Proyecto Linux
PROTECO

Proyecto Final Linux

Olivos Jimenez Luis Mario, Marquina Vazquez Alejandro Jair

1. Introducción

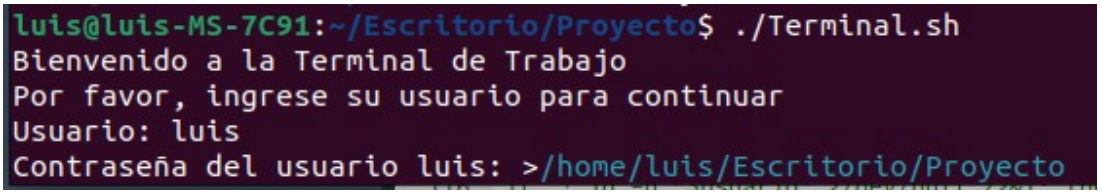
El proyecto consiste en la creación de una terminal de trabajo que permita al usuario trabajar con archivos, utilizar comandos, consultar información en el sistema, reproducir música y jugar juegos programados por el usuario. La terminal requerirá autenticación de usuario y contraseña para acceder y mostrará información esencial del usuario y la ubicación en la carpeta de trabajo. Además, el programa debe ser capaz de interpretar comandos programados por el usuario y los disponibles en el sistema operativo anfitrión. La única forma de salir de la terminal será a través de un comando específico, evitando que los comandos Crtl + C o Crtl + Z cierren el programa. Se han especificado una serie de comandos que deben ser programados, incluyendo comandos para mostrar una lista de comandos disponibles, información del sistema, fecha y hora, búsqueda de archivos, créditos del programador y juegos programados por el usuario. El proyecto también incluirá un reproductor de música con una interfaz gráfica personalizada y opciones para navegar y reproducir canciones.

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

Script 1. Terminal
Aqui tenemos el codigo de nuestra terminal en donde realizamos lo solicitado que es aceptar los comandos ayuda, buscar, creditos, fecha, infosis y gato. Tambien se muestra que al principio definimos dos funciones control c y control z, que se ejecutaran cuando reciba la señal SIGINT (Ctrl-C) y la señal SIGTSTP (Ctrl-Z), respectivamente. Esta función sirve para que nuestro programa no sea interrumpido al usar ctrl c o z y que pueda continuar. A continuación, se solicita al usuario que ingrese su nombre de usuario y contraseña. Si el usuario no existe en el sistema o la contraseña es incorrecta, el script termina. Por ultimo, definimos un arreglo de comandos y un bucle, aqui el script busca l comando ingresado en el arreglo de comandos válidos y, si lo encuentra, ejecuta un script correspondiente. Si el comando es "salir", se sale del bucle y se termina el programa.



Script 2. Comando ayuda

Aqui solo usamos echo para mostrar en pantalla cada comando con una breve explicación de su funcionamiento.

```
>/home/luis/Escritorio/Proyecto  ayuda
Comandos disponibles:
ayuda: Muestra esta ayuda
infosis: Muestra información del sistema
fecha: Muestra la fecha actual
buscar: Busca un archivo en una carpeta especifica
gato: Despliega el juego llamado gato
reproductor: Despliega un reproductor mp3 con interfaz grafica
creditos: Muestra los creditos de los programadores
>/home/luis/Escritorio/Proyecto
```

Script 3. Comando infosis

Este comando inprime informacion como la memoria RAM, arquitectura del sistema y la versión de sistema operativo. Además, se usa el comando "grep"para buscar en el archivo /proc/meminfo la línea que contenga "MemTotal", y luego usamos .ªwk"para imprimir el segundo campo dividido por 1024 dos veces (para convertir los bytes a GB). Esto imprime la cantidad total de memoria RAM disponible en el sistema. También, las líneas comentadas con "#"proveen información adicional sobre la memoria RAM, incluyendo la memoria libre, en uso y el porcentaje de uso. Para la arquitectura del sistema usamos el comando üname -mz por último usamos "lsbrelease d"para imprimir la descripción del sistema operativo, y luego .ªwk"para imprimir el segundo, tercer y cuarto campo (que corresponden a la versión del SO).

```
>/home/luis/Escritorio/Proyecto  infosis
Información del sistema:
-----------------------
Memoria RAM:
Total: 15.547 GB
Libre: 7.88457 GB
En uso: 1.38656 GB
0.0032692 GB
1.38329 GB
Porcentaje de uso:
Arquitectura del sistema: x86_64
Versión del SO: Ubuntu 22.04.2 LTS
>/home/luis/Escritorio/Proyecto
```

Script 4. Comando fecha

Este comando establece la hora local del sistema en el RTC, en lugar de la hora universal coordinada (UTC). timedatectl set-local-rtc 1

Con la siguiente parte del código se imprime el contenido del archivo /proc/driver/rtc, buscando la línea que contiene la fecha del RTC, y utiliza una expresión regular para extraer la fecha en formato YYYY-MM-DD. echo "La fecha de hoy es:çat /proc/driver/rtc —grep 'rtcdate' —grep '[0-9]*-[0-9]*-[0-9]*' -o

La siguiente parte imprime el contenido del archivo /proc/driver/rtc, buscando la línea que contiene la hora del RTC, y utiliza una expresión regular para extraer la hora en formato HH:MM:SS. echo "La hora actual es:çat /proc/driver/rtc —grep 'rtctime' —grep '[0-9]*:[0-9]*:[0-9]*' -o

```
La fecha de hoy es:
2023-04-22
La hora actual es:
19:27:42
```

Script 5. Comando buscar

Este comando busca un archivo en una carpeta y usamos la expresión ¡ -d"para verificar si la carpeta no existe. En caso de que exista la carpeta, se verifica si el archivo especificado se encuentra dentro de ella. La expresión ¡ -f"se utiliza para verificar si el archivo no existe. Ahora si el archivo no se encuentra en la carpeta, se mostrará un mensaje que indique se no se encontró, de lo contrario, si el archivo se encuentra en la carpeta, el script emite un mensaje que indica que el archivo se encontró en la carpeta especificada. Es importante que debemos dar la ruta exacta y el nombre del archivo completo.

```
Contraseña del usuario luis: >/home/luis/Escritorio/Proyecto  buscar
Ingrese la carpeta donde desea buscar: /home/luis/Escritorio
Ingrese el nombre del archivo a buscar: Terminal.txt
El archivo 'Terminal.txt' se encontró en la carpeta '/home/luis/Escritorio'
>/home/luis/Escritorio/Proyecto
```

Script 6. Comando creditos

Este comando solo nos muestra los créditos y aquí solo usamos echo para mostrar texto en la pantalla.

```
>/home/luis/Escritorio/Proyecto-Linux/Proyecto  creditos
==================================
          Créditos
==================================
Programador : [Olivos Jimenez Luis Mario]
Email : [luismarioolivos05@gmail.com]
Git Hub : [https://github.com/LuisOJ]
Programador : [Marquina Vazquez Alejandro Jair]
Email : [alejandromarquina69@gmail.com]
Git Hub : [https://github.com/alejandromarquina69]
==================================
```

Script 7. Comando gato

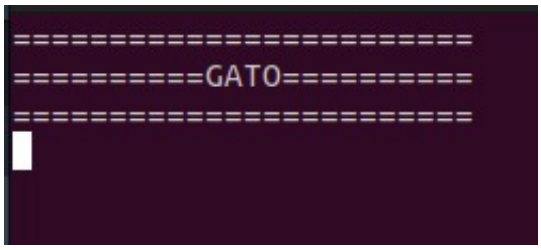
El código del gato comienza definiendo las marcas que ocupara cada jugador:
jugador1="X"jugador2=.º"
A continuación se ocupa la variable de turno para llevar el seguimiento del turno actual
turno=1
Con la siguiente variable se sabe si el juego sigue en marcha o para – juegoencendido=true
A continuación, se ocupa la siguiente variable para saber si se jugara con la computador o con un humano
juego=1
La siguiente matriz representa el tablero movimiento=(1 2 3 4 5 6 7 8 9)
Con la siguiente función damos la bienvenida al juego "bienvenida()"
Esta funcion muestra el tablero en la pantalla "tablero()"
La siguiente es una función que pide al jugador que escoja la casilla en la que quiere poner su marca y actualiza el tablero con la selección del jugador
"seleccionjugador()"
La siguiente función verifica si tres casillas son iguales y marca la bandera juegoencendido como false si es así.
çoicidencia()"

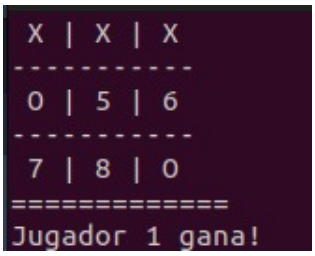
La siguiente parte del código manda a llamar a todas las funciones anteriores para que comience a funcionar el juego

```
bienvenida
tablero
echo "Quieres jugar con la computadora? [Y/n]"
read quien
echo $quien
if [[ $quien == "n" ]]
then
  juego=0
fi

while $juego_encendido
do
  seleccion_jugador
  tablero
  checar_ganador
done
```

Capturas de funcionamiento

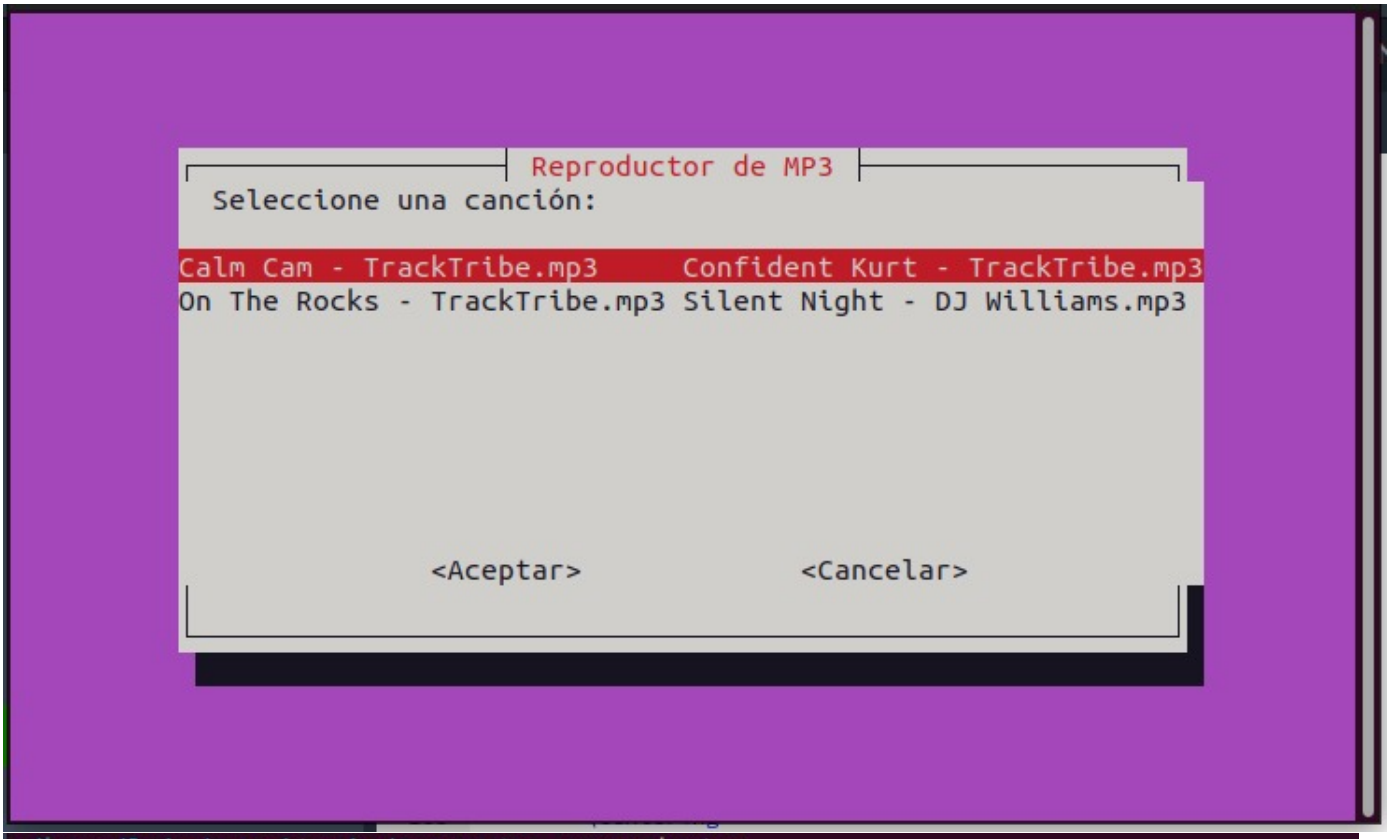




Script 8. Comando reproductor

Este comando muestra un reproductor de música con una interfaz gráfica, aquí verificamos si el programa mpg123 está instalado en el sistema utilizando el comando çommand -v mpg123". Si el programa no está instalado, el script le pregunta al usuario si desea instalarlo. Si el usuario responde "s", el script utiliza el comando "sudo apt-get install mpg123"para instalar el programa. Si el usuario responde "n", el script sale. Después se utiliza el comando "mp3files=(*.mp3)"para crear una matriz que contiene los nombres de archivo de todos los archivos MP3 en la carpeta actual y a continuación se utiliza el comando "whiptail"para mostrar una lista de canciones en un cuadro de diálogo y permitir que el usuario seleccione una canción para reproducir. Por último se usa el comando "mpg123"para reproducir la canción seleccionada.

La última sección se trata de un bucle que permite al usuario navegar por la biblioteca musical. Muestra la información de la canción actual, lee la entrada del usuario y procesa la entrada en función de la opción seleccionada por el usuario. Si el usuario selecciona la opción "n.º "p", muestra la lista de canciones en un cuadro de diálogo y reproduce la siguiente o la canción anterior en la lista, respectivamente. Si el usuario selecciona la opción "q", sale del reproductor. Si el usuario selecciona una opción inválida, el script emite un mensaje de error.




```
>/home/luis/Escritorio/Proyecto  reproductor
High Performance MPEG 1.0/2.0/2.5 Audio Player for Layers 1, 2 and 3
        version 1.29.3; written and copyright by Michael Hipp and others
        free software (LGPL) without any warranty but with best wishes


Terminal control enabled, press 'h' for listing of keys and functions.

Playing MPEG stream 1 of 1: Calm Cam - TrackTribe.mp3 ...

MPEG 1.0 L III vbr 44100 j-s
```

```
[r]     RVA switch
[v]     verbose switch
[l]     list current playlist, indicating current track there
[t]     display tag info (again)
[m]     print MPEG header info (again)
[c] or [C]      pitch up (small step, big step)
[x] or [X]      pitch down (small step, big step)
[w]     reset pitch to zero
[k]     print out current position in playlist and track, for the benefit of som
e external tool to store bookmarks
[h]     this help
[q]     quit

Also, the number row (starting at 1, ending at 0) gives you jump points into the
 current track at 10% intervals.
```

4. Códigos de sh

En esta sección va el código comentado de los métodos numéricos programados

- Terminal

```
#!/bin/bash

function control_c {
    echo -e "\nSeñal SIGINT recibida, el programa no se cerrará."
}

function control_z {
    echo -e "\nSeñal SIGTSTP recibida, el programa no se cerrará."
}

trap control_c SIGINT
trap control_z SIGTSTP

# Acceso a la terminal
echo "Bienvenido a la Terminal de Trabajo"
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

read -s -p "Contraseña del usuario $usuario: " contrasena

if ! echo "$contrasena" | sudo -S id -u >/dev/null 2>&1; then
    echo "La contraseña es incorrecta"
    exit 1
fi

# Comandos
trap '' INT
stty susp ^0
ruta=$(pwd)
comandos=("ayuda" "buscar" "gato" "infosisPro" "creditos" "fecha" "reproductor")
comando=""
while true;
do
    terminalnueva="$(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 "Hasta luego."
```

- Comando ayuda

```
#!/bin/bash


echo "Comandos disponibles:"
echo "ayuda: Muestra esta ayuda"
echo "infosis: Muestra información del sistema"
echo "fecha: Muestra la fecha actual"
echo "buscar: Busca un archivo en una carpeta específica"
echo "gato: Despliega el juego llamado gato"
echo "reproductor: Despliega un reproductor mp3 con interfaz grafica"
echo "creditos: Muestra los creditos de los programadores"
```

- Comando infosis

```
#!/bin/bash

echo "Información del sistema:"
echo "------------------------"
echo "Memoria RAM:"
echo "Total: $(grep MemTotal /proc/meminfo | awk '{print $2/1024/1024 " GB"}')"
#echo "Libre: $(grep MemFree /proc/meminfo | awk '{print $2/1024/1024 " GB"}')"
#echo "En uso: $(grep -w Active /proc/meminfo | awk '{print $2/1024/1024 " GB"}')"
#echo "Porcentaje de uso: $(free | awk '/^Mem:/ {print $3/$2 * 100.0 "%"}')"
echo "Arquitectura del sistema: $(uname -m)"
echo "Versión del SO: $(lsb_release -d | awk '{print $2 " " $3 " " $4}')"
```

- Comando fecha

```
#!/bin/bash

echo "La fecha y hora actual son: $(date +"%Y-%m-%d %H:%M:%S")"
```

- Comando buscar

```
#!/bin/bash


read -p "Ingrese la carpeta donde desea buscar: " carpeta
read -p "Ingrese el nombre del archivo a buscar: " archivo
if [ ! -d "$carpeta" ]; then
    echo "La carpeta '$carpeta' no existe"
elif [ ! -f "$carpeta/$archivo" ]; then
    echo "El archivo '$archivo' no se encontró en la carpeta '$carpeta'"
else
    echo "El archivo '$archivo' se encontró en la carpeta '$carpeta'"
fi
```

- Comando creditos

```
#!/bin/bash

# Función para mostrar los créditos del programador
function mostrar_creditos {
    echo "==================================="
    echo "           Créditos               "
    echo "==================================="
    echo "Programador : [Olivos Jimenez Luis Mario]"
    echo "Email : [luismarioolivos05@gmail.com]"
    echo "Git Hub : [https://github.com/LuisOJ]"
    echo "Programador : [Marquina Vazquez Alejandro Jair]"
    echo "Email : [correo electrónico]"
    echo "Git Hub : [link]"
    echo "==================================="
}

# Llamada a la función para mostrar los créditos del programador
mostrar_creditos
```

- Comando gato

```
#!/bin/bash

jugador_1="X"
jugador_2="O"

turno=1
juego_encendido=true

juego=1

movimiento=( 1 2 3 4 5 6 7 8 9 )
bienvenida() {
  clear
  echo "========================"
  echo "==========GATO=========="
  echo "========================"
  sleep 1
}

tablero() {
  clear
  echo " ${movimiento[0]} | ${movimiento[1]} | ${movimiento[2]} "
  echo "-----------"
  echo " ${movimiento[3]} | ${movimiento[4]} | ${movimiento[5]} "
  echo "-----------"
  echo " ${movimiento[6]} | ${movimiento[7]} | ${movimiento[8]} "
  echo "============="
}

seleccion_jugador(){
  # JUEGA LA PC
  if [[ $juego -eq 1 ]]
  then
   if [[ $(($turno % 2)) == 0 ]]
    then
      jugar=$jugador_2
      casilla=$(( $RANDOM % 10 ))
      echo $casilla
    else
      echo -n "JUGADOR 1 ESCOGE UNA CASILLA:"
      jugar=$jugador_1
      read casilla
    fi
  else
```

```
    # JUEGA HUMANO
    if [[ $(($turno % 2)) == 0 ]]
    then
      jugar=$jugador_2
      echo -n "JUGADOR 2 ESCOGE UNA CASILLA:"
      read casilla
    else
      jugar=$jugador_1
      echo -n "JUGADOR 1 ESCOGE UNA CASILLA:"
      read casilla
    fi
  fi
  echo "no seleccionaste nada"

  space=${movimiento[($casilla -1)]}

  if [[ $casilla =~ ^-[0-9]+$ ]] || [[ ! $space =~ ^[0-9]+$  ]]
  then
    echo "No es una casilla valida."
    seleccion_jugador
  else
    movimiento[($casilla -1)]=$jugar
    ((turno=turno+1))
  fi
  space=${movimiento[($casilla-1)]}
}

coicidencia() {
  if  [[ ${movimiento[$1]} == ${movimiento[$2]} ]]&& \
      [[ ${movimiento[$2]} == ${movimiento[$3]} ]]; then
    juego_encendido=false
  fi
  if [ $juego_encendido == false ]; then
    if [ ${movimiento[$1]} == 'X' ];then
      echo "Jugador 1 gana!"
      return
    else
      echo "Jugador 2 gana!"
      return
    fi
  fi
}

checar_ganador(){
  if [ $juego_encendido == false ]; then return; fi
  coicidencia 0 1 2
  if [ $juego_encendido == false ]; then return; fi
  coicidencia 3 4 5
  if [ $juego_encendido == false ]; then return; fi
  coicidencia 6 7 8
```

```
  if [ $juego_encendido == false ]; then return; fi
  coicidencia 0 4 8
  if [ $juego_encendido == false ]; then return; fi
  coicidencia 2 4 6
  if [ $juego_encendido == false ]; then return; fi
  coicidencia 0 3 6
  if [ $juego_encendido == false ]; then return; fi
  coicidencia 1 4 7
  if [ $juego_encendido == false ]; then return; fi
  coicidencia 2 5 8
  if [ $juego_encendido == false ]; then return; fi

  if [ $turno -gt 9 ]; then
    juego_encendido=false
    echo "Empate!"
  fi
}

bienvenida
tablero
echo "Quieres jugar con la computadora? [Y/n]"
read quien
echo $quien
if [[ $quien == "n" ]]
then
  juego=0
fi

while $juego_encendido
do
  seleccion_jugador
  tablero
  checar_ganador
done
```

- Comando reproductor

```
#!/bin/bash

# Comprobar si mpg123 está instalado
if ! command -v mpg123 &> /dev/null; then
    echo "mpg123 no está instalado"
    read -p "¿Desea instalarlo? (s/n) " option
    if [ "$option" = "s" ]; then
        # Instalar mpg123
        sudo apt-get install mpg123
    else
```

```
        echo "Saliendo del programa"
        exit
    fi
fi

# Listar archivos MP3 en la carpeta actual
mp3_files=(*.mp3)

if [ ${#mp3_files[@]} -eq 0 ]; then
    echo "No se encontraron archivos MP3 en la carpeta actual"
    exit 1
fi

# Mostrar lista de canciones en un cuadro de diálogo
selected_song=$(whiptail --title "Reproductor de MP3" --menu "Seleccione una canción:" 15 60



# Iniciar el reproductor con la canción seleccionada
mpg123 "$selected_song"

# Bucle para permitir la navegación por la biblioteca musical
while true; do
    # Mostrar información sobre la canción actual
    echo "Reproduciendo: $selected_song"

    # Leer la entrada del usuario
    read -p "Opciones: [n] Siguiente canción, [p] Canción anterior, [q] Salir " option

    # Procesar la entrada del usuario
    case $option in
        n)
            # Mostrar lista de canciones en un cuadro de diálogo
            selected_song=$(whiptail --title "Reproductor de MP3" --menu "Seleccione una can
            # Reproducir la siguiente canción en la lista
            mpg123 "$selected_song"
            ;;
        p)
            # Mostrar lista de canciones en un cuadro de diálogo
            selected_song=$(whiptail --title "Reproductor de MP3" --menu "Seleccione una can
            # Reproducir la canción anterior en la lista
            mpg123 "$selected_song"
            ;;
        q)
           # Salir del reproductor
            echo "Saliendo del reproductor de MP3"
            exit
            ;;
        *)
            echo "Opción inválida"
```

```
                ;;
        esac
    done
```

5. Conclusiones

Olivos Jimenez Luis Mario
Este proyecto fue un gran desafío para mí y me permitió adquirir habilidades valiosas en el uso de comandos y shellscript. Fue interesante el investigar mas a fondo para dar solución a problemas que se presentaron lo largo del proyecto. En general, esta experiencia fue muy buena, ya que pude reforzar y comprender mejor varios temas vistos a lo largo del curso. Sin duda, fue un proyecto complicado pero me dejo una buena experiencia, ya que al final logramos concluirlo.

Marquina Vazquez Alejandro Jair
Durante este proyecto retome varios temas vistos en el curso, pude aplicar todos los comandos para realizar los scripts de cada una de las funciones que se pueden realizar en la terminal, fue un poco complicado porque al final se tuvieron que investigar muchas cosas más que no se vieron completas, como en el caso del script que se utiliza para mostrar la fecha y la hora, se tuvo que localizar la carpeta que contiene los datos de la fecha y la hora de la computadora y tuvimos que sacar los datos en con un formato específico. El reproductor fue otra de las cosas más complicadas, darle algún formato gráfico fue tedioso pero logramos que se reprodujeran las canciones. Fue un proyecto un tanto complicado pero sin duda hizo que aprendiera más sobre Linux y las cosas que se pueden realizar en este sistema operativo.