

LABORATORIO LINUX

By: Miguel Angel Acuña Reyes

Este laboratorio mostraremos como usar una máquina virtual de Linux en este caso usamos WSL (Windows Subsystem for Linux), el cual ya hemos instalado anteriormente.

El cual es muy sencillo.

- Abre PowerShell como administrador y ejecuta el siguiente comando para habilitar WSL:

```
wsl --install
```

- Reinicia tu computadora si se te solicita.

Instalar una Distribución de Linux en este caso Ubuntu:

```
wsl --install -d Ubuntu
```

Configurar WSL:

- Después de la instalación, abre tu distribución de Linux desde el menú de inicio y sigue las instrucciones para configurar tu usuario.

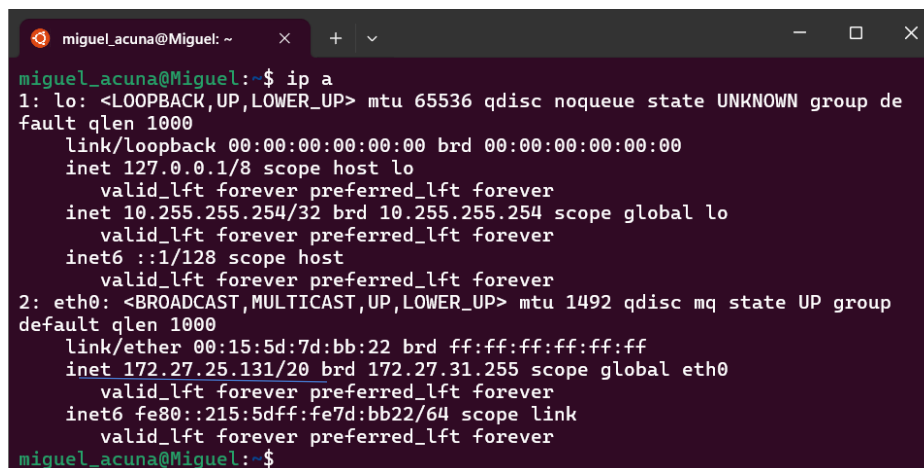
Usar SSH en WSL

- **Instalar el Cliente SSH en WSL:**
 - Abre tu terminal de WSL y asegúrate de que el cliente SSH esté instalado ejecutando:

```
sudo apt update  
sudo apt install openssh-client
```

Encontrar la IP en Linux.

Ejecutamos el comando ip a. Y encontramos nuestra ip la cual esta subrayada.



```
miguel_acuna@Miguel: ~  
miguel_acuna@Miguel:~$ ip a  
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group de  
fault qlen 1000  
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  
    inet 127.0.0.1/8 scope host lo  
        valid_lft forever preferred_lft forever  
    inet 10.255.255.254/32 brd 10.255.255.254 scope global lo  
        valid_lft forever preferred_lft forever  
    inet6 ::1/128 scope host  
        valid_lft forever preferred_lft forever  
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1492 qdisc mq state UP group  
default qlen 1000  
    link/ether 00:15:5d:7d:bb:22 brd ff:ff:ff:ff:ff:ff  
    inet 172.27.25.131/20 brd 172.27.31.255 scope global eth0  
        valid_lft forever preferred_lft forever  
    inet6 fe80::215:5dff:fe7d:bb22/64 scope link  
        valid_lft forever preferred_lft forever  
miguel_acuna@Miguel:~$
```

Acceder a una Máquina Remota desde PowerShell

1. Abrir PowerShell:

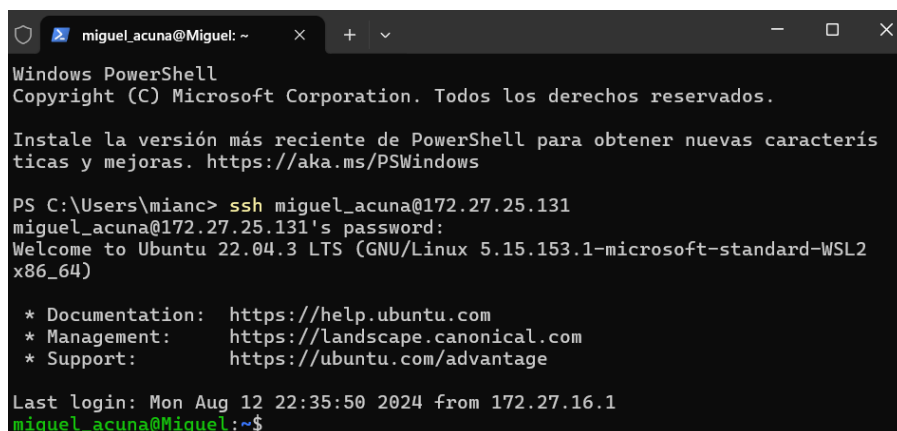
- Abre PowerShell en Windows.

2. Usar el Cliente SSH en PowerShell:

- Para conectarte a una máquina remota, usa el siguiente comando:

```
ssh usuario@direccion_ip
```

- Al igual que en WSL, reemplaza usuario con tu nombre de usuario y direccion_ip con la dirección IP de la máquina remota. Ingresa la contraseña y listo.



```
miguel_acuna@Miguel: ~
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows

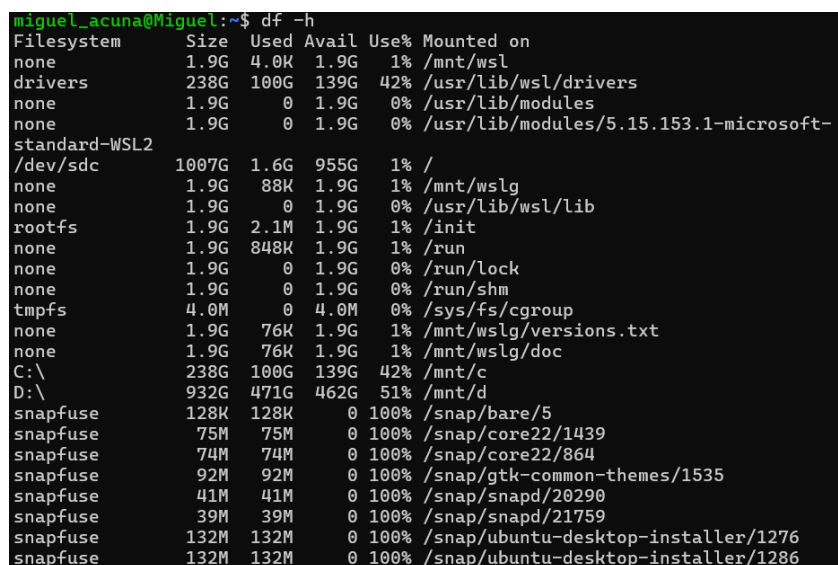
PS C:\Users\mianc> ssh miguel_acuna@172.27.25.131
miguel_acuna@172.27.25.131's password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.153.1-microsoft-standard-WSL2 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

Last login: Mon Aug 12 22:35:50 2024 from 172.27.16.1
miguel_acuna@Miguel:~$
```

1. COMANDOS BASICOS

- df -h :incluye las siguientes columnas:
 - **Filesystem**: El nombre del sistema de archivos o el dispositivo de almacenamiento.
 - **Size**: El tamaño total del sistema de archivos.
 - **Used**: La cantidad de espacio en disco que está siendo utilizado.
 - **Avail**: La cantidad de espacio en disco disponible para los usuarios.
 - **Use%**: El porcentaje de espacio en disco utilizado.
 - **Mounted on**: El punto de montaje en el sistema de archivos donde el sistema de archivos está montado.



```
miguel_acuna@Miguel:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
none            1.9G  4.0K  1.9G   1% /mnt/wsl
drivers         238G  100G  139G  42% /usr/lib/wsl/drivers
none            1.9G    0  1.9G   0% /usr/lib/modules
none            1.9G    0  1.9G   0% /usr/lib/modules/5.15.153.1-microsoft-
standard-WSL2
/dev/sdc        1007G  1.6G  955G   1% /
none            1.9G   88K  1.9G   1% /mnt/wslg
none            1.9G    0  1.9G   0% /usr/lib/wsl/lib
rootfs          1.9G  2.1M  1.9G   1% /init
none            1.9G  848K  1.9G   1% /run
none            1.9G    0  1.9G   0% /run/lock
none            1.9G    0  1.9G   0% /run/shm
tmpfs           4.0M    0  4.0M   0% /sys/fs/cgroup
none            1.9G   76K  1.9G   1% /mnt/wslg/versions.txt
none            1.9G   76K  1.9G   1% /mnt/wslg/doc
C:\             238G  100G  139G  42% /mnt/c
D:\             932G  471G  462G  51% /mnt/d
snapfuse        128K  128K    0 100% /snap/bare/5
snapfuse         75M   75M    0 100% /snap/core22/1439
snapfuse         74M   74M    0 100% /snap/core22/864
snapfuse         92M   92M    0 100% /snap/gtk-common-themes/1535
snapfuse         41M   41M    0 100% /snap/snapd/20290
snapfuse         39M   39M    0 100% /snap/snapd/21759
snapfuse        132M  132M    0 100% /snap/ubuntu-desktop-installer/1276
snapfuse        132M  132M    0 100% /snap/ubuntu-desktop-installer/1286
```

- `uname -a`: incluye las siguientes partes:

1. **Kernel Name**: El nombre del núcleo del sistema operativo (Linux).
2. **Node Name**: El nombre del nodo de red o el nombre del host.(Miguel)
3. **Kernel Release**: La versión del núcleo del sistema operativo (5.15.153.1-microsoft-standard-WSL2).
4. **Kernel Version**: La fecha y hora de la compilación del núcleo #1 SMP Fri Mar 29 23:14:13 UTC 2024).
5. **Machine**: El tipo de máquina o arquitectura del hardware (x86_64).
6. **Processor**: La arquitectura del procesador (normalmente igual al tipo de máquina, como x86_64).
7. **Hardware Platform**: La plataforma de hardware (x86_64).
8. **Operating System**: El nombre del sistema operativo (GNU/Linux).

```
miguel_acuna@Miguel:~$ uname -a
Linux Miguel 5.15.153.1-microsoft-standard-WSL2 #1 SMP Fri Mar 29 23:14:13 U
TC 2024 x86_64 x86_64 x86_64 GNU/Linux
```

- **Date**: muestra la fecha y la hora actual del sistema en un formato predeterminado. También puedes usarlo para configurar la fecha y la hora del sistema.

```
miguel_acuna@Miguel:~$ date
Mon Aug 12 23:16:27 -05 2024
```

- `mkdir` significa "make directory" (crear directorio).

```
miguel_acuna@Miguel:~$ mkdir newdirectory
miguel_acuna@Miguel:~$ ls -l
total 4
drwxrwxr-x 2 miguel_acuna miguel_acuna 4096 Aug 12 23:39 newdirectory
```

- `ls -l` en Linux se utiliza para listar el contenido de un directorio con detalles adicionales sobre cada archivo o directorio.
 - **Permisos**: Los permisos de archivo en un formato de 10 caracteres. El primer carácter indica el tipo de archivo (- para archivos regulares, d para directorios, l para enlaces simbólicos, etc.), y los siguientes nueve caracteres indican los permisos para el propietario, el grupo y otros usuarios.
 - **Número de Enlaces**: El número de enlaces duros al archivo o directorio.
 - **Propietario**: El nombre del usuario propietario del archivo.
 - **Grupo**: El nombre del grupo propietario del archivo.
 - **Tamaño**: El tamaño del archivo en bytes.
 - **Fecha y Hora de Modificación**: La fecha y hora de la última modificación del archivo.
 - **Nombre del Archivo o Directorio**: El nombre del archivo o directorio.
- `cd` significa "change directory" (cambiar de directorio).

```
miguel_acuna@Miguel:~$ cd newdirectory/
miguel_acuna@Miguel:~/newdirectory$ |
```

- `vi` : se utiliza para abrir el editor de texto Vi (o Vim) en sistemas Unix y Linux. Cuando usas `vi` seguido de un nombre de archivo, como en `vi newfile`, estás abriendo ese archivo en el editor. Si el archivo no existe, `vi` lo creará automáticamente.

```
miguel_acuna@Miguel:~/newdirectory$ vi newfile
```

Entra en modo insercion, escribes el texto que deseas, le das enter y posteriormente escribes “:wq” para guardar.

```
miguel_acuna@Miguel: ~/nev x + v
Hola, estamos editando archivos
~
~
~
:wq|
```

- chmod: se utiliza para cambiar los permisos de archivos y directorios.
- Permisos: r (lectura), w (escritura), x (ejecución).
- Modos:
- Numérico: chmod 755 archivo (propietario: lectura/escritura/ejecución; grupo/otros: lectura/ejecución).
- Simbólico: chmod u+x archivo (añade permiso de ejecución para el propietario).

```
miguel_acuna@Miguel:~/newdirectory$ ls -l
total 4
-rw-rw-r-- 1 miguel_acuna miguel_acuna 32 Aug 12 23:48 newfile
miguel_acuna@Miguel:~/newdirectory$ chmod g=rw newfile
miguel_acuna@Miguel:~/newdirectory$ ls -l
total 4
-rw-rw-r-- 1 miguel_acuna miguel_acuna 32 Aug 12 23:48 newfile
miguel_acuna@Miguel:~/newdirectory$ chmod g=r newfile
miguel_acuna@Miguel:~/newdirectory$ ls -l
total 4
-rw-r--r-- 1 miguel_acuna miguel_acuna 32 Aug 12 23:48 newfile
miguel_acuna@Miguel:~/newdirectory$ chmod 755 newfile
miguel_acuna@Miguel:~/newdirectory$ ls -l
total 4
-rwxr-xr-x 1 miguel_acuna miguel_acuna 32 Aug 12 23:48 newfile
miguel_acuna@Miguel:~/newdirectory$ |
```

- mv en Linux se usa para mover o renombrar archivos y directorios.

Uso Básico:

- o Renombrar: **mv archivo.txt nuevo_nombre.txt**

```
miguel_acuna@Miguel:~/newdirectory$ mv newfile archivonuevo
miguel_acuna@Miguel:~/newdirectory$ ls -l
total 4
-rwxr-xr-x 1 miguel_acuna miguel_acuna 32 Aug 12 23:48 archivonuevo
miguel_acuna@Miguel:~/newdirectory$ |
```

- o Mover: **mv archivo.txt /ruta/destino/**

```
miguel_acuna@Miguel:~/newdirectory$ mv archivonuevo ..
miguel_acuna@Miguel:~/newdirectory$ ls -l
total 0
miguel_acuna@Miguel:~/newdirectory$ cd ..
miguel_acuna@Miguel:~$ ls -l
total 8
-rwxr-xr-x 1 miguel_acuna miguel_acuna 32 Aug 12 23:48 archivonuevo
drwxrwxr-x 2 miguel_acuna miguel_acuna 4096 Aug 13 00:26 newdirectory
miguel_acuna@Miguel:~$ |
```

- rm -i : eliminar archivo y que pida confirmación, con rm -f no pide confirmación

```
miguel_acuna@Miguel:~$ rm -i archivonuevo
rm: remove regular file 'archivonuevo'? yes
miguel_acuna@Miguel:~$ ls -l
total 4
drwxrwxr-x 2 miguel_acuna miguel_acuna 4096 Aug 13 00:26 newdirectory
miguel_acuna@Miguel:~$ |
```

2. CREACION DE BASH

2.1.Days

2.1.1. Crear el archivo:

- Utiliza vi o nano para crear el archivo, por ejemplo:

```
vi days.sh
```

2.1.2. Escribir el script:

Dentro del editor, escribe el siguiente contenido:

```
#!/bin/bash
for days in Monday Tuesday Wednesday Thursday Friday Saturday Sunday
do
    echo "Day:$days"
done
```

```
#!/bin/bash
for days in Monday Tuesday Wednesday Thursday Friday Saturday Sunday
do
    echo "Day: $days"
done
```

2.1.3. Guardar y salir:

En vi, presiona Esc, escribe :wq, y presiona Enter para guardar y salir.

2.1.4. Ejecutar el script:

Ejecuta el script con:

```
bash days.sh
```

2.1.5. Resultado:

Cada día de la semana se imprimirá en la terminal precedido por "Day:".

```
miguel_acuna@Miguel:~/learn_bash$ bash days.sh
Day:Monday
Day:Tuesday
Day:Wednesday
Day:Thursday
Day:Friday
Day:Saturday
Day:Sunday
```

2.2. Increase

2.2.1. Escribir el script:

Dentro del editor, escribe el siguiente contenido:

```
#!/bin/bash
i=0
while [ $i -le 5 ]
do
    echo $i
    ((i++))
done
```

```
#!/bin/bash
i=0
while [ $i -le 5 ]
do
    echo $i
    ((i++))
done
```

2.2.2. Explicación:

- **i=0**: Inicia la variable i en 0.
- **while [\$i -le 5]**: Mientras i sea menor o igual a 5, ejecuta el bloque.
- **echo \$i**: Imprime el valor de i.
- **((i++))**: Incrementa i en 1 en cada iteración.

2.2.3. Resultado:

El script imprimirá los números del 0 al 5.

```
miguel_acuna@Miguel:~/learn_bash$ bash increase.sh
0
1
2
3
4
5
miguel_acuna@Miguel:~/learn_bash$ |
```

2.3.Conditional

2.3.1. Escribir el script:

Dentro del editor, escribe el siguiente contenido:

```
#!/bin/bash
invalid=true
count=1
while [ $invalid ]
do
    echo $count
    if [ $count -eq 5 ]
    then
        break
    fi
    ((count++))
done
```

```
#!/bin/bash
invalid=true
count=1
while [ $invalid ]
do
    echo $count
    if [ $count -eq 5 ]
    then
        break
    fi
    ((count++))
```

done

2.3.2. Explicación:

- ☐ **invalid=true:** Inicializa la variable invalid como true.
- ☐ **while [\$invalid]:** Mientras invalid sea true, ejecuta el bucle.
- ☐ **echo \$count:** Imprime el valor de count.
- ☐ **if [\$count -eq 5]:** Si count es igual a 5, se ejecuta el bloque then.
- ☐ **break:** Sale del bucle while cuando count es 5.
- ☐ **((count++)):** Incrementa count en 1 en cada iteración.

2.3.3. Resultado:

El script imprimirá los números del 1 al 5 y luego se detendrá.

```
miguel_acuna@Miguel:~/learn_bash$ bash conditional.sh
1
2
3
4
5
miguel_acuna@Miguel:~/learn_bash$ |
```

3. CRON & CRONTAB

Cron es una herramienta en Linux que permite programar tareas para que se ejecuten automáticamente en momentos específicos, como un temporizador. Las tareas que se programan en **cron** se llaman "jobs".

Crontab (abreviatura de "cron table") es un archivo de texto que contiene una lista de comandos y scripts que quieres que se ejecuten en horarios específicos. Puedes editar este archivo para añadir o modificar tareas programadas.

3.1. Ejemplo de uso de crontab:

- **Editar crontab:** `crontab -e`
- **Formato básico:** * * * * * comando

Cada asterisco representa una unidad de tiempo:

- Minuto (0-59)
 - Hora (0-23)
 - Día del mes (1-31)
 - Mes (1-12)
 - Día de la semana (0-7, donde 0 y 7 son domingo)
- `hwclock -w` en Linux que se utiliza para sincronizar la hora del sistema operativo con la hora del reloj de hardware (RTC, Real-Time Clock).

Explicación:

- **hwclock:** Interactúa con el reloj de hardware de la computadora.
- **-w:** Especifica que deseas escribir la hora actual del sistema en el reloj de hardware.

```
miguel_acuna@Miguel:~$ sudo hwclock -w
[sudo] password for miguel_acuna:
miguel_acuna@Miguel:~$
```

Esto asegura que la hora almacenada en el hardware sea la misma que la hora del sistema, lo cual es útil para mantener la hora correcta después de reiniciar o apagar la máquina.

- Script Bash se utiliza para liberar memoria caché en el sistema. Es útil si deseas liberar recursos del sistema, aunque en la mayoría de los casos, Linux maneja la memoria de manera eficiente por sí mismo.

Explicación:

- **sync:** Sincroniza los datos en la memoria con el disco, asegurándose de que todo lo que está pendiente de escritura se escriba en el disco.

- **echo 3 > /proc/sys/vm/drop_caches:** Esta línea le dice al sistema que libere la caché de páginas, inodos y dentries (estructura del sistema de archivos).

```
miguel_acuna@Miguel: ~
x + v
#!/bin/bash
sync; echo 3 > /proc/sys/vm/drop_caches
```

- ls -l ver los archivos y sus permisos

```
miguel_acuna@Miguel:~/learn_bash$ ls -l
total 16
-rw-rw-r-- 1 miguel_acuna miguel_acuna 52 Aug 13 10:04 cleancache.sh
-rw-rw-r-- 1 miguel_acuna miguel_acuna 147 Aug 13 01:02 conditional.sh
-rw-rw-r-- 1 miguel_acuna miguel_acuna 107 Aug 13 00:47 days.sh
-rw-rw-r-- 1 miguel_acuna miguel_acuna 61 Aug 13 00:55 increase.sh
miguel_acuna@Miguel:~/learn_bash$
```

- chmod +x cleancache.sh se utiliza para hacer que el archivo cleancache.sh sea ejecutable. Esto es necesario para que puedas ejecutar el script directamente desde la línea de comandos.

Explicación:

- **chmod +x:** Añade permisos de ejecución al archivo.
- **cleancache.sh:** El nombre del archivo al que quieres aplicar los permisos.

Después de Ejecutar el Comando:

- Ahora podrás ejecutar el script con ./cleancache.sh.

```
miguel_acuna@Miguel:~/learn_bash$ chmod +x cleancache.sh
miguel_acuna@Miguel:~/learn_bash$ ls -l
total 16
-rwxrwxr-x 1 miguel_acuna miguel_acuna 52 Aug 13 10:04 cleancache.sh
-rw-rw-r-- 1 miguel_acuna miguel_acuna 147 Aug 13 01:02 conditional.sh
-rw-rw-r-- 1 miguel_acuna miguel_acuna 107 Aug 13 00:47 days.sh
-rw-rw-r-- 1 miguel_acuna miguel_acuna 61 Aug 13 00:55 increase.sh
miguel_acuna@Miguel:~/learn_bash$ |
```

3.2. Programar la ejecución del archivo cleancache.sh usando crontab

3.2.1. Edita tu crontab:

Abre el archivo crontab para tu usuario con el siguiente comando:

```
crontab -e
```

3.2.2. Añade una línea para ejecutar el script:

En el editor de crontab, añade una línea con el formato adecuado para programar la ejecución de tu script.

Ejemplo:

Para ejecutar el script cleancache.sh todos los días a las 2 AM, añade lo siguiente:

```
0 2 * * * /home/miguel_acuna/learn_bash/cleancache.sh
```

- ****0 2 * * **:** Ejecuta el script a las 2:00 AM todos los días.

- `/home/miguel_acuna/learn_bash/cleancache.sh`: Ruta completa del script que quieres ejecutar.

3.2.3. Guarda y cierra el editor:

- En vi, presiona Esc, luego escribe `:wq` y presiona Enter.

3.2.4. Verifica la tarea programada:

Para asegurarte de que la tarea se ha programado correctamente, usa:

`crontab -l`

Esto listará todas las tareas programadas para tu usuario.

```
miguel_acuna@Miguel:~/learn_bash$ crontab -e
crontab: installing new crontab
miguel_acuna@Miguel:~/learn_bash$ crontab -l
0 2 * * * /home/miguel_acuna/learn_bash/cleancache.sh
miguel_acuna@Miguel:~/learn_bash$ |
```

4. CONCLUSIONES

4.1. Manejo de Archivos y Directorios:

- Aprendimos a usar comandos básicos para gestionar archivos y directorios en Linux, como `ls`, `cd`, `mkdir`, `mv`, y `chmod`. Estos comandos son esenciales para organizar y manejar el sistema de archivos de manera efectiva.

4.2. Uso de crontab y cron:

- **crontab** permite programar tareas para que se ejecuten automáticamente en momentos específicos. Creamos y editamos el archivo `crontab` para programar la ejecución de scripts, aprendiendo a usar el formato de tiempo para definir la frecuencia de ejecución.

4.3. Creación y Ejecución de Scripts:

- Desarrollamos scripts simples en Bash para tareas automatizadas, como liberar la memoria caché del sistema. Estos scripts se configuran para su ejecución automática usando `cron`, facilitando la gestión de tareas repetitivas y el mantenimiento del sistema.

4.4. Configuración de Permisos y Ejecución de Scripts:

- Usamos `chmod` para ajustar los permisos de ejecución en archivos de script, asegurando que puedan ser ejecutados como comandos.

Estas actividades proporcionan una comprensión práctica de la administración de sistemas en Linux, desde la gestión de archivos y directorios hasta la programación de tareas automáticas y la configuración de permisos.