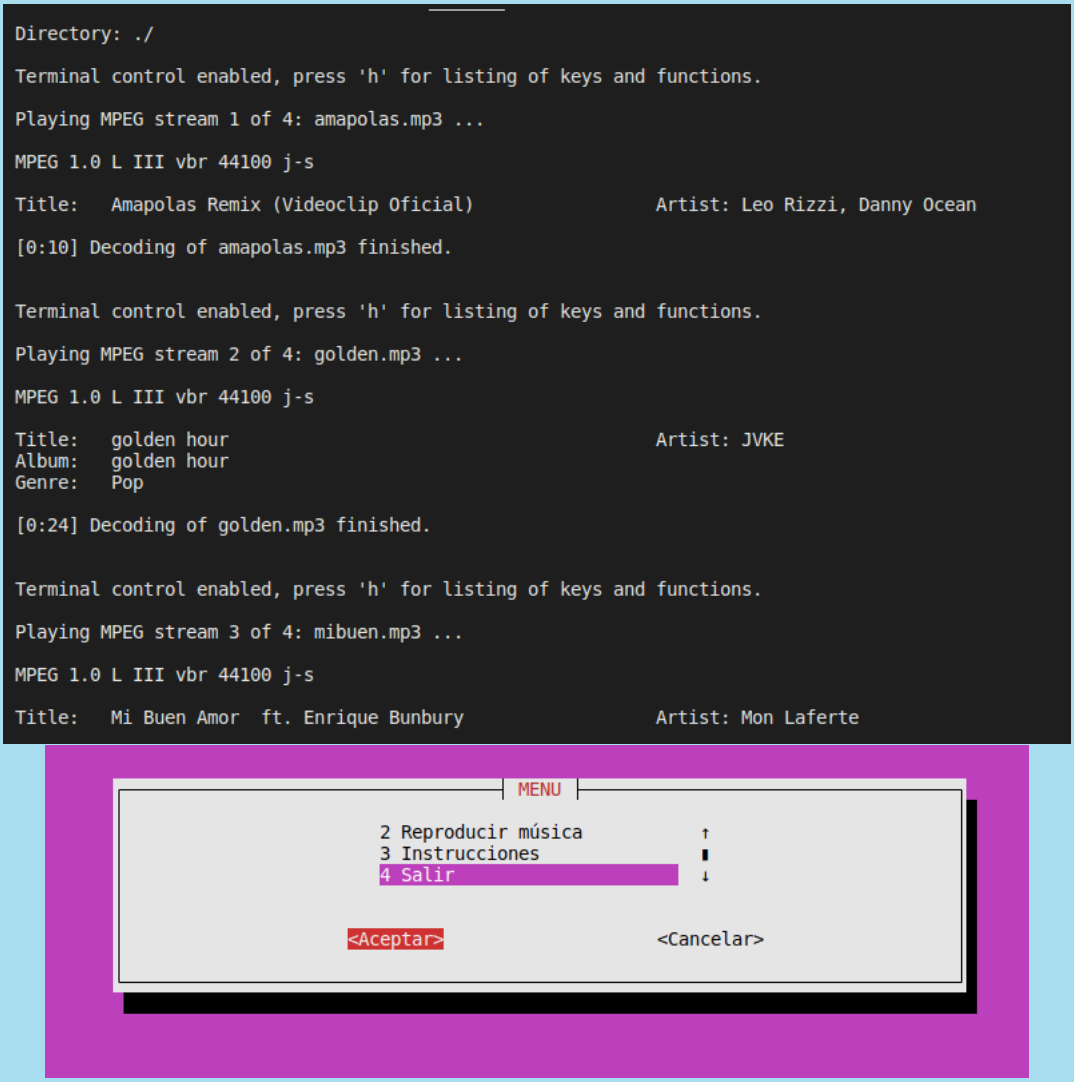
```
    #!/bin/bash

figlet -t "Creditos" | lolcat

echo "============================================================"

figlet -t "Terminal prebe" | lolcat
echo "============================================================"
figlet -f mini "Jimena Hernandez Garcia" | lolcat
figlet -f mini "Diego Leonardo Castillo Martinez" | lolcat
```

3.9. Comando mp3

Para este comando tuvimos que investigar como se usaba la herramienta mpg123, para poder instalarla en nuertro linux, tambien el como se podia reproducir de una sola carpeta para cambiar de directorios y que pudiera funcionar de forma correcta para la interfaz utilizamos una herramienta que se llama dialog que tiene diferentes opciones para mostrar cuadros de texto y seleccionesr con las flechas, te pide la ruta del archivo al principio para empezar a reproducir las canciones y tiene una sección con las instrucciones.


Ruta de la carpeta de música
Ingresa la ruta donde tienes tu música con el siguiente formato (/ruta):
/home/jimena/Documentos/proyectolinux/TerminalBuena/proyectoTerminal/musica
<Aceptar>
<Cancelar>
MENU
1 Mostrar lista de canciones
2 Reproducir música
3 Instrucciones
<Aceptar>
<Cancelar>
REPRODUCTOR DE MÚSICA
amapolas.mp3
golden.mp3
mibuen.mp3
ovejas.mp3
<Aceptar>
INSTRUCCIONES
Presiona 'h' en la terminal cuando se esté reproduciendo la música, presiona f para la siguiente canción
<Aceptar>

Código


Directory: ./
Terminal control enabled, press 'h' for listing of keys and functions.
Playing MPEG stream 1 of 4: amapolas.mp3 ...
MPEG 1.0 L III vbr 44100 j-s
Title: Amapolas Remix (Videoclip Oficial) Artist: Leo Rizzi, Danny Ocean
[0:10] Decoding of amapolas.mp3 finished.
Terminal control enabled, press 'h' for listing of keys and functions.
Playing MPEG stream 2 of 4: golden.mp3 ...
MPEG 1.0 L III vbr 44100 j-s
Title: golden hour Artist: JVKE
Album: golden hour
Genre: Pop
[0:24] Decoding of golden.mp3 finished.
Terminal control enabled, press 'h' for listing of keys and functions.
Playing MPEG stream 3 of 4: mibuen.mp3 ...
MPEG 1.0 L III vbr 44100 j-s
Title: Mi Buen Amor ft. Enrique Bunbury Artist: Mon Laferte
MENU
2 Reproducir música
3 Instrucciones
4 Salir
<Aceptar>
<Cancelar>

```
    #!/bin/bash

# Función para reproducir la música
reproducir() {
  cd "$ruta" || exit # Cambiar de directorio
  mpg123 ./*.mp3 # Reproducir la música
}

# Función para mostrar la lista de canciones
lista() {
  cd "$ruta" || exit # Cambiar de directorio
  ls *.mp3 > .lista # Crear el archivo de lista de canciones
  whiptail --title "REPRODUCTOR DE MÚSICA" --textbox .lista 20 80 # Mostrar la lista de canciones en una ventana
}

# Función para mostrar las instrucciones de uso del programa
instrucciones() {
  whiptail --title "INSTRUCCIONES" --msgbox "Presiona 'h' en la terminal cuando se esté reproduciendo la música, presiona f para la siguiente canción" 10 80
}

# Función para pedir la ruta de la carpeta con las canciones
pedir_ruta() {
  ruta=$(whiptail --title "REPRODUCTOR DE MÚSICA" --inputbox "Ingresa la ruta donde tienes tu música con el siguiente formato (/ruta):" 10 80 --title "Ruta de la carpeta de música" 3>&1 1>&2 2>&3)
  if [[ ! -d "$ruta" ]]; then # Verificar que la ruta existe
    whiptail --title "ERROR" --msgbox "La ruta que ingresaste no existe." 10 80
    pedir_ruta # Pedir de nuevo la ruta
  fi
}

# Función para verificar si el paquete mpg123 está instalado
verificar_paquete() {
  if ! command -v mpg123 &> /dev/null; then
    whiptail --title "ATENCIÓN" --yesno "Para utilizar este programa necesitas tener instalado el paquete mpg123. ¿Deseas instalarlo?" 10 80
    if [[ $? -eq 0 ]]; then
      sudo apt-get update && sudo apt-get install -y mpg123 # Instalar el paquete
    else
      exit
    fi
  fi
}

# Función principal
principal() {
  pedir_ruta # Pedir la ruta de la carpeta con las canciones
  verificar_paquete # Verificar que el paquete mpg123 está instalado
  while true; do # Mostrar el menú de opciones hasta que el usuario decida salir
```

```
47      opcion=$(whiptail --title "MENU" --menu "Elige una opción" 10 80 3 \
48               "1" "Mostrar lista de canciones" \
49               "2" "Reproducir música" \
50               "3" "Instrucciones" \
51               "4" "Salir" \
52               3>&1 1>&2 2>&3)
53      case $opcion in
54        1) lista ;;
55        2) reproducir ;;
56        3) instrucciones ;;
57        4) exit ;;
58        *) whiptail --title "ERROR" --msgbox "Opción inválida." 10 80 ;;
59      esac
60    done
61  }
62
63  principal
```

4. Conclusiones por buddy

Jimena: Me parecio un buen proyecto muy dinámico si puso a prueba mis conocimientos de shellscript fue mucha investigación lo cual me gusto pero me parecio muy complejo al final creo que se pudo hacer con tiempo y dedicación y me gusto mucho el resultado, aprendí muchisimas cosas más que en el curso por si mísmo
leonardo: Considero que este proyecto me ayudó a mejorar mi habilidad con el uso de comandos y shellscript, ya que me forzó a buscar distintas soluciones a un problema y tuve que investigar distintas fuentes para comprender los distintos comandos utilizados en este proyecto. En general, me gustó mucho este proyecto y me permitió mejorar mi nivel de programación en shell. Además, es uno de los mejores proyectos que he hecho y al que más tiempo le he dedicado..