



# Universidad Nacional Autónoma de México

FACULTAD DE INGENIERÍA

Programa de Tecnología en Cómputo

Curso: Linux

## Proyecto Final Documentación ProtoShell

### Impartió:

Brian Jassiel Bautista Pérez

Abraham Galindo Ruiz

Juan Ángeles Hernández

### Alumnos:

Carrasco Meza Alan Michel

García Acevedo Jonathan Enrique

*Generación 44 - Semestre: 2023-2*

Fecha de entrega: 22 de abril del 2023



# Índice

<b>1. Introducción</b>	<b>3</b>
<b>2. Desarrollo</b>	<b>3</b>
2.1. ProtoBash.sh . . . . .	3
2.2. Infosis.sh . . . . .	4
2.3. busqueda.sh . . . . .	4
2.4. fechaHora.sh . . . . .	6
2.5. gato.sh . . . . .	7
2.6. verificarUsuario.sh . . . . .	12
2.7. sistemaAudio.sh . . . . .	14
<b>3. Conclusión</b>	<b>20</b>
3.1. Alan Carrasco . . . . .	20
3.2. Jonathan García . . . . .	20

## 1. Introducción

Shell script es un lenguaje de programación ampliamente utilizado en diversos campo de la computación. Es particularmente útil para hacer uso de los sistemas operativos UNIX-like (Linux o macOS), al igual que mantener servidores con dichos sistemas operativos.

ProtoShell es un proyecto en el que se hace uso de Shell script para realizar tareas como : **\*LISTA DE COSAS QUE PUDIMOS ACABAR DE LA SHELL\***. Esto con la finalidad de profundizar los conocimientos adquiridos a lo largo del curso de Linux y aplicar lo aprendido.

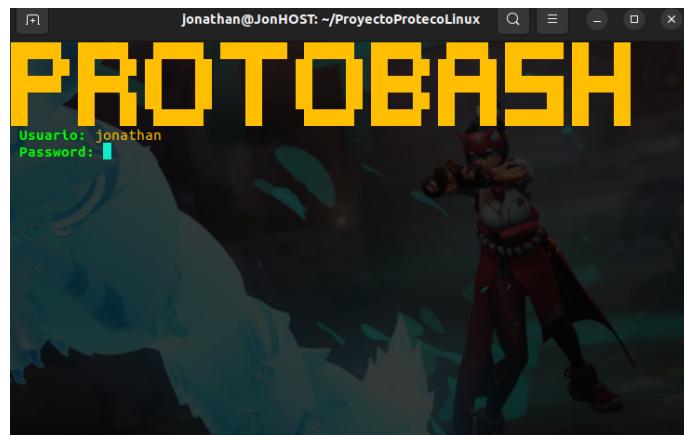
En este archivo se explica el funcionamiento y utilización de cada script que compone a la ProtoShell con la finalidad de que se evidencie el trabajo realizado por los prebecarios y se oriente a los usuarios sobre el correcto uso de este software.

## 2. Desarrollo

### 2.1. ProtoBash.sh

ProtoBash es el documento 'base', en el cual se pueden observar cada una de las funcionalidades del proyecto, como lo son el prompt en el que se va a mostrar el usuario y en que carpeta o directorio en el que se encuentra el usuario, siendo una instrucción que requiere en todo momento nuestro shell script.

Además de cumplir con una sentencia '**case**' con varias opciones, las cuales son las diferentes funcionalidades que en el presente documento se van a ir explicando.



## 2.2. Infosis.sh

En el archivo Infosis.sh podemos encontrar lo siguiente sobre el sistema

La información sobre la memoria RAM.

```
$free -t --mega; grep 'MemTotal' /proc/meminfo; grep 'SwapTotal' /proc/meminfo $
```

La arquitectura con la cual esta constituida la computadora en cuestión

```
$uname -m; $
```

Por último, la identificación de versión del sistema operativo:

```
$lsb_release -idc $
```

```
jonathan@JonHOST: ~/ProyectoProtecoLinux ~& infosis
Se muestra la memoria ram disponible:
total        used        free       shared      buff/cache   available
Memoria:    6128       2445      1376        124      2306       3302
Swap:        2147         0       2147
Total:      8275       2445      3523
MemTotal:   5984412 kB
SwapTotal:  2097148 kB

La arquitectura de su computadora es:
x86_64

La versión de su Sistema Operativo a detalle es:
Distributor ID: Ubuntu
Description:    Ubuntu 22.04.2 LTS
Codename:      jammy

jonathan@JonHOST: ~/ProyectoProtecoLinux ~&
```

## 2.3. busqueda.sh

La presente sección nos ayuda a lograr una búsqueda a partir de los parámetros ingresados por el usuario los cuales son:

El archivo a buscar por nombre con extensión de este:

```
printf "Archivo a buscar: "
```

```
read archivoBuscado
```



El directorio donde se va a buscar, Aquí es suficiente con que se precise el nombre de la carpeta. Ingresar rutas relativas o absolutas producirá un error en la búsqueda.

```
printf "Carpeta en la que buscar: "
```

```
read carpetaBusqueda
```

Una vez teniendo estos parámetros ingresados por el usuario, lo consiguiente se puede iniciar por la función buscarArchivo los cuales usan estos parámetros.

Se usa dentro de la función buscar archivo, transformándola en una variable, el comando find para el que usan como parámetros el primer punto donde se va a hacer la búsqueda

```
"/home/$USER"
```

con la bandera la cual nos afirma que la búsqueda va a hacer mediante el nombre

```
-name
```

y concluir con la variable que anteriormente se inicializo y se le dio el valor por parte del usuario para lograr la búsqueda mediante ese nombre.

```
"$carpetaBusqueda"
```

Logrando así un código final como es el siguiente de este comando:

```
find "/home/$USER" -name "$carpetaBusqueda"
```

Una vez teniendo estos valores definidos podemos empezar a realizar la búsqueda, aunque de inicio lo que se evalúa es si es que lo que se busca existe.

```
if [[ -d "$direccionDirectorio" ]]; then
```

ara continuar entonces con la búsqueda, teniendo que cuenta que se va a dar un nuevo valor para poder lograr y determinar si en verdad se logró la búsqueda

```
direccionArchivo=$(find "/home/$USER" -name "$archivoBuscado")

if [[ -f "$direccionArchivo" ]]
then
    printf "\n$G\$archivoBuscado\$' se encuentra en: $B$direccionArchivo\n"
else
    printf "\n$G\$archivoBuscado\$' $R no se encuentra\$W en la carpeta \$B$carpetaBusqueda\n"
fi

else
    printf "\nLa carpeta \$B$carpetaBusqueda\$R no se ha encontrado\n"
fi
```

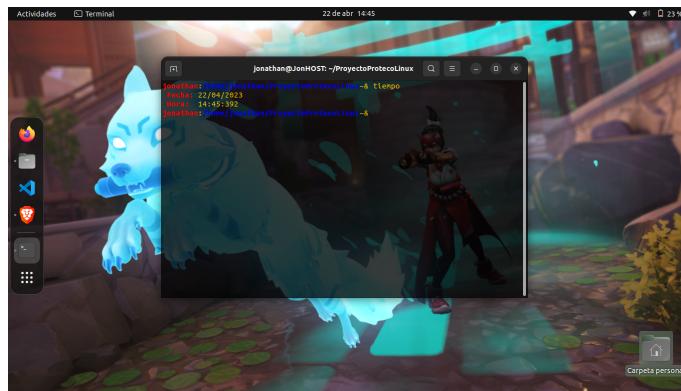
## 2.4. fechaHora.sh

En la presente sección podemos observar dos cosas, la indicación de los colores que se van a dar para la impresión de la fecha:

```
# Colores
R='\033[1;31m'
G='\033[1;32m'
B='\033[1;34m'
W='\033[0m'
```

y la segunda parte que nos muestra en formatos definidos y con ayuda de las variables designadas para el color, la fecha en tipo dd/mm/aa con la hora en formato HH:MM:SS

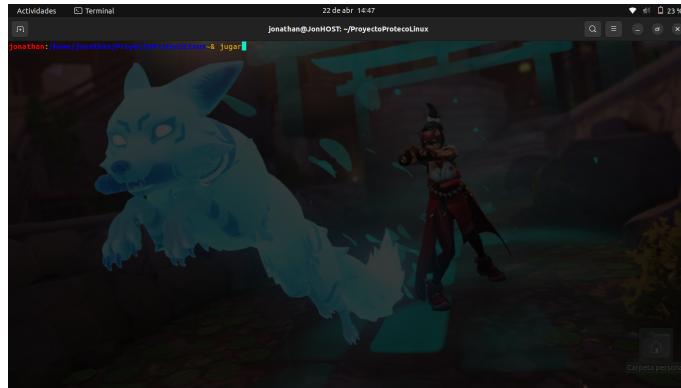
```
printf "$R Fecha:\t$W%(%d/%m/%Y)T\n"
printf "$R Hora:\t$W%(%H:%M:%S2)T\n"
```



## 2.5. gato.sh

Se declara en un arreglo los valores que simbolizan cada casilla

```
s=( [1]=1 [2]=2 [3]=3 [4]=4 [5]=5 [6]=6 [7]=7 [8]=8 [9]=9)
```

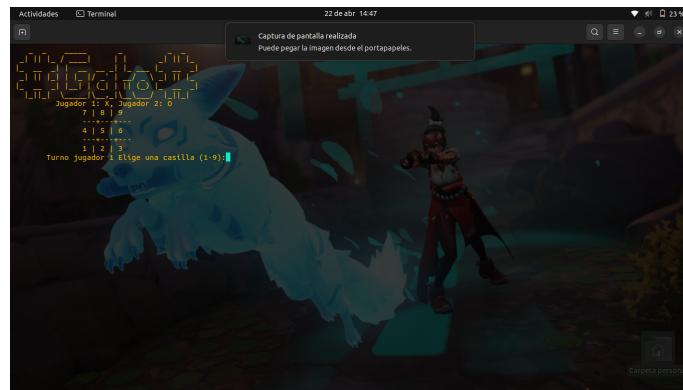


Símbolos para cada jugador

```
SIMBOLO_JUGADOR1=X
SIMBOLO_JUGADOR2=O
```

Función para dibujar el tablero

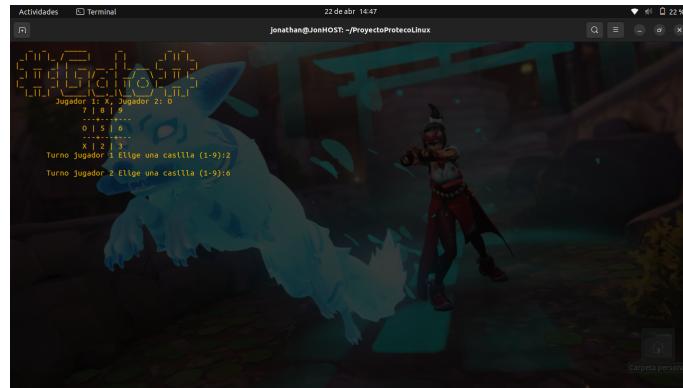
```
dibujarTablero() {
    echo "Jugador 1: ${SIMBOLO_JUGADOR1}, Jugador 2: ${SIMBOLO_JUGADOR2}"
    echo " ${s[7]} | ${s[8]} | ${s[9]} "
    echo "----+---+---"
    echo " ${s[4]} | ${s[5]} | ${s[6]} "
    echo "----+---+---"
    echo " ${s[1]} | ${s[2]} | ${s[3]} "
}
}
```



Expresión regular para verificar que el valor este entre 1 y 9

```
RANGO_NUM='^ [1-9] $'
```

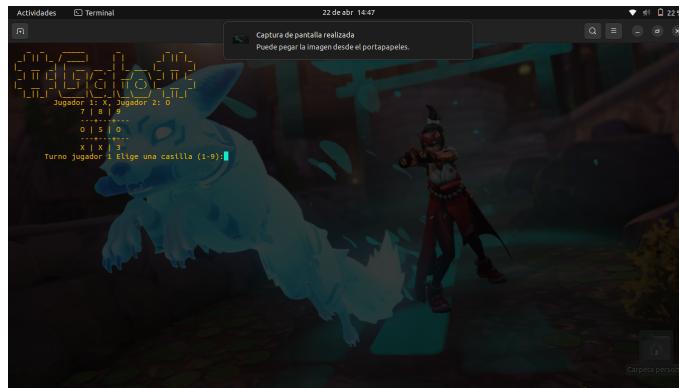
Función para pedirle al jugador 1 su entrada



```
Turno_jugador1() {
    printf "Enter your choice (1-9):"
```

```
read casilla
if ! [[ $casilla =~ $RANGO_NUM ]]; then
    echo "Debes escoger un numero entre 1 y 9"
    Turno_jugador1
fi
if ! [[ ${s[$casilla]} =~ $RANGO_NUM ]]; then
    echo "Casilla ocupada"
    Turno_jugador1
fi
s[$casilla]="$SIMBOLO_JUGADOR1"
}
```

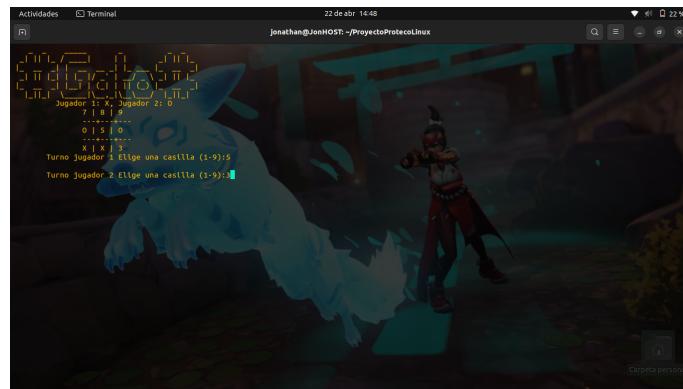
Función para pedirle al jugador 2 su entrada



```
Turno_jugador2(){
    printf "Enter your choice (1-9):"
    read casilla
    if ! [[ $casilla =~ $RANGO_NUM ]]; then
        echo "Debes escoger un numero 1 y 9"
        Turno_jugador2
    fi
    if ! [[ ${s[$casilla]} =~ $RANGO_NUM ]]; then
        echo "Casilla ocupada"
        Turno_jugador2
    fi
    s[$casilla]="$SIMBOLO_JUGADOR2"
```

{}

Función para saber qué jugador se debe imprimir

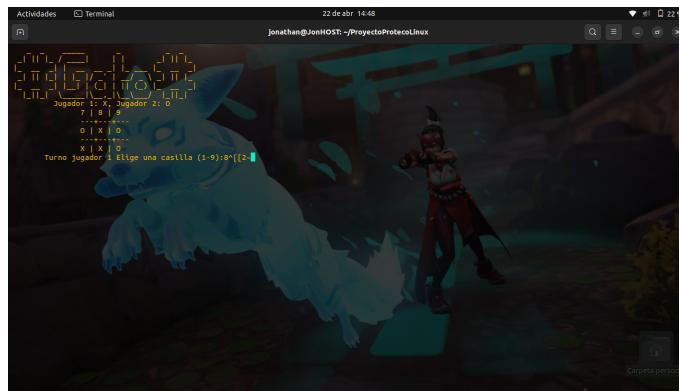


```
jugador() {  
    local SIMBOLO=$1  
    [[ $SIMBOLO == $SIMBOLO_JUGADOR1 ]] && printf "Jugador 1" || printf "Jugador 2"  
}
```

Función para mostrar al ganador

```
gana() {  
    local GANADOR=$1  
    echo "%%%%%%%%%%%%%%"  
    echo "      $(jugador $GANADOR) gana!"  
    echo "%%%%%%%%%%%%%%"  
    dibujarTablero  
    exit 0  
}
```

Función para evaluar casos de victoria



```
verificarGanador(){
```

Evaluar filas

```
for i in 1 4 7; do
  j=$((i + 1))
  k=$((i + 2))
  GANADOR=${s[$i]}
  [[ ${s[$i]} == ${s[$j]} ]] && [[ ${s[$j]} == ${s[$k]} ]] && gana $GANADOR
done
```

Evaluar columnas

```
for i in 1 2 3; do
  j=$((i + 3))
  k=$((i + 6))
  GANADOR=${s[$i]}
  [[ ${s[$i]} == ${s[$j]} ]] && [[ ${s[$j]} == ${s[$k]} ]] && gana $GANADOR
done
```

Evaluar diagonales

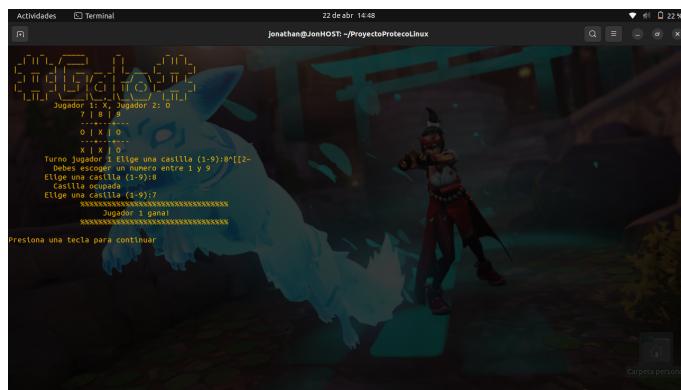
```
GANADOR=${s[5]}
[[ ${s[1]} == ${s[5]} ]] && [[ ${s[5]} == ${s[9]} ]] && gana $GANADOR
[[ ${s[7]} == ${s[5]} ]] && [[ ${s[5]} == ${s[3]} ]] && gana $GANADOR
}
while true; do
  dibujarTablero
```

```

echo "Turno jugador 1"
Turno_jugador1
verificarGanador
echo ""
echo "Turno jugador 2"
Turno_jugador2
verificarGanador

done

```



## 2.6. verificarUsuario.sh

verificarUsuario.sh es un programa que verifica la identidad del usuario y compara las credenciales que el usuario suministra contra las existentes en el sistema. En esta sección se explica paso a paso la forma en que se logró dicha comparación.

Lo primero que el script hace es solicitar las credenciales del usuario (usuario y contraseña). Estas credenciales deben existir en el sistema.

```

printf "$G Usuario: $W"
read usuario
printf "$G Password: $W"
read -s password

```



Aquí es donde empieza lo jugoso del código. Primero, se tiene que acceder al archivo shadow, ubicado en el directorio etc/. De ese archivo se recupera la cantidad de veces que el nombre de usuario aparece para saber si el usuario provisto es correcto.

```
cadena=`sudo -S grep -r $usuario /etc/shadow`
```

```
if [ ${cadena} > 1 ]; then
```

Posteriormente, la cadena obtenida con las credenciales del usuario se separan en sub-cadenas, cuyo indicador de separación es el carácter '\$'. Esta segmentación de la cadena se hace debido a que cada componente tiene información valiosa para poder validar la contraseña provista por el usuario. El primer índice del array contiene el tipo de encriptación utilizada (yescrypt por defecto para Ubuntu desde la versión 11). El segundo y tercer índice contienen la *sal* (clave) utilizada para perturbar el algoritmo de encriptación.

```
IFS='$' read -e -r -a array <<< "$cadena"
salt_hashed="\${array[1]}\${array[2]}\${array[3]}$"
```

Con lo anterior, el siguiente paso es invocar python3 para poder hacer uso del módulo crypt y poder encriptar la contraseña provista por el usuario utilizando el mismo algoritmo y la misma sal para comparar el hash generado.

```
hash=`python3 -c 'import crypt;import sys;print(crypt.crypt(sys.argv[1],sys.argv[2]));'
$password $salt_hashed`
```

Finalmente, con el hash generado y el hash registrado en el sistema, se procede a hacer una

comparación de cadenas para determinar si son compatibles.

```
match=`echo "$cadena" | grep -c "$hash"`
if [ "$match" -eq 1 ]; then
```

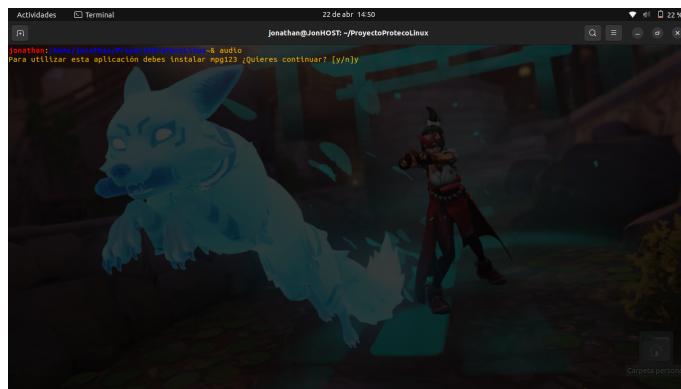
De esa forma se concluye la forma de hacer la búsqueda del usuario con su contraseña, a partir de sentencias if, con su búsqueda respectiva carácter por carácter para confirmar dicha información.

## 2.7. sistemaAudio.sh

En esta parte del sistema se desarrolló un shell script capaz de reproducir por medio de instrucciones básicas en la terminal, intentando realizar lo más cercano a una GUI que se pudo desde una terminal.

Para comenzar la descripción, se incluyeron variables que se usan para identificar los colores que se van a ocupar

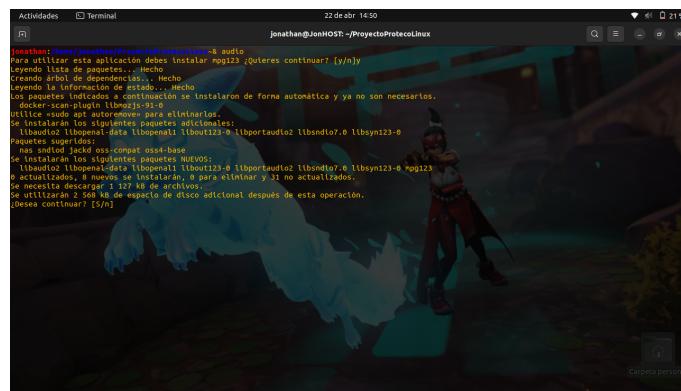
```
# Colores
R='\033[1;31m'
G='\033[1;32m'
B='\033[1;34m'
W='\033[0m'
```



Continuando podemos observar la dirección del reproductor, en su directorio para guardarla en una variable y poder ocuparlo después

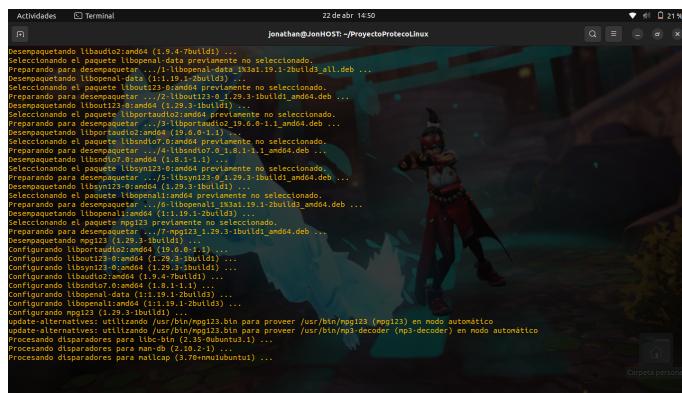
```
# Dirección del reproductor
reproductor=/bin/mpg123
```

Se verifica que este instalado el reproductor en el directorio, y si no, se prevé esta situación con una instalación posterior para lograr usar el reproductor.



```
#Comprobar que si este instalado
if [ ! -f "$reproductor" ];
then
    printf "Para utilizar esta aplicación debes instalar mpg123 ¿Quieres continuar? [y/n]"
    read instalarYN
    case $instalarYN in
        'y') sudo apt install mpg123 ;;
        'n') exit 0 ;;
        *) printf "Opción no válida, escribe \'y\' o \'n\'";;
    esac
fi
```

Después de declarar variables se hace el menú con la petición de escribir los comandos necesarios para su funcionamiento, los cuales se guardan temporalmente en una variable para su uso mientras que no se le de la opción salir.



```
# Por defecto la musica se toma de la carpeta en el proyecto  
dirMusica=musiquitaGratis/
```

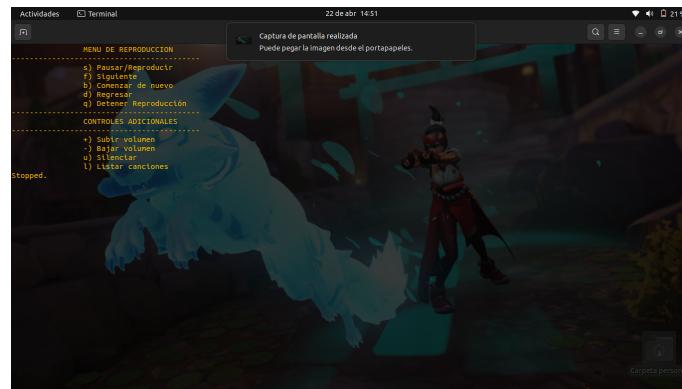
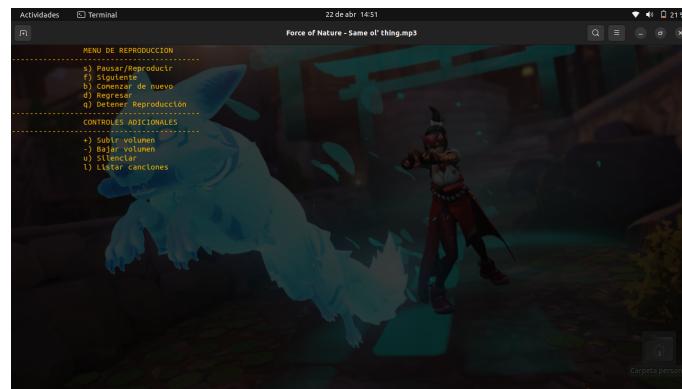
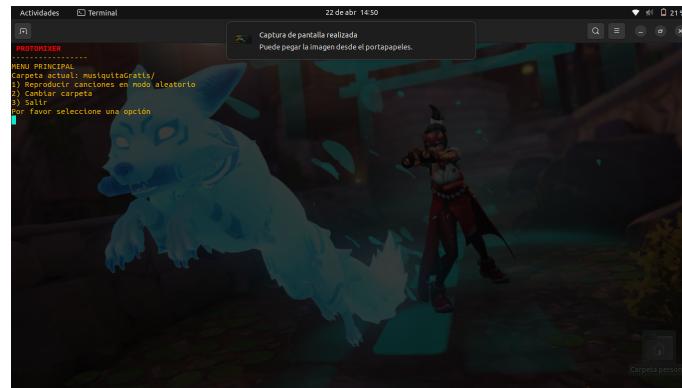
```
while [ $salir -ne 1 ]
do
    clear
    printf "$R PROTOMIXER\n$W"
    echo "-----"
    echo "MENU PRINCIPAL"
    echo "Carpeta actual: $dirMusica"
    echo "1) Reproducir canciones en la carpeta"
    echo "2) Cambiar carpeta"
    echo "3) Salir"

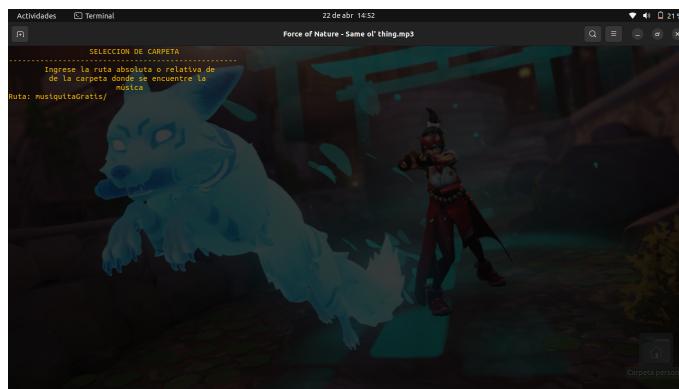
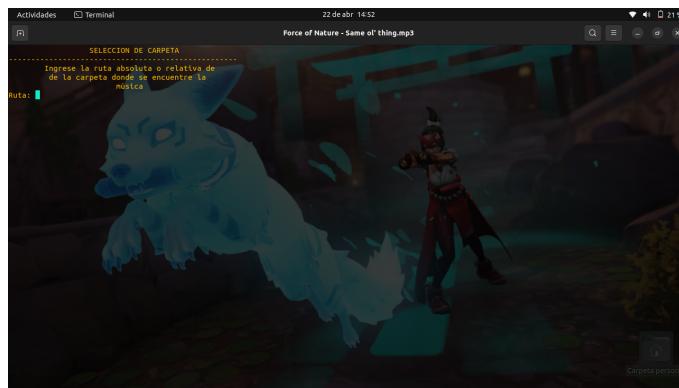
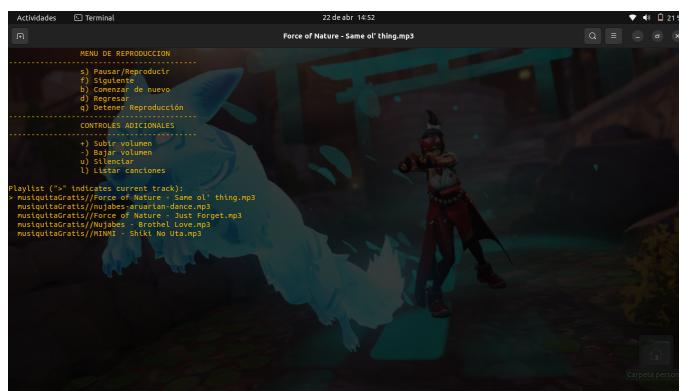
    printf "Por favor seleccione una opcion"
    read -e opcion

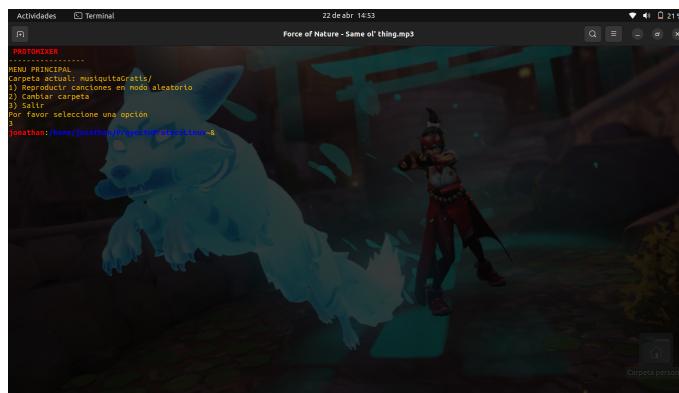
    case $opcion in
        1)
            clear
            echo ""
            echo "-----"
            echo "
```

```
echo "f) Siguiente"
echo "b) Comenzar de nuevo"
echo "d) Regresar"
echo "q) Detener Reproducción"
echo "-----"
echo "CONTROLES ADICIONALES"
echo "-----"
echo "+) Subir volumen"
echo "-) Bajar volumen"
echo "u) Silenciar"
echo "l) Listar canciones"
mpg123 -C --title -q -z "${dirMusica}"//*
;;
2)
clear
echo "SELECCION DE CARPETA"
echo "-----"
echo "Ingrese la ruta absoluta o relativa de"
echo "de la carpeta donde se encuentre la"
echo "música"
read -e -p "Ruta: " dirMusica
while [ ! -d $dirMusica ]
do
    echo "Esa dirección no es válida, inténtalo nuevamente"
    read -e -p "Ruta: " dirMusica
done
;;
3) exit 0 ;;
*)
    echo "Opción inválida, elige una opción del menú"
    sleep 1.5
;;
esac
```

done







### 3. Conclusión

#### 3.1. Alan Carrasco

En este proyecto se realizó una terminal de comandos capaz de interpretar comandos personalizados y los propios del sistema operativo. Las funciones personalizadas que se pueden ejecutar con esta terminal son, como ya se mencionó, buscar, infosis, ayuda, créditos, tiempo, jugar y audio. Cabe mencionar que el proyecto tiene sus limitaciones tanto funcionales, como estéticas. Sin embargo, para haber tomado un curso muy intensivo sobre GNU/Linux y jamás haber hecho cosas como encriptación o uso de programas para audio desde terminal, resultó muy enriquecedor.

Finalmente, quiero agradecer a los instructores de Linux por haber hecho su mejor esfuerzo por transmitirnos sus conocimientos; a mi buddy y a mis demás compañeros por haber colaborado y llegar a soluciones juntos; y a todas las personas que comparten sus proyectos y conocimientos en internet. Sin alguno de ellos este proyecto no se habría podido llevar a cabo. Gracias.

#### 3.2. Jonathan García

Puedo decir que el proyecto fue bastante laborioso y en parte difícil por la búsqueda de información, de incorporar esta información y claro, contemplar cada uno de los archivos para su acoplamiento correcto y uso correcto de cada una de estas funciones en los diferentes archivos que teníamos, tanto de tipo shell script como de tipo txt.

Contemplar en su mayoría, todos los requerimientos establecidos en el proyecto como lograr esa cohesión entre lo trabajado con mi buddy y lo que yo hice, fue algo bastante satisfactorio por el hecho de ver los resultados y lograr hacer un proyecto de calidad y diseño único.