

ProyectoShell

Flores Torres Ángel Joel y Medina Rubalcava Natalia Michell

22 September 2023

1. Índice

1. Índice.....	2
2. Objetivo.....	3
3. Introducción.....	3
4. Desarrollo.....	3
4.1 Acceso.....	3
4.2 Terminal.....	5
4.3 Comandos.....	7
4.3.1 ayuda.....	7
4.3.2 buscar.....	8
4.3.3 creditos.....	10
4.3.4 infosis.....	12
4.3.5 mp3.....	13
4.3.6 fechayhora.....	23
5. Conclusiones.....	32

2. Objetivo

Mostrar y aplicar los conocimientos adquiridos a lo largo del curso de GNU/Linux.

3. Introduction

El proyecto consiste en realizar una terminal de Shell script, en donde se tenga que iniciar sesión con alguna cuenta que ya exista en el sistema, luego se ejecuten comandos como: ayuda, información del sistema, buscar archivo, fecha y hora, juego, reproductor mp3. Además tomando en cuenta ciertas restricciones como que ciertos comandos como `Ctrl+Z`, `Ctrl+X`, `Ctrl+C` `.exit` no sean válidos dentro de la terminal y se muestre un formato que simule una terminal real

4. Desarrollo

4.1. Acceso

Primero se define un número de intentos máximo para poder ingresar, en este caso 3 intentos, luego se pide ingresar el usuario, se verifica que el usuario exista en el sistema y de ser así se pide la contraseña, misma que se verifica sea la correcta, en caso de que se ingresen bien ambos datos se permite ingresar a la terminal. En caso de que el usuario sea incorrecto se vuelve a pedir (tomando en cuenta que sólo se tienen 3 intentos), en caso de que el usuario sea correcto y la contraseña incorrecta se vuelve a pedir hasta llegar al número de intentos, si se llega al límite de intentos se muestra un mensaje y sale de la ejecución.

Código

```
1 #!/bin/bash
2
3 intentos=3
4
5 while [ $intentos -ge 0 ]; do
6
7     clear
8     printf "\n\tIngrese el usuario: "
9     read user
10
11     if ! id -u "$user" >/dev/null 2>&1; then
12         printf "\n\tEl usuario no existe"
13         read
14         clear
15         intentos=$((intentos-1))
```

```

16     else
17         printf "\n\tIngrese la contraseña: "
18         read pass
19         printf "\nValidando contraseña, espere un
           momento..\n"
20
21         # Utiliza el comando 'su' para verificar
           la contraseña
22         if su -c "true" "$user" <<< "$pass" >/dev
           /null 2>&1; then
23             printf "\nContraseña correcta,
           puede ingresar."
24                 printf "\n
           nPresiona
           enter para
           continuar"
25                 read
26                 clear
27                 ./ayuda.sh
28                 ./terminal.sh
29                 exit 0
30         else
31             printf "\nContraseña incorrecta"
32                 read
33                 clear
34             intentos=$((intentos-1))
35         fi
36     fi
37 done
38
39 printf "\n\tLlego al limite de intentos, reintente mas
           tarde"

```

```

angel@angel-ubuntu ~/D/TerminalPrebe_PROTECO (main)> ./acceso.sh

```

Figura 1: Ejecución del archivo acceso.sh

```

Ingrese el usuario: angel
Ingrese la contraseña: UbuntuTorres1204
Validando contraseña, espere un momento..
Contraseña correcta, puede ingresar.
Presiona enter para continuar

```

Figura 2: Usuario y contraseña correcta

```
Ingrese el usuario: torres
El usuario no existe
```

Figura 3: Usuario incorrecto

```
Ingrese el usuario: angel
Ingresa la contraseña: torres
Validando contraseña, espere un momento..
Contraseña incorrecta
```

Figura 4: Usuario correcto pero contraseña incorrecta

```
Llegó al límite de intentos, reintente más tarde
angel@angel-ubuntu ~/D/TerminalPrebe_PROTECO (main)>
```

Figura 5: Se supera el límite de intentos

4.2. Terminal

Se crea una función que se llamará en caso de que se ingresen los comandos inválidos (Ctrl+Z", Ctrl+X", Ctrl+Cz .exit"), para ello también se utiliza trap, luego se define una variable que ayuda a almacenar la opción que dé el usuario para ejecutar de los comandos. El programa entra a un ciclo en donde estará hasta que se indique una salida, ahí se guarda la ruta absoluta que lleva el programa, así como el usuario; luego de que se muestra la ruta el usuario puede ingresar su comando elegido, en donde dependiendo el comando se invoca al programa que ejecuta dicha función, luego se limpia la variable que almacenaba la instrucción y así en un ciclo hasta que se indique "salir", cuando éste comando sea elegido el programa se termina y

sale de la Terminal Prebe.

Código

```
1  #!/bin/bash
2      exit(){
3          printf "\n"
4          printf "\t\tcomando invalido"
5      }
6      trap ' ' INT SIGINT SIGTSTP SIGTERM
7      stty susp ^O #ignora Ctrl+bandera
8      opcion=""
9
10     while true; do
11         ruta="$(pwd)"
12         printf "\n\n"
13         printf "$USER":~"$ruta"$ "
14         read opcion
15
16         if [ "$opcion" == "ayuda" ]; then
17             ./ayuda.sh
18             opcion=""
19             #break
20         fi
21         if [ "$opcion" == "buscar" ]; then
22             ./buscar.sh
23             opcion=""
24         fi
25         if [ "$opcion" == "creditos" ]; then
26             ./creditos.sh
27             opcion=""
28         fi
29         if [ "$opcion" == "infosis" ]; then
30             ./infosis.sh
31             opcion=""
32         fi
33         if [ "$opcion" == "mp3" ]; then
34             ./mp3.sh
35             opcion=""
36         fi
37         if [ "$opcion" == "fechayhora" ]; then
38             ./fechayhora.sh
39             opcion=""
40         fi
41         if [ "$opcion" == "juego" ]; then
42             ./juego.sh
43             opcion=""
44         fi
45
46         if [ "$opcion" == "salir" ]; then
47             opcion=""
```

```

48         break
49     fi
50     $opcion
51 done

```

Ejecución

```

Nombre del equipo: angel-ubuntu

-----
Bienvenido angel
-----
Comando | Descripcion
-----
ayuda   | Muestra los comandos disponibles y su funcionalidad
buscar  | Busca un archivo en el sistema
creditos | Muestra los nombres de los desarrolladores
infosis | Muestra información del sistema
mp3      | Ejecuta un reproductor de musica
fechayhora | Muestra la fecha y hora
juego    | Muestra y ejecuta el juego programado
salir    | Sale de la terminal prebe

angel:~/home/angel/Documentos/TerminalPrebe_PROTECO$

```

Figura 6: Se imprimen los comandos y aparece la línea de comandos de la terminal

4.3. Comandos

4.3.1. ayuda

Es una tabla en donde se muestra la lista de comandos que el usuario puede usar, así como una breve descripción de los mismos. Código

```

1  #!/bin/bash
2  printf "\n\n"
3  echo -e "Nombre del equipo: $(uname -n)\n"
4  printf "\t\t\tBienvenido $USER \n"
5  echo "
6  "
7  echo "      Comando      |      Descripcion      "
8  echo "
9  "
10 echo "      ayuda      | Muestra los comandos
11      disponibles y su funcionalidad "
12 echo "      buscar     | Busca un archivo en el sistema
13 "
14 echo "      creditos    | Muestra los nombres de los
15      desarrolladores "
16 echo "      infosis     | Muestra informaci n del
17      sistema "

```

```

12 echo "      mp3      | Ejecuta un reproductor de
    musica "
13 echo "      fechayhora | Muestra la fecha y hora "
14 echo "      juego      | Muestra y ejecuta el juego
    programado "
15 echo "      salir      | Sale de la terminal prebe "

```

Ejecución

```

angel:~/home/angel/Documentos/TerminalPrebe_PROTECO$ ayuda

Nombre del equipo: angel-ubuntu

Bienvenido angel
-----
Comando | Descripcion
-----
ayuda   | Muestra los comandos disponibles y su funcionalidad
buscar  | Busca un archivo en el sistema
creditos | Muestra los nombres de los desarrolladores
infosis | Muestra información del sistema
mp3     | Ejecuta un reproductor de musica
fechayhora | Muestra la fecha y hora
juego   | Muestra y ejecuta el juego programado
salir   | Sale de la terminal prebe
angel:~/home/angel/Documentos/TerminalPrebe_PROTECO$ █

```

Figura 7: Se despliega la tabla de comandos existentes

4.3.2. buscar

Primero pide al usuario el nombre del directorio donde quiere buscar el archivo, luego se pide el nombre del archivo, se verifica que el directorio dado exista en el sistema, en caso de no existir se sale de la función, si el directorio existe se invoca a la función que lo busca buscararchivo, quién recorre el directorio y va buscando el archivo, si lo encuentra manda un mensaje de que fue encontrado y sale de la función, en caso contrario envía un mensaje de que no fue encontrado y sale del programa.

Código

```

1 #!/bin/bash
2
3 # Funci n para buscar el archivo en un directorio
4 buscar_archivo()
5 {
6     local directorio="$1"
7     local archivo="$2"
8
9     for elemento in "$directorio"/*; do
10         if [ -f "$elemento" ] && [ "$(basename "$elemento"
            )" = "$archivo" ]; then

```



```

11         printf "\nSe encontr el archivo $archivo en
           $elemento"
12         return 0
13     elif [ -d "$elemento" ]; then
14         buscar_archivo "$elemento" "$archivo"
15     fi
16 done
17
18     return 1
19 }
20
21 # Solicitar al usuario el directorio y el archivo a
  buscar
22 read -p "Por favor, ingrese la ruta absoluta del
  directorio: " directorio
23 printf "\n"
24 read -p "Por favor, ingrese el nombre del archivo a
  buscar: " archivo
25
26 # Verificar si el directorio existe
27 if [ ! -d "$directorio" ]; then
28     printf "\nEl directorio $directorio no existe."
29     exit 1
30 fi
31
32 # Llamar a la funci n para buscar el archivo
33 if buscar_archivo "$directorio" "$archivo"; then
34     exit 0
35 else
36     printf "\nNo se encontr el archivo $archivo en
  $directorio"
37     exit 1
38 fi

```

Ejecución

```

angel:~/home/angel/Documentos/TerminalPrebe_PROTECO$ buscar
Por favor, ingrese la ruta absoluta del directorio: /home/angel/Documentos

Por favor, ingrese el nombre del archivo a buscar: saludo.txt

Se encontró el archivo saludo.txt en /home/angel/Documentos/saludo.txt

```

Figura 8: Se encuentra el archivo deseado en el directorio

```

angel:~/home/angel/Documentos/TerminalPrebe_PROTECO$ buscar
Por favor, ingrese la ruta absoluta del directorio: /home/angel/Documentos

Por favor, ingrese el nombre del archivo a buscar: hola.txt

No se encontró el archivo hola.txt en /home/angel/Documentos

```

Figura 9: No se encuentra el archivo en el directorio

```
angel:~/home/angel/Documentos/TerminalPrebe_PROTECO$ buscar
Por favor, ingrese la ruta absoluta del directorio: /home/angel/Hola

Por favor, ingrese el nombre del archivo a buscar: saludo.txt

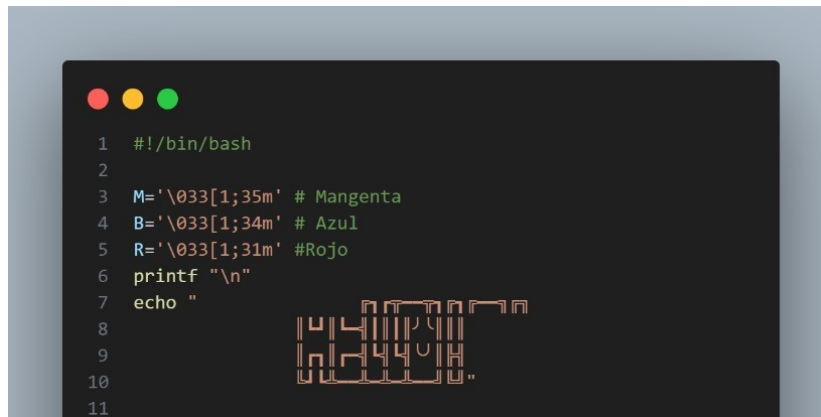
El directorio /home/angel/Hola no existe.
```

Figura 10: No existe el directorio

4.3.3. credits

Se muestra un mensaje de "HELLO" luego los autores del proyecto, finalmente muestra una imagen.

Código



```
1  #!/bin/bash
2
3  M='\033[1;35m' # Magenta
4  B='\033[1;34m' # Azul
5  R='\033[1;31m' #Rojo
6  printf "\n"
7  echo "
8
9
10
11"
11
```

```

12 printf "$R ----- Este proyecto fue realizado por -----"
13 echo ""
14 echo ""
15 printf "\t$B Flores Torres Ángel Joel"
16 printf "\n\n\t$M Medina Rubalcava Natalia Michell\n\n"
17
18 echo "
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41

```



Ejecución

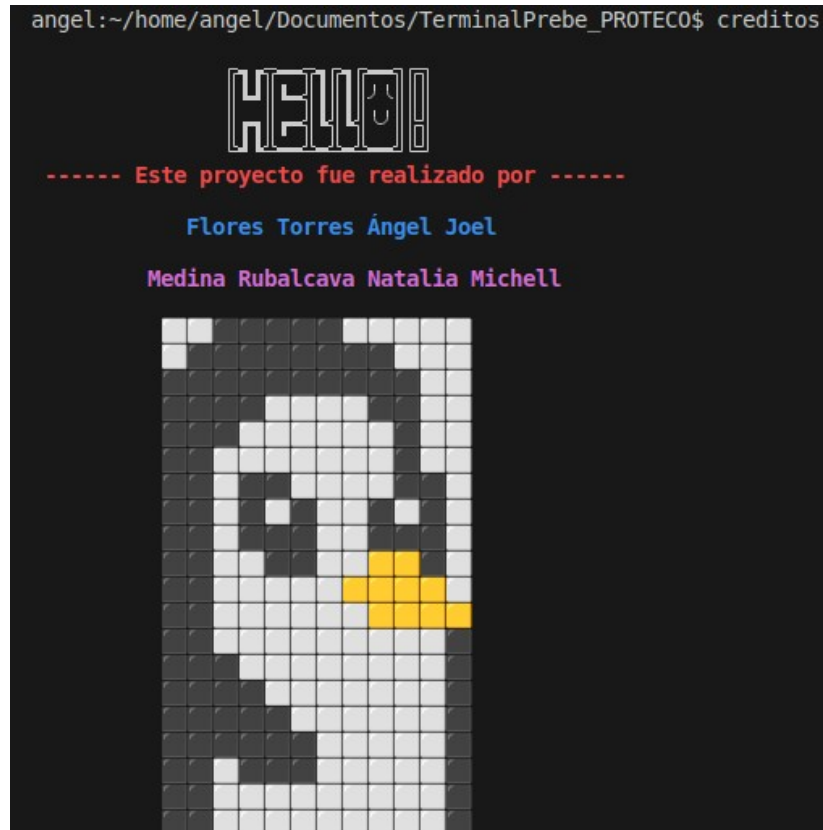


Figura 11: Se muestran los desarrolladores del programa

4.3.4. infosis

Muestra la información del sistema: nombre del equipo, el sistema operativo, la versión del kernel, información de la memoria, la memoria RAM, adicionalmente la arquitectura del sistema, la versión del sistema operativo y el tipo de procesador.

Código

```
1 #!/bin/bash
2 Glig='\e[1;32m' # Verde claro
3 reset="\033[0m"
4
5 printf "\n $Glig ---INFORMACI N DEL SISTEMA---\n
6     $reset"
```

```

7 printf "\t\nNombre del equipo: $(uname -n)\n"
8
9 printf "\t\nSistema operativo: $(uname -o)\n"
10
11 printf "\t\nVersi n del kernel: $(uname -v)\n"
12
13 printf "\t\nInformaci n de la memoria: $(uname -i)\n"
14 "
15 printf "\t\nMemoria RAM: $(free -h)\n"
16
17 printf "\t\nArquitectura del sistema: $(uname -m)\n"
18
19 printf "\t\nVersion del SO: $(uname -r)\n"
20
21 printf "\t\nTipo de procesador: $(uname -p)\n\n"

```

Ejecución

```

angel:~/home/angel/Documentos/TerminalPrebe_PROTECO$ infosis
---INFORMACIÓN DEL SISTEMA---
Nombre del equipo: angel-ubuntu
Sistema operativo: GNU/Linux
Versión del kernel: #33-22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Thu Sep  7 10:33:52 UTC 2
Información de la memoria: x86_64
Memoria RAM:
Mem:      3.7Gi      total      usado      libre compartido  búf/caché  disponible
Inter:    2.6Gi      9.0Mi      2.6Gi      158Mi      1.5Gi      1.2Gi
Arquitectura del sistema: x86_64
Version del SO: 6.2.0-33-generic
Tipo de procesador: x86_64

```

Figura 12: Se muestra la información del sistema

4.3.5. mp3

Primero se verifica que no se pueda salir con los comandos restringidos, luego se almacena en una variable la ruta de Música, se invoca a una función que imprime un logo y se comprueba que el reproductor esté instalado en la computadora, si no está instalado se pregunta si desea instalarlo o no, de ser que sí, sólo se instala con sudo, en caso de que no, se sale del mp3; si ya está instalado regresa e invoca la función para imprimir el menú, en menú muestra de qué manera quieres reproducir una canción, si es específica o aleatoria, o salirse de ahí. Se pude al usuario ingresar su opción y se verifica, dependiendo de está se hará algo diferente. Si se elige reproducir canciones aleatorias se limpia pantalla, imprime logo y se reproducen. En caso de querer una específica, se listan algunas canciones y se pide al usuario que elija la que quiere escuchar. En caso de que se decida salir, va a terminar

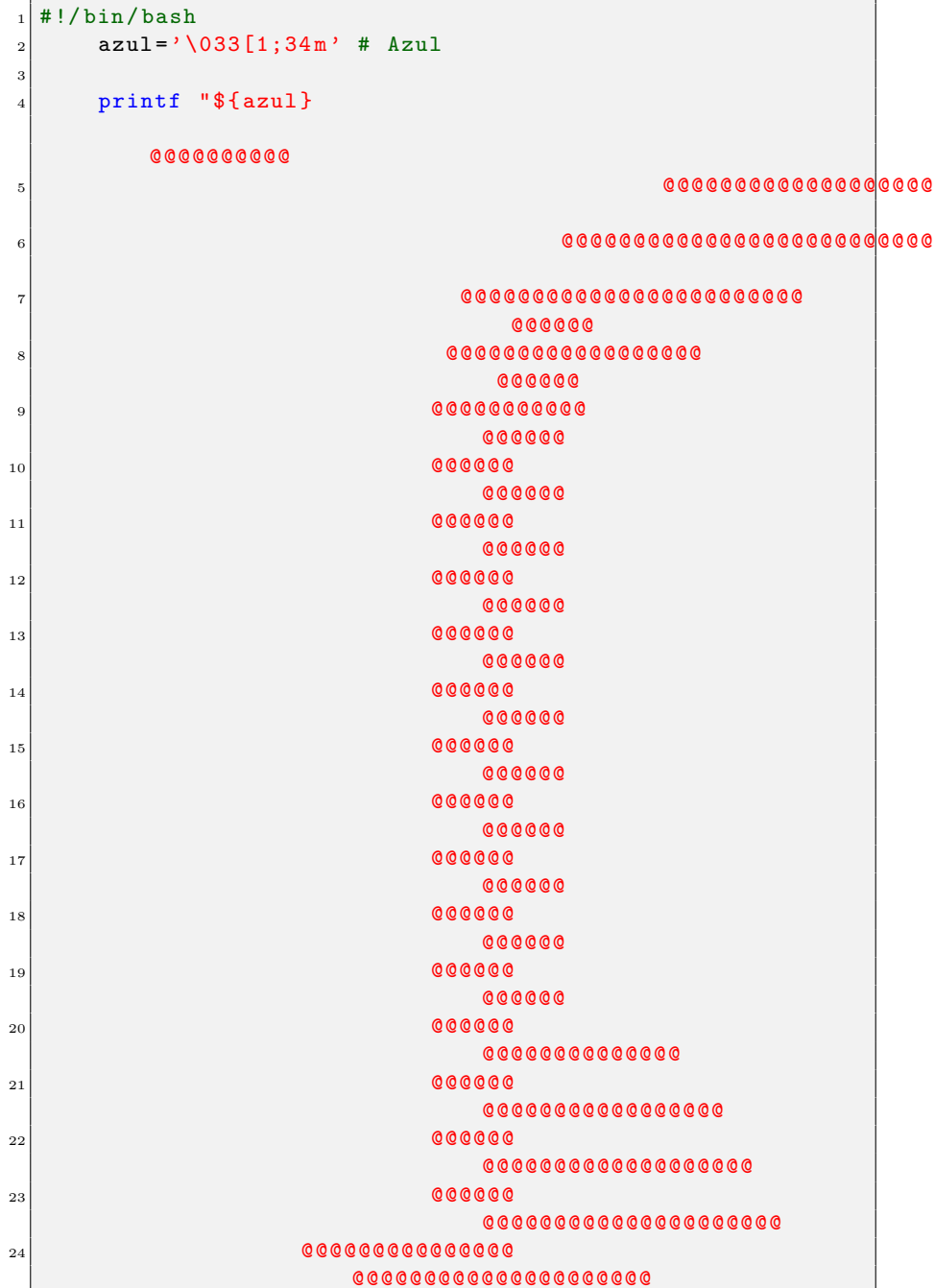
la ejecución de esa parte del programa, si elige alguna otra opción se imprime que no es válida.

Código

```

1  #!/bin/bash
2  azul='\033[1;34m' # Azul
3
4  printf "${azul}
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24

```



The image shows a large graphic of the number 14, constructed from red '0' characters. The graphic is centered on a light gray background. The number 14 is formed by a series of '0's arranged in a way that the vertical stroke of the '1' and the horizontal and curved strokes of the '4' are clearly visible. The '0's are of varying lengths and are arranged in a grid-like pattern to form the digits. The number 14 is the largest and most prominent element in the image.

```

25      @@@@@@@@@@@@@@@@@@
26      @@@@@@@@@@@@@@@@@@
27      @@@@@@@@@@@@@@@@@@
28      @@@@@@@@@@@@@@@@@@
29      @@@@@@@@@@@@@@@@@@
30      @@@@@@@@@@@@@@@@@@
31      @@@@@@@@@@@@@@@@@@
32      @@@@@@@@@@@@@@@@@@
33      @@@@@@@@@@@@@@@@@@
34
35
36
37      R='\033[1;31m' #Rojo
38      G='\033[1;32m' # Verde
39      B='\033[1;34m' # Azul
40      Y='\033[1;33m' # Amarillo
41      W='\033[0m' # Blanco
42      M='\033[1;35m' # Mangenta
43      reset="\033[0m"
44
45      imprimir_logo()
46      {
47          clear
48          ./logo.sh
49      }
50
51      comprobarmpg123() # Funci n que comprueba si ya
52      est instalado el reproductor
53      {
54          carpetampg123=/bin/mpg123 # Ac se descarga la
55          carpeta
56
57          if [ -f "$carpetampg123" ] # Comprobamos que
58          exista el reporductor
59          then
60              printf "" # Ya existe
61          else
62              printf "\n\n\t\t\t$R A D V E R T E N C I
63              A$W \n\n\t\t\t No se encuentra
64              instalado 'mpg123' \n"
65              printf "\n\t
66              -----
67              "
68              printf "$Glig\n\n\t\t\t Desea instalar
69              mpg123? [s/n]: $W"

```

```

62         read opcion
63         case $opcion in
64             's') # Queremos instalar
65                 sudo apt install mpg123
66                 ;;
67             'n') # No queremos instalar
68                 exit 0
69                 ;;
70             *)
71                 printf "Opci n no valida
72                     "
73                 esac
74         fi
75     }
76     imprimirMenu()
77     {
78         printf "\n\n\t\t$W
79         -----"
80         printf "\n\n\t\t\tM E N \n\t\t$W
81         ----- \n"
82         printf "\n\t\t\t1) Reproducir canciones de forma
83         aleatoria (de la carpeta por defecto)"
84         printf "\n\t\t\t2) Reproducir una canci n "
85         printf "\n\t\t\t3) Salir \n"
86         printf "\n\t\t\tCarpeta por defecto: $Y $HOME/
87         M sica$reset"
88         printf "\n\t\t$W
89         -----"
90         "
91     }
92
93     leerOpcionMenu()
94     {
95         printf "\n\n\t\t\t$G Qu opci n desea?:$W "
96         read opcion
97     }
98
99     varias() # Para canciones en lista
100    {
101        len=$(ls "${lugarMusica}"/*.mp3 2>/dev/null | wc
102            -l)
103
104        if [ "$len" -eq 0 ]; then
105            printf "\n\t\t\t$W
106            ----- \n"
107            printf "\n\t\t\t$G No hay canciones en la
108            carpeta.\n"
109            printf "\t\t\t$G Intente agregando
110            canciones (archivos .mp3) a la

```



```

101         carpeta: "$HOME/M sica"\n"
102         printf "\t\t $G Presiona Enter para
103             volver al men principal."
104         read
105         return
106         fi
107     printf "\n\n\t\t\t$B O P C I O N E S $W"
108     printf "\n\t\t\t ----- \n"
109     printf "\n\t\t\t d) Anterior f) Siguiente \n"
110     printf "\n\t\t\t s) Pausa/Play\t"
111     printf "\n\t\t\t u) Silenciar\t"
112     printf "\n\t\t\t l) Canci n actual\t"
113     printf "\n\t\t\t q) Salir \n"
114     printf "\t\t\t ----- \n"
115     printf "\n\t\t\t -) Bajar volumen +) Subir
116         Volumen \n"
117     printf "\n\t\t\t ----- \n"
118     "
119 }
120
121 individual() # Para canciones individuales
122 {
123     printf "\n\n\t\t\t$B O P C I O N E S $W"
124     printf "\n\t\t\t ----- \n"
125     printf "\n\t\t\t s) Pausa/Play\t"
126     printf "\n\t\t\t u) Silenciar\t"
127     printf "\n\t\t\t l) Canci n actual\t"
128     printf "\n\t\t\t q) Salir \n"
129     printf "\t\t\t ----- \n"
130     printf "\n\t\t\t -) Bajar volumen +) Subir
131         Volumen \n"
132     printf "\n\t\t\t ----- \n"
133     "
134 }
135
136 hayCanciones() # Funci n para verificar si hay
137     archivos .mp3 en la carpeta
138 {
139     len=$(ls "${lugarMusica}"/*.mp3 2>/dev/null | wc
140         -1)
141     if [ "$len" -eq 0 ]; then
142         return 1 # No hay canciones
143     else
144         return 0 # Hay canciones
145     fi
146 }

```

```

141 listarCanciones() # Funci n para listar canciones
142 {
143     listaCacniones='ls /home/$USER/M sica ' #
144         Guardamos el contenido de esa carpeta
145     printf "\n\n\t    $Y C A N C I O N E S    D I S P
        O N I B L E S $W"
146     printf "\n\t\t ----- \n"
147     for cancion in ${listaCacniones[*]}
148     do
149         printf "\n\t\t- $cancion"
150     done
151     printf "\n"
152     printf "\n\t
        -----
        "
153 }
154
155 elegirCancionFav() # Funci n para elegir una
        cacni n por su nombre
156 {
157     #listarCanciones
158     printf "\n\n$G Copia y pega el nombre de la
        caci n a reproducir de la lista (sin
        espacios):$W\n"
159     read -p " >>" cancionFav
160     #cd .. # Para que no hay problemas en la
        reproducci n
161 }
162
163 main()
164 {
165     trap ' ' INT SIGINT SIGTSTP SIGTERM
166     opcion=0
167     lugarMusica="$HOME/M sica" # Obtenemos la ruta
        de la m sica
168     imprimir_logo
169     comprobarmpg123
170
171     while [ "$opcion" != 4 ]
172     do
173         clear
174         imprimir_logo
175         imprimirMenu
176         leerOpcionMenu
177
178         case $opcion in
179             1)
180                 Canniones aleatoreas
                    clear

```

```

181         imprimir_logo
182         varias
183         mpg123 -C --title -q -z "
184             ${lugarMusica}"/*
185     ;;
186 2)                                     #
187     Canci n espec fica
188     clear
189     imprimir_logo
190     if hayCanciones; then
191         listarCanciones
192         elegirCancionFav
193     else
194         printf "\n\t\t $G
195             No hay
196             canciones en
197             la carpeta.\n"
198         printf "\t\t $G
199             Intente
200             agregando
201             canciones (
202             archivos .mp3
203             ) a la
204             carpeta: "
205             $HOME/M sica
206             "\n"
207         printf "\t\t $G
208             Presiona
209             Enter para
210             volver al
211             men
212             principal."
213         read
214         fi
215     clear
216     imprimir_logo
217     individual
218     mpg123 -C --title -q -z "
219         ${lugarMusica}"/"
220         $cancionFav"
221     ;;
222 3)                                     #
223     Salir
224     clear

```

```

205         exit 0
206     ;;
207 *)                                     #
208     Opci n no v lida
209     clear
210     imprimir_logo
211     imprimirMenu
212     printf "\n\n\t\t $M[
213         Advertencia]$W
214         Opci n no v lida.\n
215         "
216     read
217     ;;
218 esac
219 done
220 }
221 main

```

Ejecución

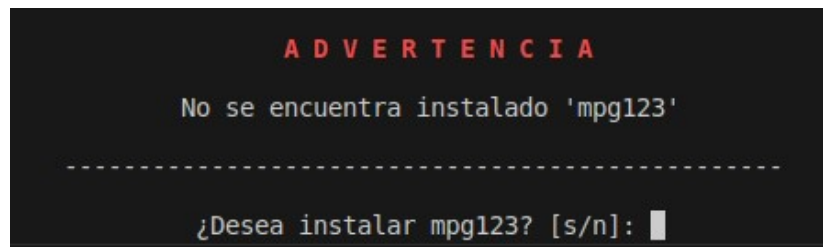


Figura 13: Se verifica si se tiene instalado el programa mpg123

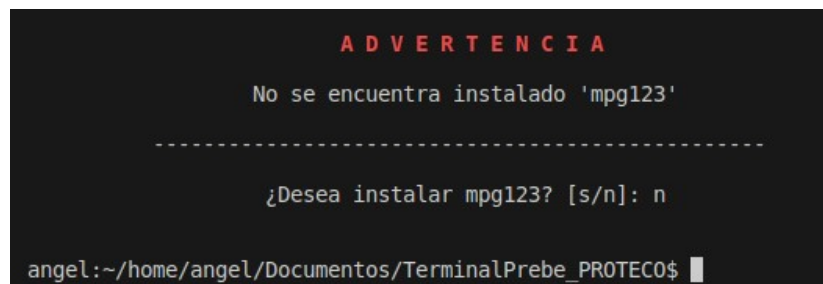


Figura 14: Si no se acepta la instalación, se cierra el mp3 y se regresa a la línea de comandos

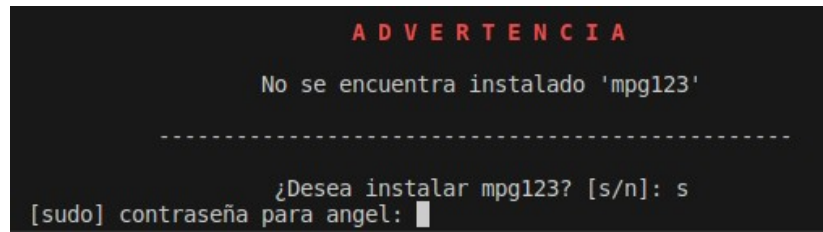


Figura 15: Si se acepta la instalación, se le pide al usuario su contraseña y empieza la instalación



Figura 16: Si ya se instalo o se cuenta con el programa desde un inicio se despliega el menu

```
No hay canciones en la carpeta.  
Intente agregando canciones (archivos .mp3) a la carpeta: /home/angel/Música  
Presiona Enter para volver al menú principal.
```

Figura 17: Si no hay música(archivos .mp3) en la carpeta, arroja un error en ambas opciones

```
          O P C I O N E S  
-----  
d) Anterior    f) Siguiete  
  
s) Pausa/Play  
u) Silenciar  
l) Canción actual  
q) Salir  
-----  
-) Bajar volumen  +) Subir Volumen  
-----
```

Figura 18: Opcion 1: Se reproduce musica aleatoria, para ello se despliega un menu de controles

```
          C A N C I O N E S   D I S P O N I B L E S  
-----  
- BandalosChinos_Demasiado.mp3  
- BandalosChinos_MiFiesta.mp3  
- DMEM_LoHice,Tedejé.mp3  
- Esteman_FuimosAmor.mp3  
- Georgel_Miranos.mp3  
-----  
Copia y pega el nombre de la canción a reproducir de la lista (sin espacios):  
>>
```

Figura 19: Opcion 2: Se reproduce una canción específica, para ello se despliega una lista con las canciones disponibles, si no se respeta el formato o se deja en blanco, al dar se regresa al menu principal

```

                                O P C I O N E S
                                -----
                                s) Pausa/Play
                                u) Silenciar
                                l) Canción actual
                                q) Salir
                                -----
                                -) Bajar volumen  +) Subir Volumen
                                -----
                                Playlist (">" indicates current track):
                                > /home/angel/Música/Esteman_FuimosAmor.mp3

```

Figura 20: Opcion 2: Si se respeta el formato se reproduce la canción seleccionada y se despliega la lista de controles

4.3.6. fechayhora

Se almacena en una variable el lugar, la fecha y la hora. No se usa "date" para mostrarla y tiene un formato específico del lugar, día y hora.

Código

```

1  #!/bin/bash
2
3  fecha_hora=$(TZ="America/Mexico_City" awk 'BEGIN {
4      split(strftime("%d %m %Y %H %M %S"), time); print
5      time[1] "-" time[2] "-" time[3] " " time[4] ":"
6      time[5] ":" time[6]}' )
7
8  printf "\n"
9  echo -e "Fecha y hora actual: $fecha_hora \n"

```

Ejecución

```

angel:~/home/angel/Documentos/TerminalPrebe_PROTECO$ fechayhora
Fecha y hora actual: 22-09-2023 21:12:15

```

Figura 21: Se muestra la fecha y la hora actual

4.3.7. juego

El juego es un ahorcado en donde primero se solicita la palabra a adivinar, se convierten todas las letras en minúsculas y se verifica que todos los caracteres sean letras, de no ser así se puede ingresarla nuevamente. Luego se saca la longitud de la palabra, se limpia la

pantalla y se da un máximo de intentos de 7, entonces entra a un ciclo en donde se invoca la función de mostrar ahorcado y dependiendo si ha cambiado el número de intentos se empieza a dibujar el muñeco del ahorcado, se da la longitud de la palabra y se invoca a la función que va mostrando la palabra oculta, según como se van adivinando las letras. En caso de que se diga teniendo intentos se va a seguir pidiendo al jugador que ingresé una letra y está se va a validar para que sea una sola letra, que sea válida y que no sea una letra repetida. Si se llega al límite de intentos se mostrará que perdió y cuál era la palabra oculta. En caso de que adivine toda la palabra se muestra un mensaje de que ganó el juego.

Código

```

1  #!/bin/bash
2
3  # Funci n para mostrar el dibujo del ahorcado
4  mostrar_ahorcado()
5  {
6      case $intentos in
7          7) echo "  _ _ _ _ "
8              echo " |         | "
9              echo " |         "
10             echo " |         "
11             echo " |         "
12             echo " |         "
13             echo " _| _ _ _ _ "
14             ;;
15          6) echo "  _ _ _ _ "
16              echo " |         | "
17              echo " |         0 "
18              echo " |         "
19              echo " |         "
20              echo " |         "
21              echo " _| _ _ _ _ "
22              ;;
23          5) echo "  _ _ _ _ "
24              echo " |         | "
25              echo " |         0 "
26              echo " |         | "
27              echo " |         "
28              echo " |         "
29              echo " _| _ _ _ _ "
30              ;;
31          4) echo "  _ _ _ _ "
32              echo " |         | "
33              echo " |         0 "
34              echo " |         / | "
35              echo " |         "
36              echo " |         "

```



```

37     echo "_|-----"
38     ;;
39     3) echo "  ----"
40     echo " |      |"
41     echo " |      0"
42     echo " |      /|\\"
43     echo " |"
44     echo " |"
45     echo "_|-----"
46     ;;
47     2) echo "  ----"
48     echo " |      |"
49     echo " |      0"
50     echo " |      /|\\"
51     echo " |      /"
52     echo " |"
53     echo "_|-----"
54     ;;
55     1) echo "  ----"
56     echo " |      |"
57     echo " |      0"
58     echo " |      /|\\"
59     echo " |      / \\"
60     echo " |"
61     echo "_|-----"
62     ;;
63     *) echo ""
64     echo " |"
65     echo " |"
66     echo " |"
67     echo " |"
68     echo "_|-----"
69     ;;
70     esac
71 }
72
73 # Funci n para mostrar la palabra oculta con letras
74   adivinadas
75 mostrar_palabra_oculta()
76 {
77     palabra_mostrada=""
78     i=1
79     while [ $i -le ${#palabra} ]; do
80         letra_actual=$(echo "$palabra" | cut -c$i)
81         if echo "$adivinadas" | grep -q "
82             $letra_actual"; then
83             palabra_mostrada="$palabra_mostrada$letra_actual"
84         else
85             palabra_mostrada="${palabra_mostrada}_ "

```

```

84         fi
85         i=$((i+1))
86     done
87     echo "$palabra_mostrada"
88 }
89
90 # Solicitar una palabra al usuario
91 clear
92 printf "\n"
93 echo -n "Ingresa una palabra para adivinar: "
94 read palabra
95
96 # Convertir la palabra a min sculas
97 palabra=$(echo "$palabra" | tr '[:upper:]' '[:lower:]')
98
99 # Verificar si la palabra contiene caracteres que no
   sean letras
100 if [[ ! "$palabra" =~ ^[a-zA-Z]+$ ]]; then
101     echo "La palabra debe contener solo letras."
   Int ntalo de nuevo."
102     exit 1
103 fi
104
105 # Obtener la longitud de la palabra
106 longitud_palabra=${#palabra}
107
108 # Limpiar pantalla
109 clear
110
111 # Inicializar variables
112 intentos=7
113 adivinadas=""
114 fin_del_juego=0
115
116 # Ciclo principal del juego
117 while [ $fin_del_juego -eq 0 ]; do
118     clear
119     mostrar_ahorcado
120     echo
121     echo "Longitud de la palabra: $longitud_palabra"
122     echo -e
123     echo "Palabra: $(mostrar_palabra_oculta)"
124
125     # Verificar si se ha perdido
126     if [ $intentos -eq 0 ]; then
127         echo -e
128         echo " Perdiste !"
129         echo "La palabra era: $palabra"
130         fin_del_juego=1

```

```

131         break
132     fi
133
134     # Solicitar una letra al jugador
135     echo -e
136     echo -n "Ingresa una letra: "
137     read letra
138     letra=$(echo "$letra" | tr '[:upper:]' '[:lower:]'
139         ')
140
141     # Verificar si la letra tiene una longitud mayor
142     # que 1
143     if [ ${#letra} -ne 1 ]; then
144         echo -e
145         echo "Por favor, ingresa solo una letra a la
146             vez."
147         echo "Presiona enter para continuar..."
148         read _
149     else
150         # Verificar si la letra es una letra v lida
151         if [[ ! "$letra" =~ ^[a-zA-Z]$ ]]; then
152             echo -e
153             echo "Ingresa solo letras."
154             echo "Presiona enter para
155                 continuar..."
156             read _
157         else
158             # Verificar si la letra ya se adivin o si
159             # ya se ha intentado antes
160             letra_ya_adivinada=0
161             for letra_adivinada in $(echo $adivinadas |
162                 fold -w1); do
163                 if [ "$letra_adivinada" = "$letra" ];
164                 then
165                     letra_ya_adivinada=1
166                     break
167                 fi
168             done
169
170             if [ $letra_ya_adivinada -eq 1 ]; then
171                 echo -e
172                 echo "Ya adivinaste esa letra o ya la
173                     intentaste antes. Intenta con
174                     otra."
175                 echo "Presiona enter para continuar
176                     ..."
177                 read _ # Espera a que el usuario
178                     presione Enter
179             else
180                 # Verificar si la letra est en la

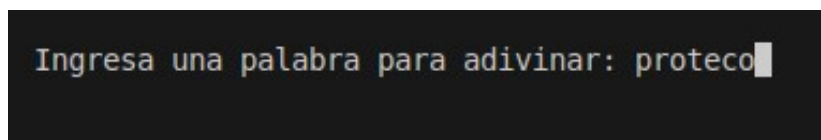
```

```

170         palabra
171     if echo "$palabra" | grep -q "$letra"
172     ; then
173         adivinadas="$adivinadas$letra"
174     else
175         intentos=$((intentos - 1))
176         echo
177         echo "Letra incorrecta. Te quedan
178             $intentos vidas."
179         echo "Presiona enter para
180             continuar..."
181         read -
182     fi
183
184     # Verificar si se ha ganado
185     palabra_mostrada=$(
186         mostrar_palabra_oculta)
187     if [ "$(echo "$palabra_mostrada" | tr
188         -d ' ') = "$palabra" ]; then
189         clear
190         mostrar_ahorcado
191         echo "Palabra: $palabra_mostrada"
192         echo -e
193         echo "  Ganaste !"
194         fin_del_juego=1
195         break
196     fi
197 fi
198 fi
199 fi
200 done

```

Ejecución



Ingresa una palabra para adivinar: proteco

Figura 22: Cuando se ejecuta se le pide al usuario la palabra a adivinar

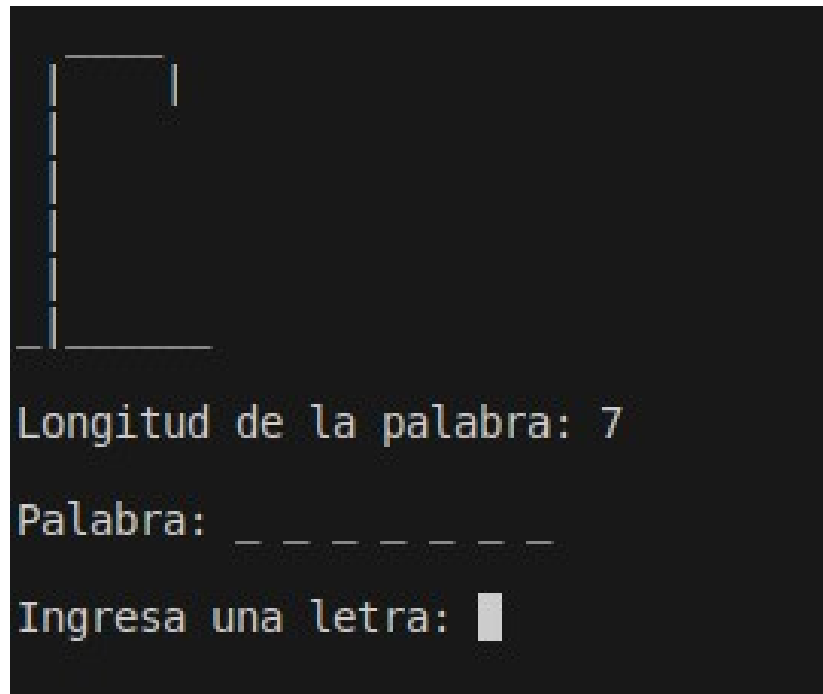


Figura 23: Se muestra el tablero, y se le pide al jugador ingresar las letras

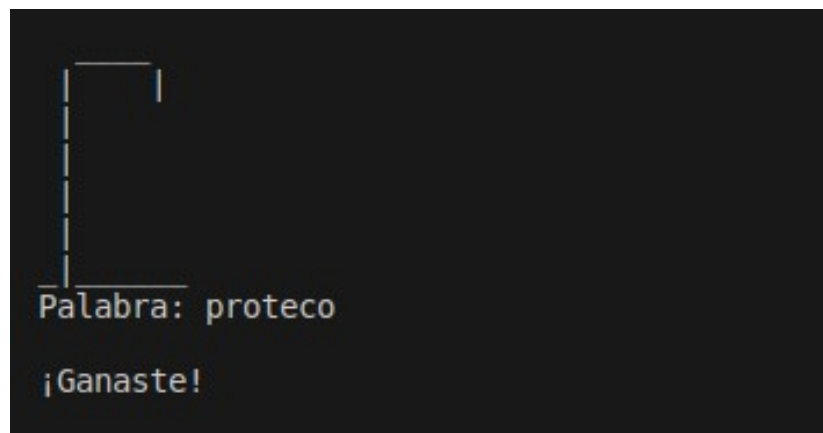


Figura 24: Mensaje que se muestra si de adivina la palabra

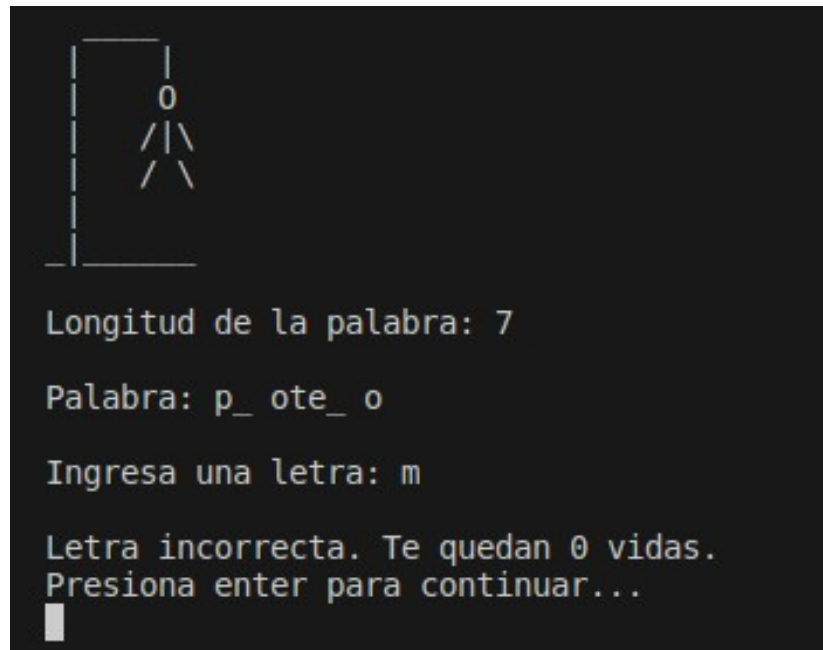


Figura 25: Se muestra la cantidad de intentos que le quedan

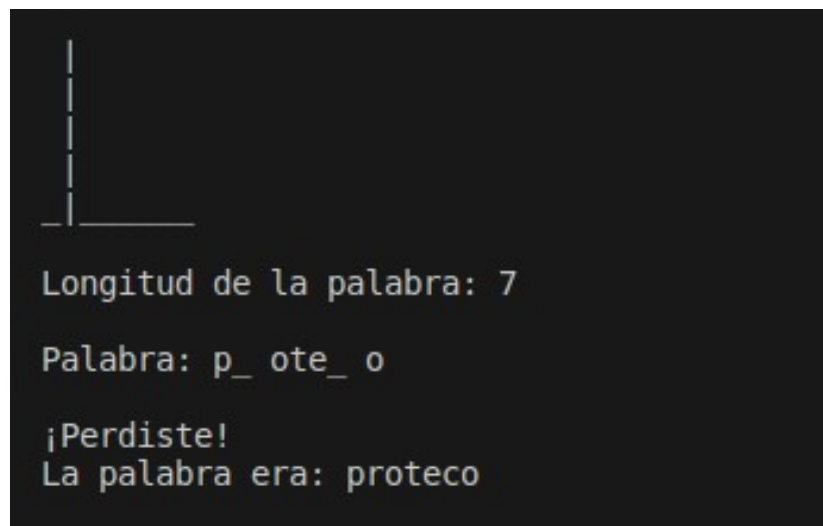


Figura 26: Mensaje que se muestra si no se adivina la palabra y se sale a la línea de comandos

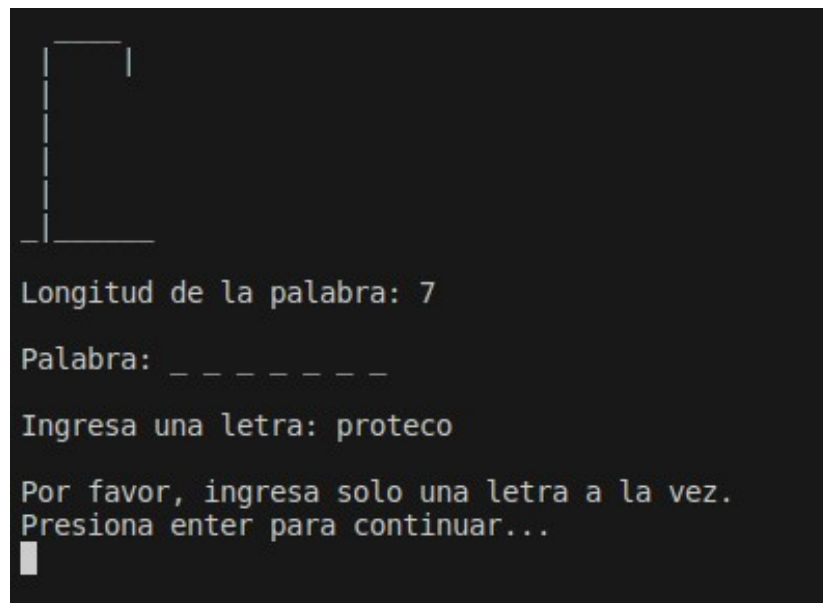


Figura 27: Solo se admite ingresar una letra a la vez

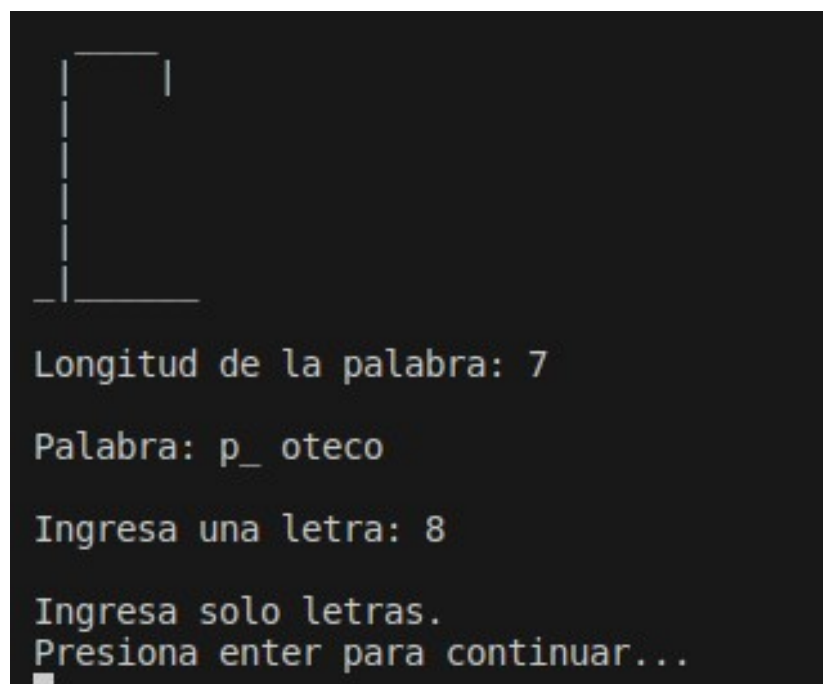


Figura 28: Solo se admiten letras, nada de caracteres de otro tipo

4.4. Conclusiones

Ángel: La realización de este proyecto me ayudo considerablemente a mejorar mis habilidades con el uso de comandos en Linux y la programación de Shell scripts, principalmente porque una vez que termine el curso de GNU-Linux hace unas semanas, deje de practicar, y retomarlo otra vez me pareció algo gratificante. Este proyecto me resulto bastante entretenido y muy bueno, pues a pesar de las pequeñas dificultades que a veces se presentaban fue justo eso lo que me orillaba a investigar distintas soluciones a un problema específico. A pesar de que considero que no le dedique el tiempo suficiente que yo hubiera querido, estoy muy satisfecho con el resultado, incluso estoy motivado a terminarlo o mejorarlo más en algún punto.

Natalia: A lo largo del desarrollo del proyecto pudimos observar que todo lo visto en clase fue de gran utilidad, ya que al aplicarlo se facilitó el uso de algunas funciones, fue un poco pesado pero de buen nivel.