UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROTECSA

Proyecto Final

Terminal Linux

PRESENTA

Jimenez Cruz Gustavo Emir

Hernàndez Hernàndez Deissy Jovita

1ra Generacion PROTECSA

Índice

1. Introducción

En este proyecto se desarrolló un programa en **Bash** que simula una terminal de trabajo personalizada y funcional. Esta terminal incluye funcionalidades como la autenticación del usuario, interacción mediante comandos tanto del sistema como personalizados, reproducción de música, un juego interactivo. La idea principal es ofrecer una terminal no solo funcional, sino también agradable para el usuario, integrando elementos visuales y funcionalidades completas. El objetivo principal fue profundizar en el uso de Bash scripting, así como integrar conocimientos adquiridos en temas de redes, programación y sistemas operativos.

2. Objetivos

- Simular una terminal funcional en Bash.
- Implementar un sistema de autenticación básico.
- Crear comandos personalizados.
- Desarrollar un mini juego textual.
- Agregar un reproductor musical.

3. Desarrollo del Proyecto

El proyecto se organiza dentro de una carpeta principal llamada `proyecto_terminal`, que contiene los siguientes elementos:

3.1. Autenticación de Usuario

El sistema de autenticación se implementó en el script **terminal.sh** que simula una interfaz de inicio de sesión, donde el usuario debe ingresar un nombre y contraseña para acceder. A continuación, se describen las funcionalidades principales que se integraron:

- **Arte ASCII**: Al iniciar el script, se despliega un título decorativo en caracteres ASCII, generado desde la herramienta en línea patorjk.com. Esto mejora la experiencia visual y da una identidad al sistema.
- **Ventana de Bienvenida**: Se simula una caja de diálogo utilizando caracteres como `ANSI`, mostrando el mensaje de bienvenida centrado y con colores mediante secuencias ANSI.
- **Entrada de Usuario y Contraseña**: Se solicita al usuario ingresar su nombre y contraseña. La contraseña se oculta en pantalla utilizando la opción `-s` del comando `read`, lo cual proporciona mayor privacidad.
- **Verificación de Credenciales**: La validación se realiza utilizando el comando `su -c .exit"` para intentar autenticar al usuario con la contraseña proporcionada. Si el intento es exitoso, se permite el acceso; de lo contrario, se muestra un mensaje de error y se detiene la ejecución del script.

- **Inicio del Entorno**: Si las credenciales son válidas, se muestra un mensaje de ïniciando...", se limpia la pantalla con `clear`, y finalmente se ejecuta el script principal del proyecto (`shell.sh`) usando el comando `exec`, lo cual reemplaza el proceso actual y da una experiencia más fluida.
- **Uso de Colores**: Para mejorar la estética, se usaron códigos ANSI en variables de color (como `\033[0;31m` para rojo, `0;32m` para verde, etc.), lo que permite resaltar mensajes clave en la terminal.

Listing 1: Script para mostrar el inicio de sesion dentro de la terminal

```
#!/bin/bash

# Colores estandar
R='\033[1;31m'   # Rojo brillante
G='\033[1;32m'   # Verde brillante
Y='\033[1;33m'   # Amarillo brillante
B='\033[1;34m'   # Azul brillante
M='\033[1;35m'   # Magenta brillante
C='\033[1;36m'   # Cian brillante
W='\033[1;37m'   # Blanco brillante (gris claro)


# Titulo generado en https://patorjk.com/software/taag/
echo -e "${B0}"



        echo -e "${C}\t\t\
                                                t
            "
        echo -e "${C}\t\t\ t  ␣␣${W}␣␣␣␣B␣I␣E␣N␣V␣E␣N␣I␣D␣O␣␣␣␣␣␣␣␣${C}    "
        echo -e "${C}\t\t\
                                                t
            "


# Entrada de usuario y contrase a

        user="$C\t\t\t␣Usuario:␣$W␣"
        pass="$C\t\t\t␣Contrase a:$W␣"

        read -p "$(echo␣-e␣"$user")" user
        read -s -p "$(echo␣-e␣"$pass")" pass


# Verifica usuario con el sistema
        if su -c "exit" "$user" <<< "$pass" &>/dev/null; then
                echo -e "${G}\n\n\t\t\t\t␣ACCESO␣AUTORIZADO"
        else
                echo -e "${R}\n\n\t\t\t␣USUARIO␣O␣CONTRASE A␣INCORRECTOS"
        exit 1
        fi

# Ejecutar la terminal personalizada
        echo -e "${G}\n\t\t\t␣I␣N␣I␣C␣I␣A␣N␣D␣O...${W}"
sleep 3
clear
exec ./shell.sh
```

3.2. Comandos Personalizados

Dentro de la terminal simulada, se implementaron varios comandos, los cuales fueron programados como un script independiente y llamado desde un ciclo principal en `shell.sh`.

3.3. Comando ayuda

`ayuda` - Este comando es una guia basica para identificar los comandos permitidos, ademas de darnos una breve descripccion de estos. Se decidio desarrollar en el script ayuda.sh, ya que le asignamos colores y no queriamos que fuera solo un texto plano.

Listing 2: Script para mostrar los comandos validos dentro de la terminal

```
1
2  #!/bin/bash
3  clear
4  # Colores estandar
5  M='\033[1;35m'   # Magenta brillante
6  C='\033[1;36m'   # Cian brillante
7  W='\033[1;37m'   # Blanco brillante (gris claro)
8
9  echo -e "${M}"
10
11 echo -e "${M}\n\t\tC␣O␣M␣A␣N␣D␣O␣S␣␣V␣A␣L␣I␣D␣O␣S"
12 echo -e "${M
       }===-===-===-===-===-===-===-===-===-===-===-===-===-===-===-===-===-"
13 echo -e "␣␣${C}ayuda${W}Tu␣guia␣rapida␣por␣este␣universo␣de␣comandos"
14 echo -e "␣␣${C}infosis${W} Curioso ␣sobre␣tu␣PC?␣Descubre␣sus␣secretos␣aqui"
15 echo -e "␣␣${C}tiempo${W} Perdido ␣en␣el␣tiempo?␣Averigua␣el␣dia␣y␣hora"
16 echo -e "␣␣${C}buscar${W} Donde ␣est ␣ese␣archivo?␣Yo␣te␣ayudo␣a␣encontrarlo"
17 echo -e "␣␣${C}creditos${W}Conoce␣a␣los␣genios␣detras␣de␣esta␣terminal"
18 echo -e "␣␣${C}jugar${W} Aburrido ?␣ Diviertete ␣con␣nuestro␣minijuego!"
19 echo -e "␣␣${C}musica${W} Necesitas ␣ritmo?␣Pon␣tus␣canciones␣favoritas"
20 echo -e "␣␣${C}salir${W} Ya ␣te␣vas?␣Hasta␣la␣proxima␣aventura..."
21 echo -e "${M
       }===-===-===-===-===-===-===-===-===-===-===-===-===-===-===-===-===-${W}"
```

3.4. Comando infosis

`infosis` - Este comando muestra información del sistema operativo. incluye la versión de Linux, arquitectura y memoria. Es útil para tener un panorama general del entorno de ejecución. Esta informacion esta dada de la siguiente forma:

- **L**a cantidad total de memoria RAM en megabytes.
- **L**a arquitectura del sistema (32 o 64 bits).
- **E**l modelo del procesador.
- **E**l nombre completo del sistema operativo (por ejemplo, Ubuntu 24.04).
- **L**a base sobre la que está construido el sistema (por ejemplo, Debian).

El script toma esta información directamente de archivos del sistema como /proc/meminfo, /proc/cpuinfo y /etc/os-release, y la presenta de forma organizada para facilitar la comprensión del entorno en el que corre la terminal personalizada.

Listing 3: Script para mostrar información del sistema

```
#!/bin/bash
clear

M='\033[1;35m'   # Magenta brillante
C='\033[1;36m'   # Cian brillante
W='\033[1;37m'   # Blanco brillante (gris claro)

echo -e "${M
    }===-===-===-===-===-===-===-===-===-===-===-===-===-===-===-===-===-===-===-===${W}"

# Memoria RAM (en KB) desde /proc/meminfo
mem_total=$(grep MemTotal /proc/meminfo | awk '{print␣$2}')
mem_total_mb=$((mem_total / 1024))

echo -e "${C}\n\t\t␣R␣A␣M␣␣T␣O␣T␣A␣L:␣${W}␣$mem_total_mb␣MB"

# Arquitectura del sistema
echo -e "${C}\tA␣R␣Q␣U␣I␣T␣E␣C␣T␣U␣R␣A:␣${W}␣$(getconf␣LONG_BIT)␣bits"

# Modelo de procesador
echo -e "${C}\tP␣R␣O␣C␣E␣S␣A␣D␣O␣R:␣${W}␣$(egrep␣"model name"␣/proc/cpuinfo␣|␣
    uniq␣|␣cut␣-d␣":"␣-f2)"

echo -e "${C}\n\t\tS␣I␣S␣T␣E␣M␣A␣␣O␣P␣E␣R␣A␣T␣I␣V␣O${W}"

# Nombre del sistema operativo
nombre_sisop=$(grep "PRETTY_NAME" /etc/os-release | grep -o '".*"' | sed 's
    /"//g')
echo -e "${W}\n\t\t\t$nombre_sisop"

# Tipo de sistema (basado en Debian, etc.)
basado_sisop=$(grep "ID_LIKE" /etc/os-release | grep -o '=.*' | sed 's/=//')
echo -e "${C}\t\t␣B␣A␣S␣A␣D␣O␣␣E␣N:␣${W}␣$basado_sisop"
```

3.5. Comando tiempo

`tiempo` - Muestra la hora y fecha actual del sistema. Es útil para registrar eventos o simplemente tener control del tiempo durante el uso de la terminal.

Para obtener los datos, el script accede directamente al archivo /proc/driver/rtc, extrayendo la fecha (rtc date) y la hora (rtc time) mediante comandos cat, grep y awk. De esta forma, se garantiza que la información mostrada es precisa y en tiempo real.

Listing 4: Script para mostrar fecha y hora del sistema

```
#!/bin/bash
clear
# Colores
M='\033[1;35m'  # Magenta
C='\033[1;36m'  # Cian
G='\033[1;32m'  # Verde
R='\033[1;31m'  # Rojo
W='\033[0m'     # Reset

# Obtener fecha y hora desde /proc/driver/rtc
fecha=$(cat /proc/driver/rtc | grep rtc_date | awk '{print␣$3}')
```

```
hora=$(cat /proc/driver/rtc | grep rtc_time | awk '{print $3}')

echo -e "${C}\n\t\t============================="
echo -e "\t\t         Fecha y Hora"
echo -e "\t\t============================="
echo -e "${M}\t\t Fecha actual:${W} $fecha"
echo -e "${M}\t\t Hora actual: ${W} $hora"
```

3.6. Comando buscar

`buscar` - Nos permite buscar un documento, dando unicamente dos parametros, la carpeta y nombte del archivo. Permite buscar archivos en una ruta específica usando palabras clave. Es útil para localizar información o scripts dentro de una estructura de carpetas.

Al ejecutarlo, se limpia la pantalla y se muestra un encabezado en arte ASCII, como los comandos anteriores, seguido de mensajes que guían al usuario para introducir el nombre de una carpeta y el nombre de un archivo.

El script primero busca la carpeta indicada dentro del directorio personal del usuario /home/ USER) Si la carpeta existe, procede a buscar dentro de ella el archivo especificado. Si el archivo se encuentra, se muestra la ruta completa donde fue localizado. En caso contrario, se informa al usuario si la carpeta no fue hallada o si el archivo no se encuentra dentro de ella.

Listing 5: Script para Buscar algun archivo en un directorio

```

# Pedir nombre de la carpeta
echo -e "${C}\n\t\t Ingresa el nombre de la carpeta donde buscar${W}"
read -p "Carpeta: " carpeta

# Pedir nombre del archivo
echo -e "${C}\n\t\t Ingresa el nombre del archivo que deseas buscar${W}"
read -p "Archivo: " archivo

# Buscar la carpeta por nombre dentro de /home/$USER
carpeta=$(find "/home/$USER" -type d -name "$carpeta" 2>/dev/null | head -n 1)

# Verificar si la carpeta fue encontrada
if [[ -d "$carpeta" ]]; then
    # Buscar el archivo dentro de la carpeta encontrada
    archivo=$(find "$carpeta" -type f -name "$archivo" 2>/dev/null | head -n
        1)

    # Verificar si el archivo fue encontrado
    if [[ -f "$archivo" ]]; then

        echo -e "\n\t\t\t ${G}Archivo encontrado en:${W}"
        echo -e "\t\t$archivo\n"
    else

        echo -e "\n\t\t\t${Y}No se encontro el archivo en la carpeta.${W}"
    fi
else

    echo -e "\n\t\t\t${R}La carpeta no existe en el sistema.${W}"
fi
```

3.7. Comando creditos

`creditos` - Nos muestra a los desarrolladores de la terminal.

Listing 6: Script para mostrar a los desarrolladores

```
1
2  #echo -e "   "
3
4  echo -e "${C}"
5  echo -e "\t\t␣␣␣␣␣␣␣.--------------------------."
6  echo -e "\t\t␣␣␣␣␣␣␣|␣.------------------.␣|==|"
7  echo -e "\t\t␣␣␣␣␣␣␣|␣|␣................␣|␣|␣␣|"
8  echo -e "\t\t␣␣␣␣␣␣␣|␣|␣::::PROTECSA[]:::␣|␣|␣␣|"
9  echo -e "\t\t␣␣␣␣␣␣␣|␣|␣:::::::::::::::::␣|␣|␣␣|"
10 echo -e "\t\t␣␣␣␣␣␣␣|␣|␣::::${W}Dey${C}::::::::::␣|␣|␣␣|"
11 echo -e "\t\t␣␣␣␣␣␣␣|␣|␣:::::::::::::::::␣|␣|␣␣|"
12 echo -e "\t\t␣␣␣␣␣␣␣|␣|␣::::::${W}gossbosther${C}␣|␣|␣␣|"
13 echo -e "\t\t␣␣␣␣␣␣␣|␣|␣:::::::::::::::::␣|␣|␣,|"
14 echo -e "\t\t␣␣␣␣␣␣␣|␣!--------------------!␣|(c|"
15 echo -e "\t\t␣␣␣␣␣␣␣!----------------------!__!"
16 echo -e "\t\t␣␣␣␣␣␣/␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣\␣"
17 echo -e "\t\t␣␣␣␣␣/␣␣[][][][][][][][][][][][][][][][]␣␣\␣"
18 echo -e "\t\t␣␣␣␣/␣␣[][][][][][][][][][][][][][][][]␣␣\␣"
19 echo -e "\t\t␣␣␣(␣␣[][][][][][____________][][][][]␣␣)␣"
20 echo -e "\t\t␣␣␣␣\␣------------------------------␣/␣"
21 echo -e "\t\t␣␣␣␣␣\----------------------------/␣"
22
23 echo -e "${M}\n\t.:*~*:._.:*~*:._.:*~*:._.:*~*:._.:*~*:._.:*~*:._.:*~*:._.:*~*:.
       .:*~*:."
24 echo -e "${C}\t\t\\t␣␣␣␣Desarrollado␣por:${W}"
25 echo -e "\n\t␣D␣E␣I␣S␣S␣Y␣␣J␣O␣V␣I␣T␣A␣␣H␣E␣R␣N␣A␣N␣D␣E␣Z␣␣H␣E␣R␣N␣A␣N␣D␣E␣Z"
26 echo -e "\n\t\t␣G␣U␣S␣T␣A␣V␣O␣␣E␣M␣I␣R␣␣J␣I␣M␣E␣N␣E␣Z␣␣C␣R␣U␣Z␣"
27 echo -e "${M}\n\t.:*~*:._.:*~*:._.:*~*:._.:*~*:._.:*~*:._.:*~*:._.:*~*:._.:*~*:.
       .:*~*:.${W}\n"
```

3.8. comando jugar

`jugar` - Ejecuta un mini juego textual interactivo dentro de la terminal. Se implementa el clásico juego del gato en la terminal, ofreciendo una experiencia interactiva entre dos jugadores. El juego se representa con un tablero de 3x3 mediante un arreglo de nueve posiciones que se actualiza a medida que los jugadores ingresan sus jugadas.

Al iniciar, se limpia la pantalla y se muestra un título en arte ASCII junto con un tablero vacío. Cada turno, el script solicita al jugador actual (X u O) que ingrese un número del 1 al 9 para marcar su jugada. El programa valida la entrada y evita sobreescrituras en casillas ocupadas.

Tras cada jugada, el sistema verifica si existe una combinación ganadora (horizontal, vertical o diagonal). Si un jugador gana, se muestra una animación en ASCII y un mensaje colorido celebrando la victoria. En caso de empate (cuando se llenan las 9 casillas sin ganador), se imprime el mensaje de empate.

Listing 7: Script para iniciar el juego de Gato

```
1
```

```
# Inicializar tablero vacio
tablero=("␣" "␣" "␣" "␣" "␣" "␣" "␣" "␣" "␣")
turno="X"
jugadas=0

imprimir_tablero() {
    clear
echo -e "${G}"

        #printf "\n\t\t    /\_/\  \n"
        #printf "\t\t   ( o.o ) \n"
        #printf "\t\t    > ^ <  \n"

    echo -e "${Y}"
    printf "\t\t␣␣␣␣${tablero[0]}␣|␣${tablero[1]}␣|␣${tablero[2]}␣\n"
    printf "\t\t␣␣␣---+---+---␣\n"
    printf "\t\t␣␣␣␣${tablero[3]}␣|␣${tablero[4]}␣|␣${tablero[5]}␣\n"
    printf "\t\t␣␣␣---+---+---\n"
    printf "\t\t␣␣␣␣${tablero[6]}␣|␣${tablero[7]}␣|␣${tablero[8]}␣\n"
    echo -e "${W}"
}

verificar_ganador() {
    local combinaciones=(
        "0␣1␣2"
        "3␣4␣5"
        "6␣7␣8"
        "0␣3␣6"
        "1␣4␣7"
        "2␣5␣8"
        "0␣4␣8"
        "2␣4␣6"
    )

    for comb in "${combinaciones[@]}"; do
        set -- $comb
        if [[ "${tablero[$1]}" == "$turno" && "${tablero[$2]}" == "$turno" &&
            "${tablero[$3]}" == "$turno" ]]; then
            return 0
        fi
    done

    return 1
}

jugar_turno() {
    local casilla
    while true; do
        read -p "␣␣␣␣␣␣␣␣Jugador␣$turno,␣elige␣una␣casilla␣(1-9):␣" casilla
        if [[ "$casilla" =~ ^[1-9]$ && "${tablero[$((casilla␣-␣1))]}" == "␣"
            ]]; then
            tablero[$((casilla - 1))]=$turno
            ((jugadas++))
            break
        else
            echo -e  "${R}\t\tCasilla␣invalida␣o␣ya␣ocupada.␣Intenta␣de␣nuevo.
                ${W}"
        fi
    done
```

```
}

# Juego principal
while true; do
    imprimir_tablero
    jugar_turno
    imprimir_tablero

    if verificar_ganador; then

        echo -e "${W}"
printf "\t␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣/)␣␣␣␣\n"
printf "\t␣␣␣␣␣␣␣␣␣␣␣/\___/\␣((␣␣␣␣␣\n"
printf "\t␣␣␣␣␣␣␣␣␣␣␣\␣@_@'/␣␣))␣␣␣␣\n"
printf "\t␣␣␣␣␣␣␣␣␣␣␣{_:Y:.}_//␣␣␣␣␣\n"
printf "\t----------{_}^-'{_}----------\n"

        echo -e "${G}\tJUGADOR␣${Y}$turno␣${G}␣ha␣${Y}␣G␣A␣N␣A␣D␣O${G}${W}␣"
        break
    elif [[ $jugadas -eq 9 ]]; then

        echo -e "${W}"
printf "\t\t␣␣␣␣␣␣/\_____/\␣␣␣␣␣\n"
printf "\t\t␣␣␣␣␣/␣␣º␣␣␣º␣␣\␣␣␣␣\n"
printf "\t\t␣␣␣␣(␣==␣␣^␣␣==␣)␣␣␣\n"
printf "\t\t␣␣␣␣␣)␣␣␣␣␣␣␣␣␣(␣␣␣␣\n"
printf "\t\t␣␣␣␣(␣␣␣␣␣␣␣␣␣␣)␣␣␣\n"
printf "\t\t␣␣␣(␣(␣␣)␣␣␣(␣␣)␣)␣␣\n"
printf "\t\t␣␣(__(__)___(__)__)␣\n"

       echo -e "${C}\n\t\tE␣M␣P␣A␣T␣E!${W}"
        break
    fi
    # Cambiar de turno
    [[ "$turno" == "X" ]] && turno="O" || turno="X"
done
```

3.9. comando musica

`musica` - Inicia el reproductor de música.

El comando musica fue implementado como un reproductor de MP3 en la terminal usando Bash y el programa mpg123. El script permite al usuario elegir entre dos modos: reproducir todas las canciones de una carpeta de forma aleatoria o seleccionar una canción específica para escuchar. Se incluye un menú interactivo con diseño colorido y controles que funcionan en tiempo real durante la reproducción, como pausar, cambiar de canción o ajustar el volumen. El script también verifica automáticamente si mpg123 está instalado y ofrece instalarlo en caso de que no lo esté.

Listing 8: Script para iniciar el reproductor de musica

```
# Carpeta de musica
MUSIC_DIR="$HOME/proyecto_terminal/musicalista"

# Verificacion de mpg123
if ! command -v mpg123 &> /dev/null; then
    echo -e "\n${R}[!]␣mpg123␣no␣esta␣instalado."
```

```
    echo -ne "${G} Deseas instalarlo? (s/n):"
    read instalar
    if [[ "$instalar" == "s" ]]; then
        sudo apt update && sudo apt install -y mpg123
    else
        echo -e "${R}No se puede continuar sin mpg123. Saliendo..."
        exit 1
    fi
fi

# Men  principal
while true; do
    clear
    echo -e "${C}"

    echo -e "${W}"
    echo -e "${Y}\t\t[1]${W} Reproducir canciones aleatorias"
    echo -e "${Y}\t\t[2]${W} Elegir una cancion especifica"
    echo -e "${Y}\t\t[3]${W} Salir"
    echo ""
    echo -ne "${G}\t\t E L I G E    U N A O P C I O N: ${W}"
    read opcion

        if [[ "$opcion" == "1" ]]; then
       clear
       echo -e "${C}\n\t\t Reproduciendo musica aleatoria desde\n\t\t ${
          MUSIC_DIR}"

 #clear
        #       canciones=("$MUSIC_DIR"/*.mp3)
        #       i=1
        #for cancion in "${canciones[@]}"; do
         #   ((i++))
        #done

        #echo -e "${C}\n\t\t\t Reproduciendo:\n\t\t\t\t $(basename "$cancion"
           .mp3)"

        echo -e "${C}\n\t\t\t        <<            >>       "
        echo -e "${Y}"

        printf "\t\t Controles durante la reproduccion  \n"
        printf "\t\t s = Pausa / Play                   \n"
        printf "\t\t d = Cancion anterior               \n"
        printf "\t\t f = Cancion siguiente              \n"
        printf "\t\t u = Silenciar                      \n"
        printf "\t\t + = Subir volumen                  \n"
        printf "\t\t - = Bajar volumen                  \n"
        printf "\t\t l = Muestra lista de canciones     \n"
        printf "\t\t q = Salir                          \n"

        echo -e "${W}"
        mpg123 -C --title -q "$MUSIC_DIR"/*.mp3
        echo ""
#        read -p "           Presiona Enter para volver al menu..."

    elif [[ "$opcion" == "2" ]]; then
        clear
        echo -e "${C}\n\t\tL I S TA    C A N C I O N E S :"
```

```
        echo -e  "
            ===_===_===_===_===_===_===_===_===_===_===_===_===_===_===_===_===_===
            ${W}"
        canciones=("$MUSIC_DIR"/*.mp3)
        i=1
        for cancion in "${canciones[@]}"; do
            echo "            [$i] $(basename "$cancion")"
            ((i++))
        done
        echo ""
        echo -ne "${G}\t\t\tSelecciona el numero de cancion: ${W}"
        read seleccion

        if [[ "$seleccion" =~ ^[0-9]+$ ]] && (( seleccion >= 1 && seleccion <
            i )); then
            cancion="${canciones[$((seleccion-1))]}"
 clear
            echo -e "${C}\n\t\t Reproduciendo:\n\n\t\t\t$(basename "$cancion")
                "
            echo -e "${C}\n\t\t\t       <<            >>       "


           echo -e "${Y}"

            printf "\t\t Controles durante la reproduccion  \n"

            printf "\t\t s = Pausa / Play                    \n"
            printf "\t\t u = Silenciar                       \n"
            printf "\t\t + = Subir volumen                   \n"
            printf "\t\t - = Bajar volumen                   \n"
            printf "\t\t l = Mostrar nombre de cancion       \n"
            printf "\t\t q = Salir                           \n"
            echo -e "${W}"
            mpg123 -C --title -q "$cancion"
        else
            echo -e "${R}\t\t\tOpcion invalida.${W}"
            sleep 1.5
        fi
     echo ""
#        read -p "        Presiona Enter para volver al menu..."

    elif [[ "$opcion" == "3" ]]; then
        echo -e "${C}\n\t\t        H  A S T A  L U E G O!${W}\n"
        exit 0
        clear
    else
        echo -e "${R}\t\tOpcion invalida. Intenta otra vez.${W}"
        sleep 1.5
    fi
done
```

3.10. comando salir

`salir` - Finaliza la simulación de la terminal. Es fundamental para darle control al usuario sobre el cierre del entorno. En el mostramos un mensaje de que se esta cerando su sesion, y un mensaje en ASCII diciendole ¡Hata luego!

Listing 9: Script para mostrar el cierre de la sesion

```
salir)
  clear
      echo -e "\n\t\t${M}␣C␣E␣R␣R␣A␣N␣D␣O␣␣␣␣S␣E␣S␣I␣O␣N..."

  echo -e "${C}"
  echo -e "
      .+*+.+*+.+*+.+*+.+*+.+*+.+*+.+*+.+*+.+*+.+*+.+*+.+*+.+*+.+*+.+*+.+*+.+*+.+*+.
      ␣"

  echo -e "
      .+*+.+*+.+*+.+*+.+*+.+*+.+*+.+*+.+*+.+*+.+*+.+*+.+*+.+*+.+*+.+*+.+*+.+*+.+*+.
      ␣"

  echo -e "${W}"

  sleep 3
  clear
      exit 0
      ;;
```

4. Ejecución del Proyecto

Para ejecutar el proyecto completo:

```
cd proyecto_terminal
./terminar.sh
```

Después de autenticarse, se redirige automáticamente a `shell.sh`, donde comienza la simulación de la terminal.

Figura 1: Inicio de sesion incorrecto

Figura 2: Inicio de sesion correcto

Figura 3: Ejecucion del comando 'ayuda'

Figura 4: Ejecucion del comando 'infosis'

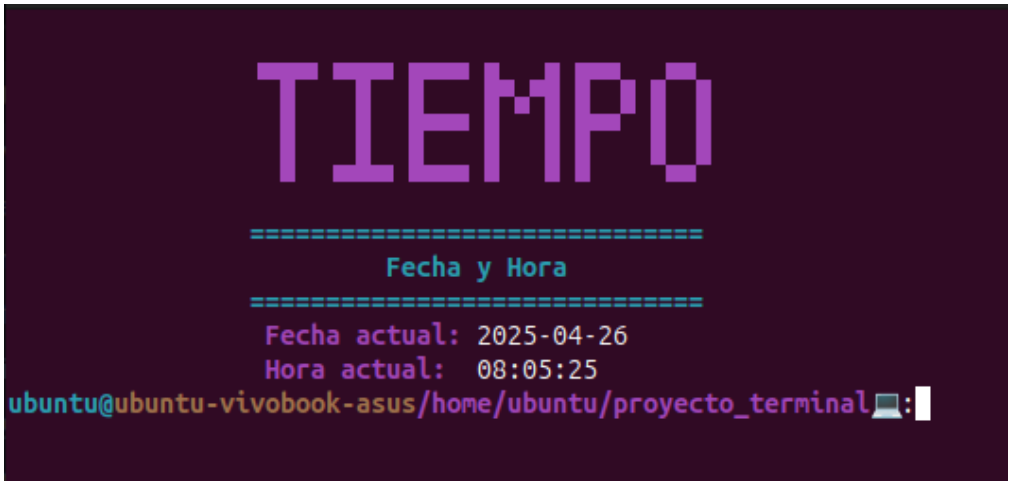

Figura 5: Ejecucion del comando 'tiempo'

Figura 6: Ejecucion del comando 'buscar'

Figura 7: Ejecucion del comando 'buscar' error en la carpeta

Figura 8: Ejecucion del comando 'buscar' no existe el archivo

Figura 9: Ejecucion del comando 'jugar' con partida ganada

Figura 10: Ejecucion del comando 'jugar' con partida empatada

Figura 11: Ejecucion del comando 'musica' menu

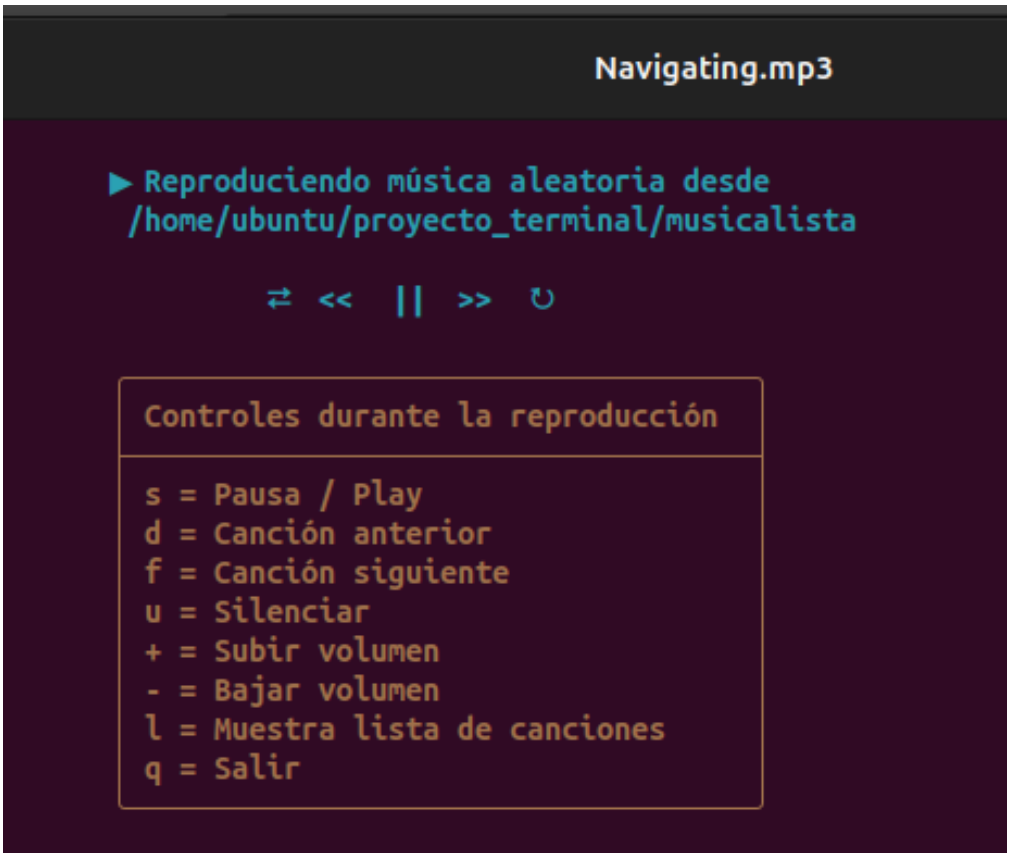

Figura 12: Ejecucion del comando 'musica' reproduccion aleatoria

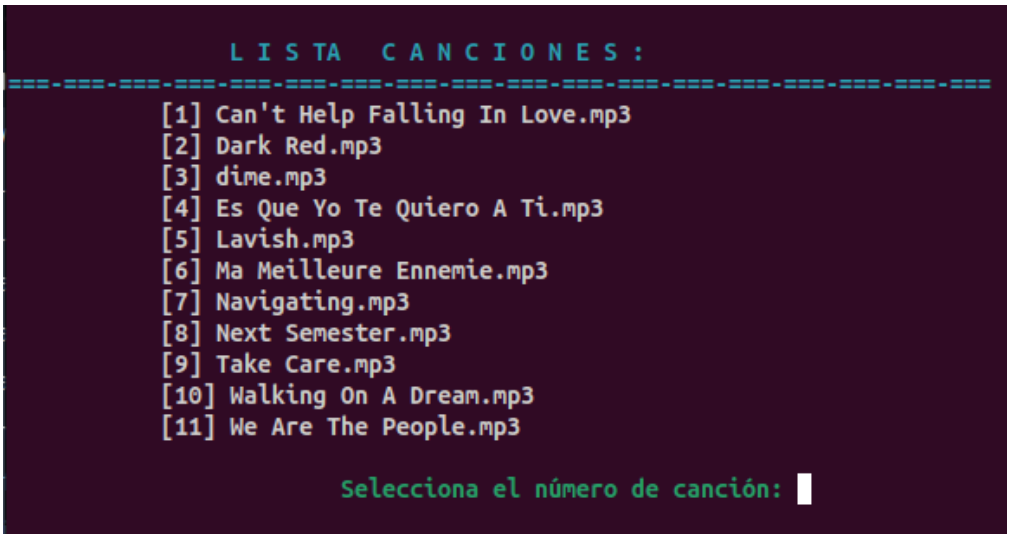

Figura 13: Ejecucion del comando 'musica' elegir cancion

Figura 14: Ejecucion del comando 'musica' reproduccion elegida

Figura 15: Ejecucion del comando 'creditos'

Figura 16: Ejecucion del comando 'creditos'

5. Conclusión

Hernandez Hernandez Deissy Jovita:

El desarrollo de este proyecto me ayudo a seguir poniendo en práctica lo aprendido durante el curso de Linux. A través de la creación de scripts personalizados, pude reforzar comandos básicos del sistema, así como explorar nuevas herramientas, estructuras y funcionalidades del lenguaje **bash.**

Durante el proceso, trabajé con la ejecución de scripts, el uso de colores en la terminal para mejorar la experiencia visual, y la obtención de información directa del sistema, como la fecha, la hora, la memoria RAM, sistema operativo, etc. Estas acciones me ayudaron a comprender mejor cómo interactuar con los archivos del sistema y cómo automatizar tareas cotidianas mediante programación en shell. Aunque en un inicio enfrenté algunas dificultades principalmente por la complejidad que implicaba unir tantos conceptos, sin embargo al final, no solo logré que todos los comandos funcionaran correctamente, sino que también disfruté mucho del resultado final, que me permitió tener una terminal personalizada, interactiva y funcional.

Otra cosa que quiero resaltar de este proyecto, es que fue la primera vez que trabajé con la herramienta *LaTeX*, la cual me pareció sumamente útil para la creación de documentos técnicos con un formato profesional. Aprender a usar *LaTeX* no solo fue importante para documentar el proyecto, sino también para desarrollar habilidades nuevas en cuanto a redacción técnica y estructuración de reportes.

El proyecto no solo me ayudó a reforzar conocimientos teóricos y prácticos, sino también a desarrollar mis habilidades de investigación, resolución de problemas, autonomía y organización. Incluso me permitió explorar mi lado creativo.