

UNIVERSIDAD DE COSTA RICA

ESCUELA DE INGENIERÍA ELÉCTRICA

IE0117: PROGRAMACIÓN BAJO PLATAFORMAS ABIERTAS

Reporte

Laboratorio 1

Prof. Carolina Trejos Quirós

Estudiante: José Armando León Solano, B94249

4 de septiembre de 2024

Índice

1. Introducción	2
2. Implementación	2
2.1. Parte 1	2
2.2. Parte 2	2
3. Resultados	3
3.1. Parte 1	3
3.2. Parte 2	3
4. Conclusiones y recomendaciones	4
4.1. Parte 1	4
4.2. Parte 2	6
Referencias	7

1. Introducción

El primer laboratorio, consiste en hacer un primer script en bash que reciba como parámetro la dirección de un archivo e imprima en el stdout los permisos del usuario, grupo y otros y luego hacer un segundo script que cree un usuario y grupo, además de limitar los permisos de ejecución para el mismo archivo utilizado anteriormente.

2. Implementación

2.1. Parte 1

Para resolver esta sección se siguieron las recomendaciones dadas en las instrucciones, luego haciendo uso del operador if y de las condiciones respectivas para archivos se completó la parte a y b. Seguidamente para la segunda parte se utilizó el comando sugerido y el resultado de este se le asignó a una variable. Finalmente para la función llamada `get_permissions_verbose`, se tomó la variable con los permisos del archivo y se separó sus caracteres en 3 variables, luego con el operador `case` se buscó el patrón `r,w`, y `x` que corresponde a los permisos.

2.2. Parte 2

Para la creación de este script, primero es necesario comprobar que el usuario ingrese correctamente un usuario y grupo, por lo que se cuenta el número de variables que ingresa al script vs al esperado, luego se necesita un comando que verifique que en el sistema los usuarios vs los ingresados, luego se pueden usar comandos más directos con permisos de administrador para realizar las acciones solicitadas.

3. Resultados

3.1. Parte 1

```
jose@jose-vb:~/Plataformas$ ./ejercicio1.sh
Ingrese la ruta del archivo, ejemplo: ./ejercicio1.sh /home/user/Documents/file
```

Figura 1: Ejecución errónea del script [1].

En la figura 1, se muestra el resultado de ejecutar el script sin ingresar una dirección para un archivo, junto con mensaje de ayuda.

```
jose@jose-vb:~/Plataformas$ ./ejercicio1.sh /home/jose/Plataformas/ejercicio100.sh
El archivo ejercicio100.sh no existe, errorcode 404
```

Figura 2: Ejemplo de un archivo que no existe[1].

En la figura 2 se muestra que sucede si el usuario ingresar la dirección de un archivo inexistente.

```
jose@jose-vb:~/Plataformas$ ./ejercicio1.sh /home/jose/Plataformas/ejercicio1.sh
El archivo ejercicio1.sh existe.
Permisos del usuario:
- Lectura
- Escritura
- Ejecución
Permisos del grupo:
- Lectura
- Escritura
- Ejecución
Permisos de otros:
- Lectura
- Sin escritura
- Ejecución
```

Figura 3: Ejemplo que muestra los permisos del archivo [1].

Finalmente en figura 3 se muestra el resultado de ingresar un archivo que existe. Se muestran permisos respectivos para el usuario, el grupo, y otros.

3.2. Parte 2

```
jose@jose-vb:~/Plataformas$ ls -l
total 8
-rwxrwxr-x 1 jose jose 2935 Sep  3 16:49 ejercicio1.sh
-rwxrwxr-x 1 jose jose 2088 Sep  3 20:58 ejercicio2.sh
```

Figura 4: Permisos de los scripts antes de ser ejecutados [1].

En la figura 4, se muestran los permisos que tienen los archivos antes de ejecutar el script ejercicio2.sh.

```
jose@jose-vb:~/Plataformas$ ./ejercicio2.sh
Ingrese un usuario y grupo, ejemplo: ./ejercicio2.sh Juan Estudiantes
```

Figura 5: Error al ejecutar el script [1].

En la figura 5 se muestra que sucede si el usuario no ingresa correctamente los parámetros necesarios para ejecutar el script, además de un ejemplo de guía.

```
jose@jose-vb:~/Plataformas$ ./ejercicio2.sh Cesar Chefs
[sudo] password for jose:
Usuario 'Cesar' ha sido creado.
Grupo 'Chefs' ha sido creado.
Los usuarios 'Cesar' y 'jose' fueron agregados al grupo 'Chefs'.
```

Figura 6: Salida en el stdout al ejecutar el script [1].

La figura 6 se puede ver el resultado de ejecutar el script, el cual pide permisos de administrador para ejecutar los comandos y se puede ver que el usuario y grupo a sido creado exitosamente.

```
jose@jose-vb:~/Plataformas$ ls -l
total 8
----r-x--- 1 jose Chefs 2935 Sep  3 16:49 ejercicio1.sh
-rwxrwxr-x 1 jose jose  2088 Sep  3 20:58 ejercicio2.sh
```

Figura 7: Cambio en los permisos de ejecución [1].

Finalmente la figura 7 muestra el cambio de los permisos y grupo dueño del archivo, para que solo miembros del grupo creado puedan ejecutarlo.

4. Conclusiones y recomendaciones

4.1. Parte 1

En la función `get_permissions_verbose` se pudo mejorar implementado un operador `for` para recorrer los caracteres y variables a comprar para evitar el uso repetido del operador `case`.

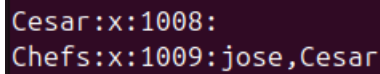
```
1
2 # Funci n para interpretar y obtener permisos detallados
3 get_permissions_verbose() {
4     local permisos="$1"
5
6     # Extraer los caracteres que simbolizan los permisos para usuario, grupo y
    otros
7     local usuario="${permisos:1:3}"
8     local grupo="${permisos:4:3}"
9     local otros="${permisos:7:3}"
10
```

```
11     echo "Permisos del usuario:"
12
13     # Utilizando el operador case en se compara el valor de "leeido" con los
14     # caracteres rwx, si el valor es el mismo, se imprime el nombre de permiso, de lo
15     # contrario se idica que no tiene permiso
16     case ${usuario:0:1} in
17         r) echo "    - Lectura" ;;
18         -) echo "    - Sin lectura" ;;
19     esac
20     case ${usuario:1:1} in
21         w) echo "    - Escritura" ;;
22         -) echo "    - Sin escritura" ;;
23     esac
24     case ${usuario:2:1} in
25         x) echo "    - Ejecuci n" ;;
26         -) echo "    - Sin ejecuci n" ;;
27     esac
28
29 \begin{lstlisting}[style=mystyle, caption={Funci n get\_permissions\_verbose},
30 label=lst:]
31
32     echo "Permisos del grupo:"
33     case ${grupo:0:1} in
34         r) echo "    - Lectura" ;;
35         -) echo "    - Sin lectura" ;;
36     esac
37     case ${grupo:1:1} in
38         w) echo "    - Escritura" ;;
39         *) echo "    - Sin escritura" ;;
40     esac
41     case ${grupo:2:1} in
42         x) echo "    - Ejecuci n" ;;
43         -) echo "    - Sin ejecuci n" ;;
44     esac
45
46     echo "Permisos de otros:"
47     case ${otros:0:1} in
48         r) echo "    - Lectura" ;;
49         -) echo "    - Sin lectura" ;;
50     esac
51     case ${otros:1:1} in
52         w) echo "    - Escritura" ;;
53         -) echo "    - Sin escritura" ;;
54     esac
55     case ${otros:2:1} in
56         x) echo "    - Ejecuci n" ;;
57         -) echo "    - Sin ejecuci n" ;;
58     esac
59 }
```

Listing 1: Función get_permissions_verbose

4.2. Parte 2

Como se puede ver en las figuras respectivas a la parte 2, se muestra que los resultados de ejecutar el script, se puede ver que todo funciona correctamente. Una posible mejora al código sería mostrarle al usuario con el comando - getent group y getent passwd que efectivamente el usuario y grupo fueron creados, tal y como se puede ver en la figura 8



```
Cesar:x:1008:  
Chefs:x:1009:jose,Cesar
```

Figura 8: Prueba de que usuario y grupo fueron creados^[1].

Referencias

1. L. S. Armando, “Imágenes del autor,”