



UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE

SISTEMAS OPERATIVOS

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

Ejecución de Comandos y Scripts

Estudiante:

Ednan Josué Merino Calderón

Docente:

Ing. Washington Loza

Índice general

1. Objetivos	3
1.1. Objetivo General	3
1.2. Objetivos Específicos	3
2. Desarrollo	4
2.1. Estructura de directorios y archivos	4
2.1.1. Crear un directorio principal	4
2.1.2. Creación de subdirectorios para almacenar entradas, salidas y archivos de configuración	4
2.1.3. Crear al menos un archivo vacío en cada subdirectorio	5
2.2. Gestión de usuarios y grupos	6
2.2.1. Crear un grupo de usuarios específico para el proyecto	6
2.2.2. Agregar dos usuarios al sistema y asignarlos al grupo creado	6
2.2.3. Configurar permisos de acceso a los directorios	6
2.3. Creación del script	7
2.4. Configuración y permisos	8
2.4.1. Asegurarse de que el script tenga permisos de ejecución.	8
2.4.2. Configurar los permisos de los archivos y directorios	8
2.5. Validación de acceso y pruebas	9
2.5.1. Cambiar al usuario creado y verificar que pueda ejecutar el script y acceder a los directorios del proyecto	9
2.5.2. Intentar acceder a los directorios como un usuario que no per- tenezca al grupo y comprobar que el acceso está restringido	10
3. Conclusiones	11

Índice de figuras

2.1. Creación de Directorio Principal	4
2.2. Creación de Subdirectorios	5
2.3. Creación de archivos para simular datos iniciales	5
2.4. Creación de un Grupo de usuarios	6
2.5. Creación de usuarios	6
2.6. Permisos de acceso a los directorios	7
2.7. Script gestor.sh	8
2.8. Permisos de Ejecución	8
2.9. Permisos de Archivos y Directorios	9
2.10. Ejecución de Script con usuario con permisos	9
2.11. Script Ejecutado	9
2.12. Permiso denegado	10

1. Objetivos

1.1. Objetivo General

Desarrollar la capacidad de ejecutar comandos y crear scripts básicos en un sistema Linux.

1.2. Objetivos Específicos

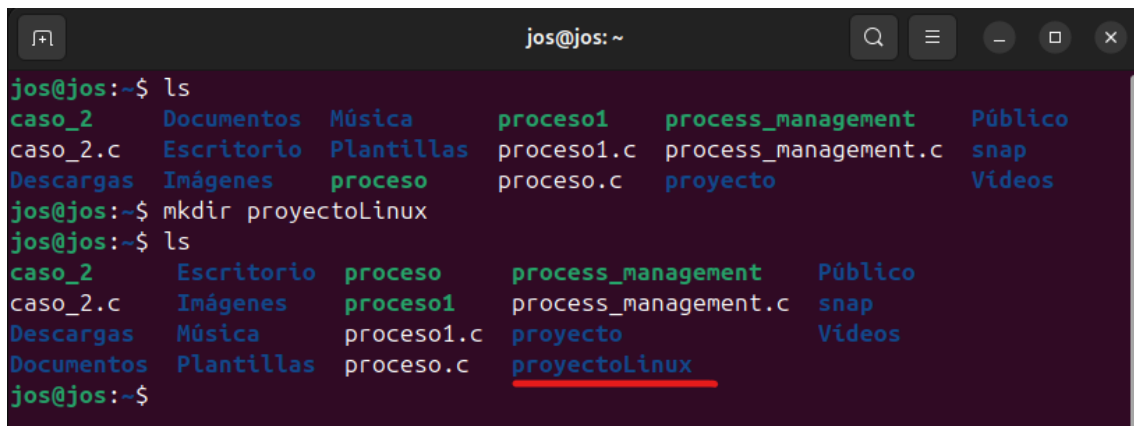
- Utilizar herramientas disponibles en el sistema operativo para realizar tareas de gestión de archivos, directorios, usuarios, y grupos, así como automatizar procesos mediante scripts.
- Configurar permisos de acceso a directorios y archivos para garantizar la seguridad y control sobre la manipulación de datos en un entorno multiusuario.

2. Desarrollo

2.1. Estructura de directorios y archivos

2.1.1. Crear un directorio principal

Para organizar el proyecto, mediante el comando **mkdir** se crea el directorio principal llamado **proyectoLinux**, se evidencia la creación después de utilizar el comando **ls**:

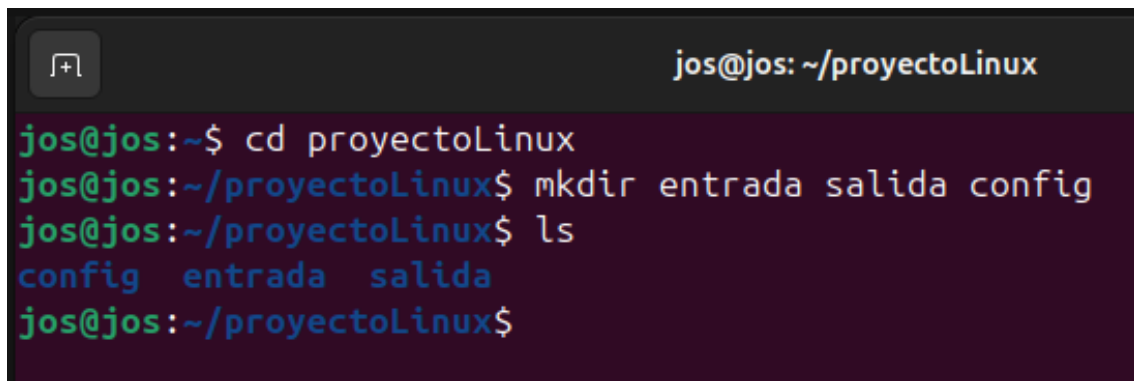
A terminal window titled 'jos@jos: ~' with standard window controls. It shows the execution of 'ls' and 'mkdir proyectoLinux' commands. The 'ls' output lists various files and directories in a color-coded format. After running 'mkdir proyectoLinux', the 'ls' command is run again, and 'proyectoLinux' is now listed among the directories, underlined in red.

```
jos@jos:~$ ls
caso_2      Documentos  Música     proceso1   process_management  Público
caso_2.c    Escritorio  Plantillas proceso1.c  process_management.c snap
Descargas  Imágenes   proceso    proceso.c  proyecto           Videos
jos@jos:~$ mkdir proyectoLinux
jos@jos:~$ ls
caso_2      Escritorio  proceso    process_management  Público
caso_2.c    Imágenes   proceso1   process_management.c snap
Descargas  Música     proceso1.c proyecto           Videos
Documentos Plantillas proceso.c    proyectoLinux
```

Figura 2.1: Creación de Directorio Principal

2.1.2. Creación de subdirectorios para almacenar entradas, salidas y archivos de configuración

Se ubica dentro del directorio principal y se crea los subdirectorios: entrada, salida y config.

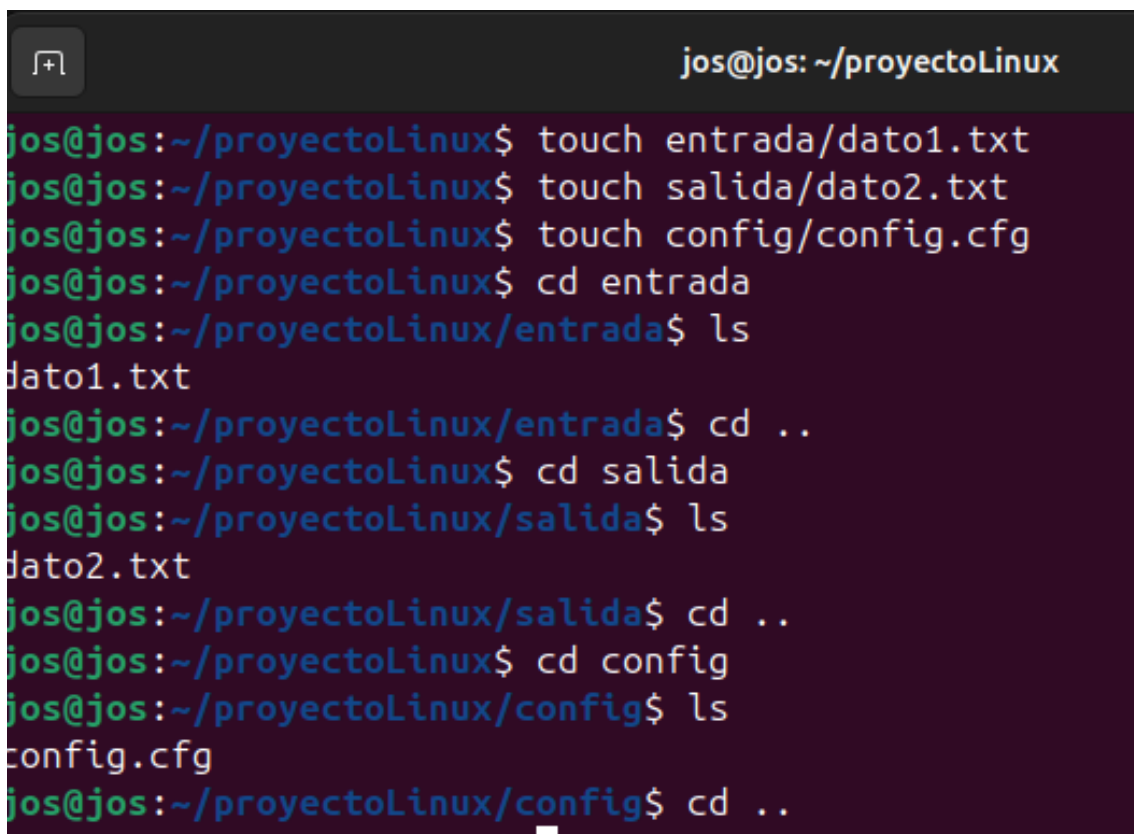
A terminal window with a dark background. The title bar shows a window icon and the text 'jos@jos: ~/proyectoLinux'. The terminal content shows a user named 'jos' at a machine named 'jos' in the directory '~'. They run 'cd proyectoLinux', then 'mkdir entrada salida config', and finally 'ls', which lists 'config', 'entrada', and 'salida'.

```
jos@jos:~$ cd proyectoLinux
jos@jos:~/proyectoLinux$ mkdir entrada salida config
jos@jos:~/proyectoLinux$ ls
config  entrada  salida
jos@jos:~/proyectoLinux$
```

Figura 2.2: Creación de Subdirectorios

2.1.3. Crear al menos un archivo vacío en cada subdirectorio

Para simular datos iniciales mediante el comando touch se generan archivos vacíos y se comprueban.

A terminal window with a dark background. The title bar shows a window icon and the text 'jos@jos: ~/proyectoLinux'. The terminal content shows the user creating files with 'touch entrada/dato1.txt', 'touch salida/dato2.txt', and 'touch config/config.cfg'. They then navigate to each directory ('cd entrada', 'cd salida', 'cd config') and run 'ls' to verify the files. Finally, they navigate back to the parent directory with 'cd ..' for each subdirectory.

```
jos@jos:~/proyectoLinux$ touch entrada/dato1.txt
jos@jos:~/proyectoLinux$ touch salida/dato2.txt
jos@jos:~/proyectoLinux$ touch config/config.cfg
jos@jos:~/proyectoLinux$ cd entrada
jos@jos:~/proyectoLinux/entrada$ ls
dato1.txt
jos@jos:~/proyectoLinux/entrada$ cd ..
jos@jos:~/proyectoLinux$ cd salida
jos@jos:~/proyectoLinux/salida$ ls
dato2.txt
jos@jos:~/proyectoLinux/salida$ cd ..
jos@jos:~/proyectoLinux$ cd config
jos@jos:~/proyectoLinux/config$ ls
config.cfg
jos@jos:~/proyectoLinux/config$ cd ..
```

Figura 2.3: Creación de archivos para simular datos iniciales

2.2. Gestión de usuarios y grupos

2.2.1. Crear un grupo de usuarios específico para el proyecto

Se crea el grupo:

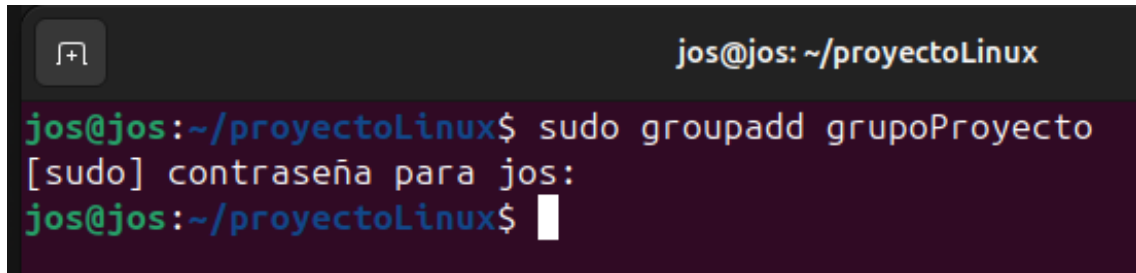
A terminal window with a dark background. The prompt is 'jos@jos: ~/proyectoLinux'. The command 'sudo groupadd grupoProyecto' is entered. The prompt changes to '[sudo] contraseña para jos:' and then back to 'jos@jos: ~/proyectoLinux\$' after a password is entered (indicated by a white rectangle).

Figura 2.4: Creación de un Grupo de usuarios

2.2.2. Agregar dos usuarios al sistema y asignarlos al grupo creado

Se agregan los usuarios:

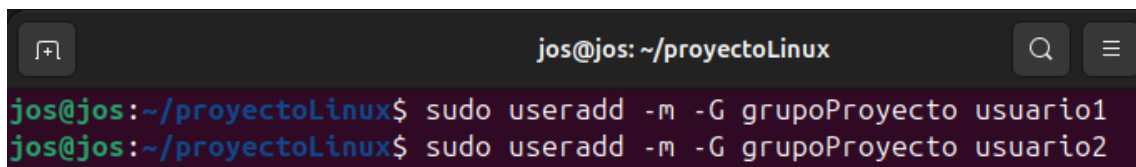
A terminal window with a dark background. The prompt is 'jos@jos: ~/proyectoLinux'. Two commands are entered: 'sudo useradd -m -G grupoProyecto usuario1' and 'sudo useradd -m -G grupoProyecto usuario2'. The prompt remains 'jos@jos: ~/proyectoLinux\$'.

Figura 2.5: Creación de usuarios

La explicación del comando es la siguiente:

- useradd → Crea un nuevo usuario en el sistema.
- -m → Crea automáticamente un directorio /home/usuario1.
- -G grupoProyecto → Agrega usuario1 al grupo grupoProyecto además del grupo predeterminado del usuario.

2.2.3. Configurar permisos de acceso a los directorios

Para que solo los miembros del grupo puedan modificarlos:

```
jos@jos:~$ sudo chown -R :grupoProyecto proyectoLinux
jos@jos:~$ sudo chmod -R 770 proyectoLinux
```

Figura 2.6: Permisos de acceso a los directorios

Explicación del comando `sudo chown -R :grupoProyecto proyectoLinux`:

- `chown` → Cambia el propietario de un archivo/directorio.
- `-R` → Aplica el cambio de forma recursiva a todos los archivos y subdirectorios.
- `:grupoProyecto` → Cambia solo el grupo propietario a `grupoProyecto` (el usuario dueño sigue siendo el mismo).

Explicación del comando `sudo chmod -R 770 proyectoLinux`

- `chmod` → Cambia los permisos de archivos y directorios.
- `-R` → Aplica los cambios recursivamente.
- `770` →
 - `7 (rwx)` → Dueño puede leer, escribir y ejecutar.
 - `7 (rwx)` → Grupo puede leer, escribir y ejecutar.
 - `0 (—)` → Otros no tienen acceso.

2.3. Creación del script

Crear un script en bash llamado **gestor.sh** que realice las siguientes funciones:

- Copiar archivos desde el subdirectorio de entrada al subdirectorio de salida.
- Registrar en un archivo de log la fecha, hora y los nombres de los archivos copiados.
- Mostrar un mensaje en la terminal indicando el éxito de la operación.

Mediante el comando `nano gestor.sh` se crea el script:



```
jos@jos: ~/proyectoLinux
GNU nano 7.2 gestor.sh *
#!/bin/bash

#Copiar archivo de entrada y salida
cp entrada/* salida/

#Registrar en log
echo "$(date) - Archivos copiados: $(ls entrada/)" >> log.txt

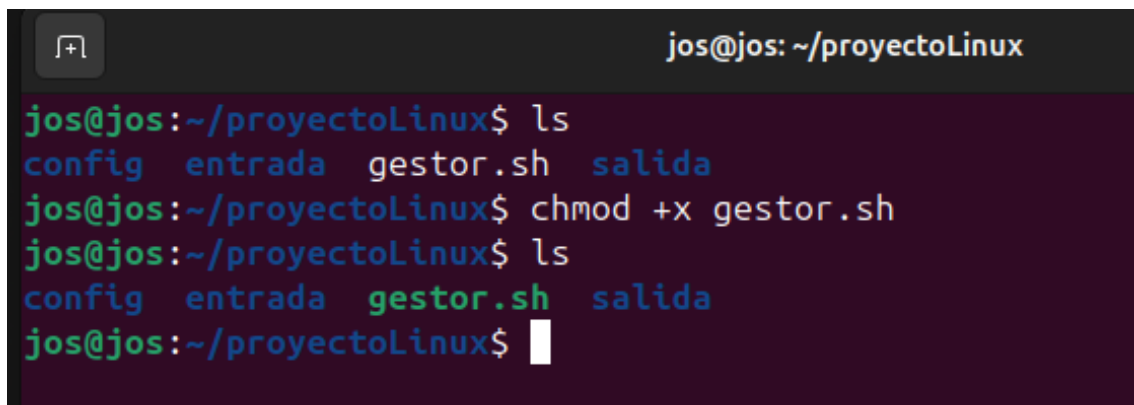
#Mensaje de exito
echo "Operacion completada con exito."
```

Figura 2.7: Script gestor.sh

2.4. Configuración y permisos

2.4.1. Asegurarse de que el script tenga permisos de ejecución.

Se le da permisos al archivo:

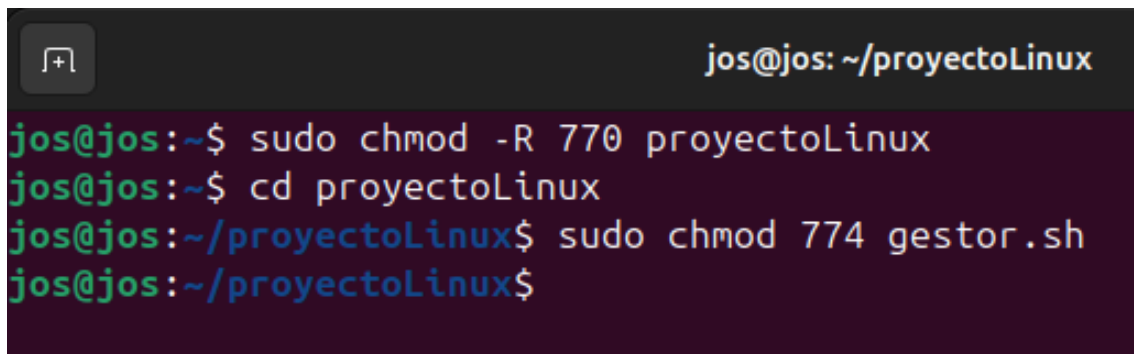


```
jos@jos: ~/proyectoLinux
jos@jos:~/proyectoLinux$ ls
config entrada gestor.sh salida
jos@jos:~/proyectoLinux$ chmod +x gestor.sh
jos@jos:~/proyectoLinux$ ls
config entrada gestor.sh salida
jos@jos:~/proyectoLinux$
```

Figura 2.8: Permisos de Ejecución

2.4.2. Configurar los permisos de los archivos y directorios

Para protegerlos de accesos no autorizados:



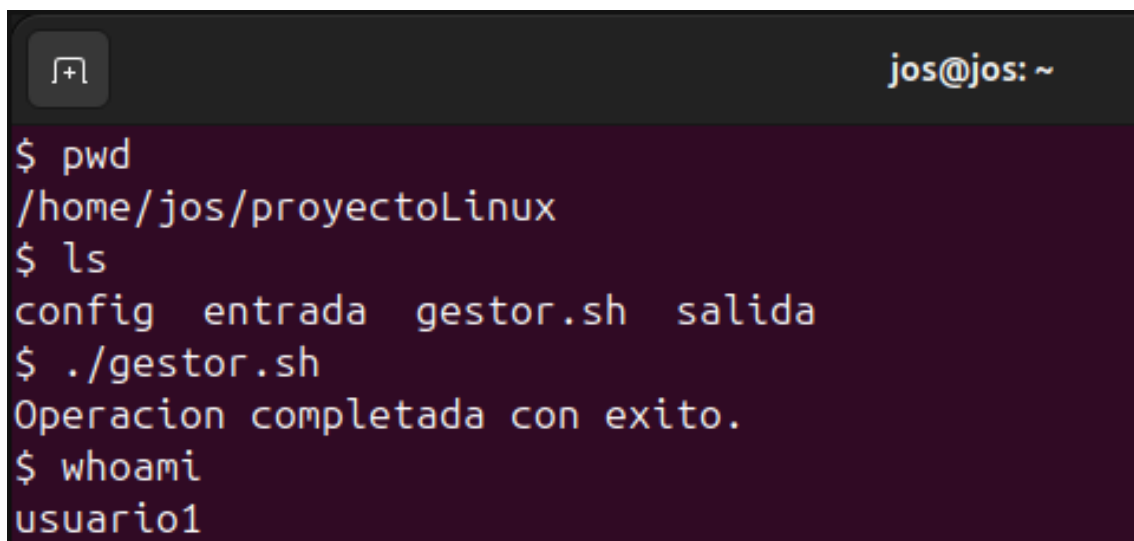
```
jos@jos: ~/proyectoLinux
jos@jos:~$ sudo chmod -R 770 proyectoLinux
jos@jos:~$ cd proyectoLinux
jos@jos:~/proyectoLinux$ sudo chmod 774 gestor.sh
jos@jos:~/proyectoLinux$
```

Figura 2.9: Permisos de Archivos y Directorios

2.5. Validación de acceso y pruebas

2.5.1. Cambiar al usuario creado y verificar que pueda ejecutar el script y acceder a los directorios del proyecto

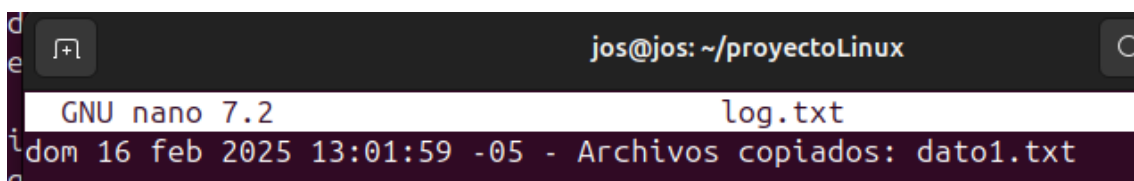
Se ejecuta el script con el usuario1:



```
jos@jos: ~
$ pwd
/home/jos/proyectoLinux
$ ls
config entrada gestor.sh salida
$ ./gestor.sh
Operacion completada con exito.
$ whoami
usuario1
```

Figura 2.10: Ejecución de Script con usuario con permisos

Se genera el archivo de los logs al ejecutar el script

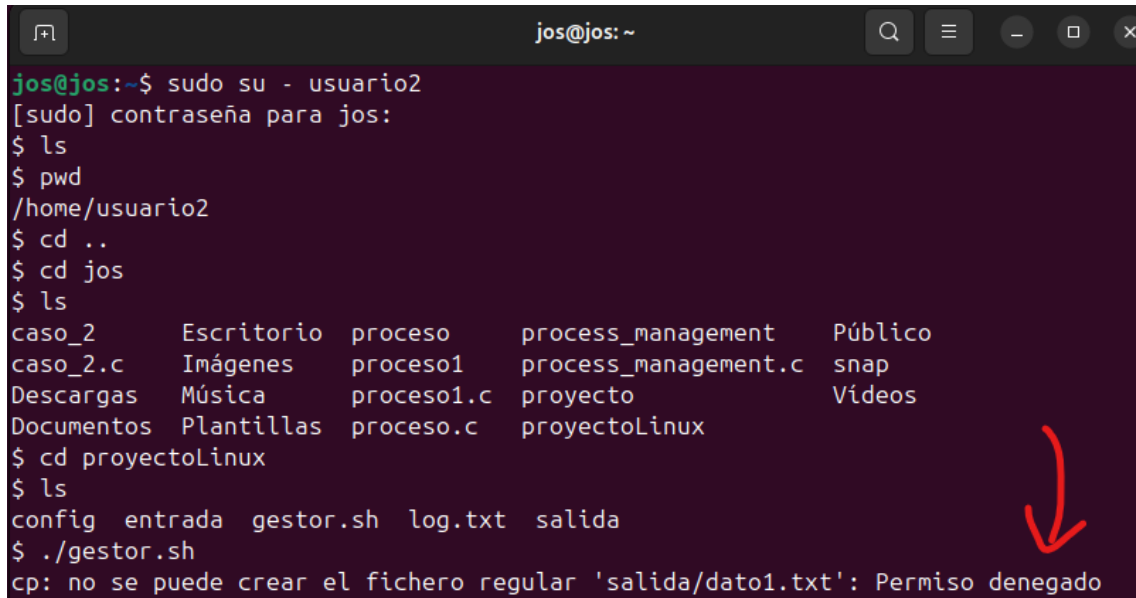


```
GNU nano 7.2 log.txt
dom 16 feb 2025 13:01:59 -05 - Archivos copiados: dato1.txt
```

Figura 2.11: Script Ejecutado

2.5.2. Intentar acceder a los directorios como un usuario que no pertenezca al grupo y comprobar que el acceso está restringido

Se intenta acceder y se comprueba que el acceso está restringido:



```
jos@jos: ~  
jos@jos:~$ sudo su - usuario2  
[sudo] contraseña para jos:  
$ ls  
$ pwd  
/home/usuario2  
$ cd ..  
$ cd jos  
$ ls  
caso_2      Escritorio  proceso    process_management  Público  
caso_2.c    Imágenes   proceso1    process_management.c  snap  
Descargas  Música     proceso1.c  proyecto             Vídeos  
Documentos Plantillas proceso.c    proyectoLinux  
$ cd proyectoLinux  
$ ls  
config entrada gestor.sh log.txt salida  
$ ./gestor.sh  
cp: no se puede crear el fichero regular 'salida/dato1.txt': Permiso denegado
```

A red arrow points to the error message at the bottom of the terminal output.

Figura 2.12: Permiso denegado

3. Conclusiones

- Se logró la creación y gestión de directorios, archivos y permisos en un entorno Linux, garantizando el acceso seguro solo a los usuarios autorizados.
- Se implementó un script Bash funcional que automatiza la copia de archivos y registra logs, demostrando la utilidad de los scripts en la administración del sistema.
- Se configuraron correctamente los permisos y la gestión de usuarios, validando restricciones de acceso y asegurando que solo el grupo autorizado pueda modificar los archivos del proyecto.
- La práctica permitió reforzar conocimientos sobre comandos de Linux y administración del sistema, esenciales para la gestión eficiente de entornos multiusuario.

Bibliografía

- [1] Enterprise Open Source and Linux — Ubuntu. (s. f.). Ubuntu.
<https://ubuntu.com/>