

Terminal de trabajo PREBE

Lechuga Milpas Ángel Alberto
Meza Sánchez Luis Arturo

23 de abril de 2023

Índice

1. Introducción	2
2. Desarrollo	3
2.1. Composición de la Terminal	3
2.1.1. Sistema de acceso	3
2.1.2. Sistema de interacción con el usuario	4
2.2. Comandos programados	4
2.2.1. Comando 'ayuda'	4
2.2.2. Comando 'buscar'	4
2.2.3. Comando 'creditos'	5
2.2.4. Comando 'infosis'	5
2.2.5. Comando 'jugar'	5
2.2.6. Comando 'limpiar'	6
2.2.7. Comando 'musica'	6
2.2.8. Comando 'tiempo'	7
2.2.9. Comando 'salir'	7
3. Capturas de la temrinal	8
4. Conclusiones	9

1. Introducción

Este proyecto tiene como finalidad poner en práctica los conocimientos adquiridos durante el curso de GNU Linux. Se aplicarán conceptos y conocimientos de Bash así como el manejo de los comandos de Linux para realizar una Terminal la cual será funcional y tendrá diversas funciones que se programan por nosotros mismos.

Este proyecto junta nuestra habilidad de investigación con nuestra capacidad de poder resolver los distintos retos que se nos presenten durante la realización de esta terminal.

2. Desarrollo

2.1. Composición de la Terminal

Para la realización de la 'Terminal PREBE' lo que se hizo fue tratar de dividir todo el proyecto en partes, lo más pequeñas posibles, para que todo el desarrollo del código y el manejo de los archivos que componen al mismo fueran más sencillos de comprender y manipular.

Por lo que podemos decir que se divide en dos partes:

1. Sistema de acceso.
2. Sistema de interacción con el usuario.

2.1.1. Sistema de acceso

Lo que se hizo en este script fue pedir el nombre de usuario y la contraseña. Posteriormente se creó una función para comprobar si estos datos coinciden, esto con la ayuda de un método de encriptación de Python, esa parte del código es tomada del repositorio de Jhon y Abraham, me reuní con Alan para poder implementar esta parte en nuestros respectivos códigos y salió bastante bien. En pocas palabras esta parte del programa solo verifica si la información del usuario es correcta y procede a dar acceso a la siguiente parte, el sistema de interacción.

```
1  validacionDatos(){
2      # Comprobar si el nomUsuario existe en el sistema
3      if id "$nomUsuario" >/dev/null 2>&1
4      then
5          # Comprobar si la contraseña del nomUsuario es correcta
6          match=`echo "$cadena" | grep -c "$hash"`
7          if [ "$match" -eq 1 ]
8          then
9              lineaxD
10             printf "\n\n\t\t\t $Glig B I E N V E N I D O\n $W
11             ↪ \t\t\t\t $nomUsuario \n\n"
12             sleep 2
13             ./interaccion.sh
14         else
15             lineaxD
16             printf "\n\n\t\t\t $M C O N T R A S E Ñ A   I N C O R R E
17             ↪   C T A\n $W \n\n"
18         fi
19     else
20         lineaxD
21         printf "\n\n\t\t\t $B   U S U A R I O   N O   E X I S T E\n $W
22         ↪   \n\n"
```

```
20         fi
21     }
```

2.1.2. Sistema de interacción con el usuario

El sistema de interacción con el usuario, es por así decirlo, el controlador con el cual se irán llamando los diferentes scripts que se programaron para cada comando.

En esta parte solamente se le pide una entrada al usuario para ir accediendo a los comandos programados y también a los comandos propios del SO, esto se logró con una sentencia 'case' ya que así se pueden leer también los comandos nativos sin ningún problema.

2.2. Comandos programados

2.2.1. Comando 'ayuda'

Para el comando de 'ayuda' lo que se hizo fue generar un archivo en formato '.txt' en este solo se muestran los comandos disponibles y una breve descripción del mismo. Solo usamos el comando 'cat' para ver este archivo

2.2.2. Comando 'buscar'

Para este script lo que se le solicita al usuario es una carpeta y un archivo a buscar, lo que se hace es que primero se comprueba la existencia de la carpeta, en caso de que no existe se manda un mensaje de advertencia. En caso contrario, que sí exista, se busca dentro de esa carpeta el archivo solicitado, y se manda un mensaje avisando al usuario si existe o no tal archivo dentro de esta carpeta
Imagen: ??

```
1     buscarInformacion(){
2 #find <ruta> <opciones> <patrón>
3     #find . -name "archivoBuscado" Parabuscarenelmismodirectorio
4     ubiCarpeta=$(find "/home/$USER" -name "$carpeta") # 1. Buscamos la
      ↳ carpeta
5
6     # test -d "direccion"
7     if [[ -d "$ubiCarpeta" ]] # 2. Validamos si la carpeta existe con "test
      ↳ -f/d"
8     then # 2.a) Existe la carpeta
9         ubiArchivo=$(find "/home/$USER" -name "$archivoBuscado") # 3.
      ↳ Buscamos archivo
10        if [[ -f "$ubiArchivo" ]] # 4. Validamos si existe el archivo
11        then # 4.a) Existe archivo en carpeta
12            lineaJajs
```

```

13         printf "\n\t Archivo$Glig encontrado$W
           ↪ en:\n$ubiArchivo"
14     else # 4.b) No existe el archivo en la carpeta
15         lineaJajs
16         printf "\n No existe el archivo $Y en la carpeta. $W"
17     fi
18
19     else # 2.b) No existe la carpeta
20         lineaJajs
21         printf "\n La carpeta$R no existe$W en el sistema."
22     fi
23 }

```

2.2.3. Comando 'creditos'

Este es un archivo de texto, el cual solo es un pequeño cartel en el que se muestran los nombres de los desarrolladores del proyecto, al igual que con el archivo de texto de ayuda, solo se usa el comando 'cat'.

2.2.4. Comando 'infosis'

Para realizar este comando se realizó un script en cual se obtiene la información de la memoria RAM, la arquitectura de la computadora y la versión del sistema operativo con el siguiente código:

```

imprimirInfo(){
printf "$B\n\t1. Aquitectura\n$W$arquitect \n"
printf "$B\n\t2. Memoria RAM\n$W $memorRAM \n"
printf "$B\n\t3. Versión del SO\n$W $versionSO \n"
printf "\n"
}

```

2.2.5. Comando 'jugar'

Para este comando se creó todo un script con un juego, este juego fue el juego de "Gato", el cual tiene varias funciones. Una función para elegir la casilla según el turno del jugador y la función de comprobar el ganador. Así como funciones de imprtesión para darle formato al tablero y a los turnos. Es un juego sencillo pero funcional.

El único detalle que tiene es que si se colocan mal las casillas, el jugador que eligió las casillas va a tener doble turno y así ganar más facilmente, es algo que no pude solucionar pero ahí se quedó. No afecta mucho.

2.2.6. Comando 'limpiar'

Este es un comando muy simple, solo se ejecuta el comando 'clear' el cual ya es nativo del SO, solo que quería agregarlo.

2.2.7. Comando 'musica'

Este comando fue uno de los más complicados. Cabe recalcar que para este reproductor se uso el programa 'mpg123'.

Con este comando lo que se hace es que se ejecuta un script el cual hace varias cosas. Primero verifica si el programa antes mencionado está instalado, si no se pregunta si se quiere instalar. Posteriormente se nos dirige a la ubicación de la música la cual es una carpeta que se encuentra dentro de la carpeta del proyecto donde se encuentra toda la música.

Una vez situados ahí tenemos tres opciones. la de reproducir la música en aleatorio, la de reproducir alguna canción deseada y la de salir. Y con los mismos comandos del programa 'mpg123' se hace el control de la reproducción de las canciones.

```
main(){
opcion=0
lugarMusica=musicaPREBE/ # Obtenemos la ruta de la música
imprimirTitulo # 1. Imprimimos para que se vea bonito
comprobarmpg123 # 2. Comprobamos que exista el reproductor

while [ "$opcion" != 3 ]
do
clear
imprimirTitulo # Imprimimos para que se vea bonito
imprimirMenu # Mostramos menú
leerOpcionMenu # Vemos qué quiere hacer el usuario

case $opcion in
1) # Canciones aleatorias
clear
imprimirTitulo
imprimirControlador
mpg123 -C --title -q -z "${lugarMusica}"/ * # Este comando Alan me lo proporcionó
;;
2) # Canción específica
clear
imprimirTitulo
elegirCancionFav
clear
imprimirTitulo
imprimirControladorIndividual
mpg123 -C --title -q -z "${lugarMusica}"/"$cancionFav"
```

2.2.8. Comando 'tiempo'

```
imprimirFechaHora(){
printf "$B\n\t\tFecha:$W %(%Y-%m-%d)T \n"
printf "$B\n\t\tHora:$W %(%H:%M:%S)T \n"
}
```

Este comando es un simple 'exit 0' de la sentencia case, una vez que se ejecuta este comando el flujo del programa de interacción con el usuario se rompe y se nos muestra una cuenta regresiva que programé y nos saca completamente de la TerminalPrebe.



Figura 1: Sistema de acceso

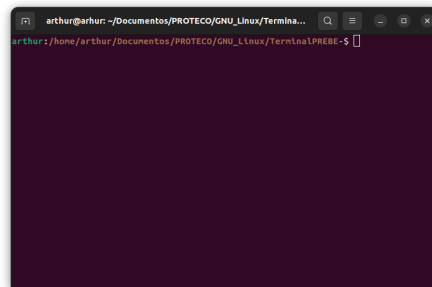


Figura 2: Sistema de interaccion

3. Capturas de la temrinal

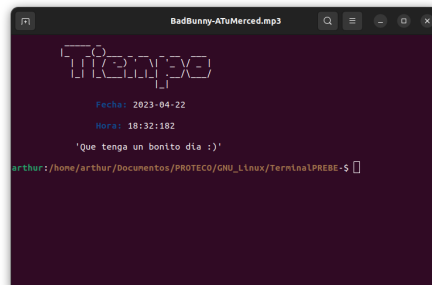
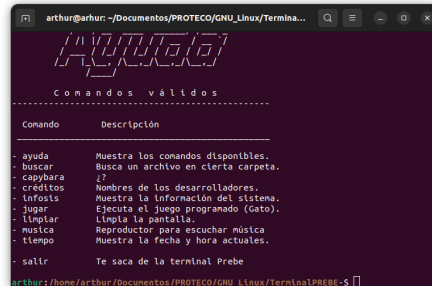


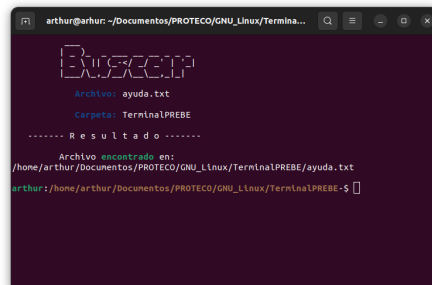
Figura 9: Comando 'tiempo'



```
arthur@arhur: ~/Documentos/PROTECO/GNU_Linux/TerminalPREBE
Ayuda
-----
Comandos válidos
-----
Comando      Descripción
-----
- ayuda      Muestra los comandos disponibles.
- buscar     Busca un archivo en cierta carpeta.
- capybara   ¿?
- créditos   Muestra los nombres de los desarrolladores.
- infosis    Muestra la información del sistema.
- jugar      Ejecuta el juego programado (Gato).
- limpiar    Limpia la pantalla.
- musica     Reproductor para escuchar música.
- tiempo     Muestra la fecha y hora actuales.
- salir      Te saca de la terminal Prebe

arthur:/home/arthur/Documentos/PROTECO/GNU_Linux/TerminalPREBE-$
```

Figura 3: Comando 'ayuda'



```
arthur@arhur: ~/Documentos/PROTECO/GNU_Linux/TerminalPREBE
Buscar
-----
Archivo: ayuda.txt
Carpeta: TerminalPREBE
----- Resultado -----
Archivo encontrado en:
/home/arthur/documentos/PROTECO/GNU_Linux/TerminalPREBE/ayuda.txt
arthur:/home/arthur/Documentos/PROTECO/GNU_Linux/TerminalPREBE-$
```

Figura 4: Comando 'buscar'

4. Conclusiones

Con la realización de este proyecto, lo que se logró fue poner en práctica los conocimientos que se adquirieron en Linux. Se resolvieron los diversos problemas que se fueron presentando a lo largo del trabajo con ayuda de la búsqueda de información en internet y consulta de fuentes en diversos idiomas. Se puso en práctica la creatividad para poder realizar un código lo más limpio posible, así mismo un formato de salida sencillo y agradable para una mejor experiencia de usuario. Se adquirieron muchos conocimientos acerca de bash y sus funciones y aplicaciones. De igual manera se puso en práctica la lógica de programación para tener un buen flujo en el código.

```
arthur@arhur: ~/Documentos/PROTECO/GNU_Linux/Terminal$  
  
Sistema  
Información  
  
1. Arquitectura  
x86_64  
  
2. Memoria RAM  
total used free shared buff/cache availabl  
e  
Memoria: 6.7Gi 2.8Gi 922Mi 115Mi 3.8Gi 3.5Gi  
Swap: 7.4Gi 0B 7.4Gi  
  
3. Versión del SO  
Distributor ID: Ubuntu  
Description: Ubuntu 22.04.2 LTS  
Codename: jammy  
  
arthur@arhur:~/Documentos/PROTECO/GNU_Linux/Terminal$
```

Figura 5: Caption

```
arthur@arhur: ~/Documentos/PROTECO/GNU_Linux/Terminal$  
  
Sistema  
Información  
  
1. Arquitectura  
x86_64  
  
2. Memoria RAM  
total used free shared buff/cache availabl  
e  
Memoria: 6.7Gi 2.8Gi 922Mi 115Mi 3.8Gi 3.5Gi  
Swap: 7.4Gi 0B 7.4Gi  
  
3. Versión del SO  
Distributor ID: Ubuntu  
Description: Ubuntu 22.04.2 LTS  
Codename: jammy  
  
arthur@arhur:~/Documentos/PROTECO/GNU_Linux/Terminal$
```

Figura 6: Caption

```
arthur@arhur: ~/Documentos/PROTECO/GNU_Linux/Terminal$  
  
M=3  
GATO  
Jugador1 : o Jugador2 : x  
  
7 | 8 | 9  
-----  
4 | x | 6  
-----  
0 | 2 | 3  
  
[Jugador1] Elige un número: (1-9):
```

Figura 7: Comando 'jugar'

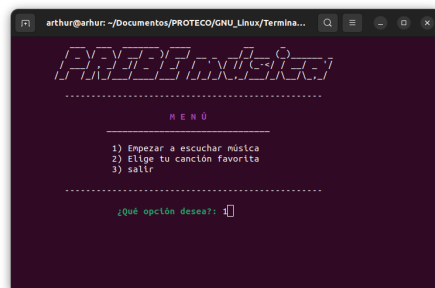


Figura 8: Comando 'musica'