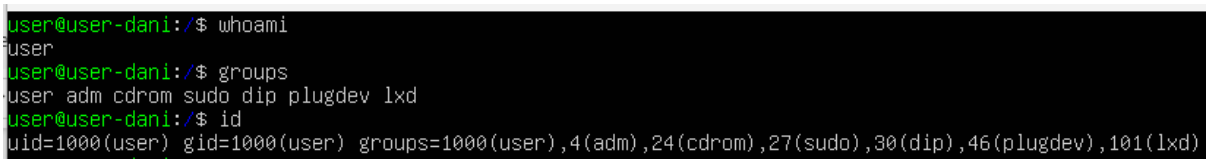
PR_01.3 Dani Gayol Rodríguez

Bloque 1: Comandos de Información y Preparación

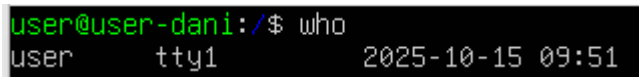
1. Identidad del Usuario: Abre una terminal y ejecuta un comando para saber qué usuario eres y a qué grupos perteneces.

Para saber que usuario soy se ejecuta el comando “whoami”, para los grupos, el comando “groups” y si quieres información más detallada, el comando “id”.



2. Usuarios Conectados: Muestra quién está conectado actualmente al sistema. Luego, ejecuta otro comando que te dé información más detallada, como el tiempo que llevan conectados y qué están ejecutando.

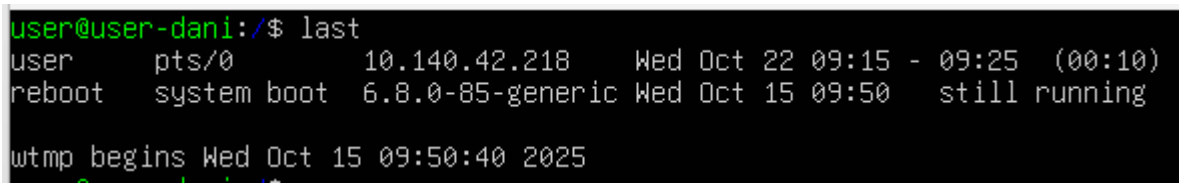
Para saber quién está conectado, se utiliza el comando “who”



Para obtener información más detallada se usa el comando “w”

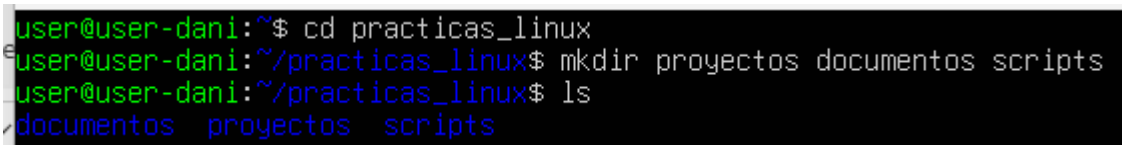
3. Historial de Conexiones: Lista los últimos inicios de sesión en el sistema.

Para listar los últimos inicios de sesión en el sistema se utiliza el comando “last”, si además quieres ver la última conexión de cada usuario en el sistema puedes usar el comando “lastlog”



4. Crear Entorno de Trabajo: En tu directorio personal (/home/tu_usuario), crea una carpeta principal para todos los ejercicios llamada practicas_linux .

5. Estructura de Directorios: Dentro de practicas_linux , crea la siguiente estructura de directorios: proyectos, documentos y scripts.



Bloque 2: Gestión de Usuarios y Grupos

1. Crear Grupos: Crea tres nuevos grupos en el sistema: desarrolladores, analistas y becarios.

Para crear grupos en el sistema, utilizamos el comando “groupadd”

```
user@user-dani:~/practicas_linux$ sudo groupadd desarrolladores
[sudo] password for user:
user@user-dani:~/practicas_linux$ sudo groupadd analistas
user@user-dani:~/practicas_linux$ sudo groupadd becarios
```

2. Verificar Grupos: Confirma que los grupos se han creado correctamente buscando sus nombres en el archivo /etc/group .

Comprobación de que los grupos fueron creados correctamente

```
user@user-dani:~/practicas_linux$ grep -E 'desarrolladores|analistas|becarios' /etc/group
desarrolladores:x:1001:
analistas:x:1002:
becarios:x:1003:
```

3. Crear un Usuario Básico: Crea un nuevo usuario llamado juan.

Para crear un usuario se utiliza el comando “useradd” y si queremos que se cree el directorio automáticamente se utiliza “-m”

```
user@user-dani:~/practicas_linux$ sudo useradd -m juan
user@user-dani:~/practicas_linux$ ls /home
juan  user
```

4. Crear Usuario con Grupo Primario: Crea una usuaria llamada ana y asígnala directamente al grupo primario desarrolladores.

Para crear un usuario y asignarlo a un grupo primario, se añade “-g” y el nombre del grupo

```
user@user-dani:~/practicas_linux$ sudo useradd -m -g desarrolladores ana
user@user-dani:~/practicas_linux$ ls /home
ana  juan  user
```

5. Crear Usuario Completo: Crea un usuario david asignándolo al grupo primario analistas y, a la vez, como miembro de los grupos secundarios desarrolladores y becarios.

Para crear un usuario y asignarlo a un grupo secundario, se añade “-G” en mayúscula

```
user@user-dani:~/practicas_linux$ sudo useradd -m -g analistas -G desarrolladores,becarios david
user@user-dani:~/practicas_linux$ ls /home
ana  david  juan  user
```

6. Establecer Contraseñas: Asigna una contraseña a los usuarios juan, ana y david.

Para asignar contraseñas a los usuarios, utilizamos el comando “passwd” y nos pedirá que introduzcamos la contraseña manualmente

```
user@user-dani:~/practicas_linux$ sudo passwd juan
New password:
Retype new password:
passwd: password updated successfully
user@user-dani:~/practicas_linux$ sudo passwd ana
New password:
Retype new password:
passwd: password updated successfully
user@user-dani:~/practicas_linux$ sudo passwd david
New password:
Retype new password:
passwd: password updated successfully
```

7. Verificar Usuarios: Comprueba que los tres nuevos usuarios existen en el sistema, inspeccionando el final del archivo /etc/passwd .

Para comprobar que los usuarios existen en el sistema lo miramos en el archivo “/etc/passwd” y como añadimos tres usuarios utilizaremos “-n 3” para que solo nos muestren estos tres usuarios

```
user@user-dani:~/practicas_linux$ tail -n 3 /etc/passwd
juan:x:1001:1004::/home/juan:/bin/sh
ana:x:1002:1001::/home/ana:/bin/sh
david:x:1003:1002::/home/david:/bin/sh
```

8. Cambiar de Usuario: Conviértete en el usuario juan usando el comando su . Una vez dentro de su sesión, comprueba quién eres y en qué directorio te encuentras. Vuelve a tu sesión de usuario original.

Para cambiar de usuario usamos el comando “su” y si queremos volver al usuario anterior ponemos el comando “exit” y nos devuelve a nuestro usuario anterior

```
user@user-dani:~/practicas_linux$ su juan
Password:
$ whoami
juan
$ pwd
/home/user/practicas_linux
$ exit
```

9. Modificar Grupos de un Usuario: Modifica al usuario juan para que su grupo primario sea becarios y añádelo también al grupo secundario analistas.

Para modificar el grupo primario y que se cambie a “becario” usamos el comando “usermod” y “-g” en minúsculas, en cambio para cambiar a un grupo secundario usamos el “-G” en mayúsculas y si en vez de cambiarlo queremos añadirlo y que los otros grupos se mantengan usamos el “-aG”, la “a” se utiliza para añadir

```
user@user-dani:~/practicas_linux$ sudo usermod -g becarios juan
[sudo] password for user:
user@user-dani:~/practicas_linux$ sudo usermod -aG analistas juan
```

10. Verificar Modificación: Comprueba que los cambios del usuario juan se han aplicado correctamente.

Para comprobar que los cambios se realizaron correctamente usamos el comando “id” y el nombre del usuario para mostrarnos la información de este usuario

```
user@user-dani:~/practicas_linux$ id juan
uid=1001(juan) gid=1003(becarios) groups=1003(becarios),1002(analistas)
```

11. Bloquear una Cuenta: Bloquea la cuenta del usuario juan para que no pueda iniciar sesión.

Este comando sirve para bloquear la cuenta modificando la contraseña en “/etc/shadow”

```
user@user-dani:~/practicas_linux$ sudo usermod -L juan
```

12. Intentar Cambiar a Usuario Bloqueado: Intenta convertirte en el usuario juan de nuevo. Debería fallar.

Nos va a mostrar un error y no nos va a dejar iniciar sesión mientras la cuenta esté bloqueada

```
user@user-dani:~/practicas_linux$ su juan
Password:
su: Authentication failure
```

13. Desbloquear una Cuenta: Desbloquea la cuenta del usuario juan.

Ahora para desbloquear una cuenta usamos el siguiente comando y si queremos verificar que se realizó correctamente, lo podemos comprobar con el siguiente comando “sudo passwd –S ……”

```
user@user-dani:~/practicas_linux$ sudo usermod -U juan
user@user-dani:~/practicas_linux$ sudo passwd -S juan
juan P 2025-10-23 0 99999 7 -1
user@user-dani:~/practicas_linux$ su juan
Password:
$ whoami
juan
$ exit
```

14. Eliminar un Grupo: Elimina el grupo becarios. ¿Qué ocurre? (Nota: Fallará si algún usuario lo tiene como grupo primario).

Si intentamos eliminar el grupo “becarios” nos va a saltar un error ya que este grupo está asignado como grupo primario para un usuario

```
user@user-dani:~/practicas_linux$ sudo groupdel becarios
groupdel: cannot remove the primary group of user 'juan'
```

15. Eliminar Usuario y su Directorio: Elimina al usuario juan y asegúrate de que su directorio personal (/home/juan) también se borre.

Si queremos eliminar a un usuario y a su directorio también, tenemos que utilizar “-r” para que se borre su directorio personal

```
user@user-dani:~/practicas_linux$ ls /home
ana  david  juan  user
user@user-dani:~/practicas_linux$ sudo userdel -r juan
userdel: group juan not removed because it is not the primary group of user juan.
userdel: juan mail spool (/var/mail/juan) not found
user@user-dani:~/practicas_linux$ ls /home
ana  david  user
```

Bloque 3: Permisos y Propiedad de Archivos

1. Crear Archivos de Prueba: Dentro de la carpeta proyectos, crea un archivo vacío llamado informe.txt. Dentro de scripts, crea otro archivo vacío llamado lanzar_app.sh.

Como ya tengo creadas las carpetas de proyectos y scripts, solo tengo que crear los archivos vacios

```
user@user-dani:~/practicas_linux$ ls
documentos  proyectos  scripts
user@user-dani:~/practicas_linux$ touch proyectos/informe.txt
user@user-dani:~/practicas_linux$ touch scripts/lanzar_app.sh
user@user-dani:~/practicas_linux$ ls proyectos
informe.txt
user@user-dani:~/practicas_linux$ ls scripts
lanzar_app.sh
```

2. Ver Permisos: Muestra los permisos por defecto de los archivos y directorios que has creado. Anota quién es el propietario y el grupo.

El propietario y el grupo pertenecen a “user”

```
user@user-dani:~/practicas_linux$ ls -l proyectos scripts
proyectos:
total 0
-rw-rw-r-- 1 user user 0 oct 27 08:09 informe.txt

scripts:
total 0
-rw-rw-r-- 1 user user 0 oct 27 08:09 lanzar_app.sh
```

3. Cambiar Propietario: Cambia el propietario del archivo informe.txt para que pertenezca a la usuaria ana .

Para cambiar el propietario del archivo usamos “chown”

```
user@user-dani:~/practicas_linux$ sudo chown ana proyectos/informe.txt
user@user-dani:~/practicas_linux$ ls -l proyectos
total 0
-rw-rw-r-- 1 ana user 0 oct 27 08:09 informe.txt
```

4. Cambiar Grupo: Cambia el grupo del directorio proyectos para que pertenezca al grupo desarrolladores.

Para cambiar el grupo de un directorio usamos “chgrp”

```
user@user-dani:~/practicas_linux$ sudo chgrp desarrolladores proyectos
```

5. Cambiar Propietario y Grupo: Cambia el propietario y el grupo del archivo lanzar_app.sh para que pertenezcan al usuario david y al grupo analistas, respectivamente, con un solo comando.

```
user@user-dani:~/practicas_linux$ sudo chown david:analistas scripts/lanzar_app.sh
user@user-dani:~/practicas_linux$ ls -l scripts
total 0
-rw-rw-r-- 1 david analistas 0 oct 27 08:09 lanzar_app.sh
```

6. Permisos con Notación Octal (Archivo): Usa la notación numérica (octal) para asignar los siguientes permisos a informe.txt: el propietario (ana) puede leer y escribir; el grupo (desarrolladores) solo puede leer; y los otros no tienen ningún permiso.

Para la lectura y escritura usamos “6”, para solo lectura usamos “4” y para no tener permisos usamos “0”

```
user@user-dani:~/practicas_linux$ sudo chmod 640 proyectos/informe.txt
user@user-dani:~/practicas_linux$ ls -l proyectos/informe.txt
-rw-r----- 1 ana user 0 oct 27 08:09 proyectos/informe.txt
```

7. Permisos con Notación Octal (Directorio): Asigna permisos de lectura, escritura y ejecución para el propietario y solo de lectura y ejecución para los miembros del grupo al directorio documentos.

Para la lectura, escritura y ejecución usamos “7”, para solo lectura y ejecución usamos “5”

```
user@user-dani:~/practicas_linux$ chmod 750 documentos
user@user-dani:~/practicas_linux$ ls -ld documentos
drwxr-x--- 2 user user 4096 oct 23 07:58 documentos
```

8. Verificar Permisos: Lista el contenido de practicas_linux para verificar que todos los cambios de propietario y permisos se han aplicado correctamente.

Para listar todo el contenido juntos a los cambios y permisos usamos “-lR”

```
user@user-dani:~/practicas_linux$ ls -lR ~/practicas_linux
/home/user/practicas_linux:
total 12
drwxr-x--- 2 user user            4096 oct 23 07:58 documentos
drwxrwxr-x 2 user desarrolladores 4096 oct 27 08:09 proyectos
drwxrwxr-x 2 user user            4096 oct 27 08:09 scripts

/home/user/practicas_linux/documentos:
total 0

/home/user/practicas_linux/proyectos:
total 0
-rw-r----- 1 ana user 0 oct 27 08:09 informe.txt

/home/user/practicas_linux/scripts:
total 0
-rw-rw-r-- 1 david analistas 0 oct 27 08:09 lanzar_app.sh
```

9. Permisos con Notación Simbólica (Añadir): Usa la notación simbólica para añadir el permiso de ejecución al propietario del script lanzar_app.sh.

Para darle permisos de ejecución solo al propietario usamos “u+x”

```
user@user-dani:~/practicas_linux$ sudo chmod u+x scripts/lanzar_app.sh
user@user-dani:~/practicas_linux$ ls -l scripts/lanzar_app.sh
-rwxrw-r-- 1 david analistas 0 oct 27 08:09 scripts/lanzar_app.sh
```

10. Permisos con Notación Simbólica (Quitar): Quita el permiso de lectura al “resto del mundo” (otros) en el directorio proyectos.

Para quitarle el permiso de lectura al resto del mundo usamos “o-r”

```
user@user-dani:~/practicas_linux$ sudo chmod o-r proyectos
user@user-dani:~/practicas_linux$ ls -ld proyectos
drwxrwx--x 2 user desarrolladores 4096 oct 27 08:09 proyectos
```

11. Permisos Recursivos: Dentro de proyectos, crea una nueva carpeta version2 con un archivo notas.txt dentro. Luego, cambia el propietario de la carpeta proyectos y todo su contenido para que pertenezca a david con un solo comando recursivo.

Primero creamos la carpeta junto al archivo y después usamos “-R” para que el propietario de la carpeta se cambie

```
user@user-dani:~/practicas_linux$ mkdir proyectos/version2
user@user-dani:~/practicas_linux$ touch proyectos/version2/notas.txt
user@user-dani:~/practicas_linux$ sudo chown -R david proyectos
user@user-dani:~/practicas_linux$ ls -lR proyectos
ls: cannot open directory 'proyectos': Permission denied
user@user-dani:~/practicas_linux$ sudo ls -lR proyectos
proyectos:
total 4
-rw-r----- 1 david user    0 oct 27 08:09 informe.txt
drwxrwxr-x 2 david user 4096 oct 27 08:37 version2

proyectos/version2:
total 0
-rw-rw-r-- 1 david user 0 oct 27 08:37 notas.txt
```

12. Permiso Especial SGID en Directorio: Establece el permiso especial SGID en el directorio documentos. Después, cambia a ser el usuario david (su david) y crea un nuevo archivo dentro de documentos. Verifica a qué grupo pertenece el nuevo archivo (debería heredar el del directorio documentos). Vuelve a tu usuario.

Para establecer el permiso especial SGID usamos “g+s”

```
user@user-dani:~/practicas_linux$ sudo chmod g+s documentos
user@user-dani:~/practicas_linux$ ls -ld documentos
drwxr-s--- 2 user user 4096 oct 23 07:58 documentos
```

El problema está cuando intento acceder a la carpeta de “documentos” ya que solo tienen permiso el usuario o el grupo “user”, por lo tanto no me va a dejar hacer nada. Una opción sería cambiar el grupo de la carpeta como por ejemplo a “desarrolladores”

```
user@user-dani:~/practicas_linux$ ls -ld documentos
drwxr-s--- 2 user user 4096 oct 23 07:58 documentos
user@user-dani:~/practicas_linux$ sudo chown david:desarrolladores documentos
user@user-dani:~/practicas_linux$ sudo chmod 2770 documentos
user@user-dani:~/practicas_linux$ ls -ld documentos
drwxrws--- 2 david desarrolladores 4096 oct 23 07:58 documentos
```

```
user@user-dani:~/practicas_linux$ sudo su david
$ pwd
/home/user/practicas_linux
$ cd documentos
$ touch nuevo_archivo.txt
$ ls -l
total 0
-rw-r--r-- 1 david desarrolladores 0 oct 27 09:00 nuevo_archivo.txt
$ exit
```

13. Permiso Especial SUID: Establece el permiso SUID en el script lanzar_app.sh. (Nota: Explica a tus alumnos qué implicaría esto si fuera un programa compilado).

Para darle el permiso especial SUID usamos “u+s”

```
user@user-dani:~/practicas_linux$ sudo chmod u+s scripts/lanzar_app.sh
user@user-dani:~/practicas_linux$ ls -l scripts/lanzar_app.sh
-rwsrw-r-- 1 david analistas 0 oct 27 08:09 scripts/lanzar_app.sh
```

En un programa compilado, el SUID permite que el programa se ejecute con los privilegios del propietario del archivo, esto no tiene efecto en scripts, pero sí en binarios como “/usr/bin/passwd”.

14. Comprobar umask : Muestra el valor umask actual de tu sesión.

La umask es una máscara de permisos por defecto que se aplica cuando se crean nuevos archivos o directorios

```
user@user-dani:~/practicas_linux$ umask
0002
```

En mi caso, el “0002” significa que sistema está configurado para que otros usuarios del mismo grupo puedan leer y escribir los archivos que crees

15. Efecto de umask : Cambia temporalmente tu umask a 077. Crea un nuevo archivo llamado privado.txt. Comprueba sus permisos por defecto. Luego, restaura el umask a su valor original.

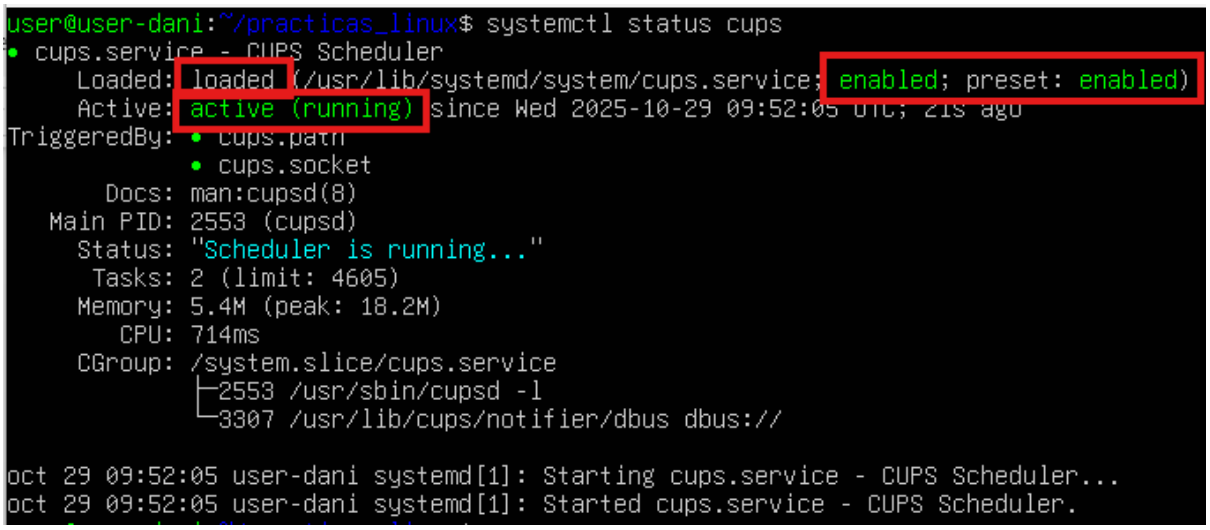
Cambiando la umask a “077” hace que solo el propietario pueda leer y escribir en el archivo

```
user@user-dani:~/practicas_linux$ umask 077
user@user-dani:~/practicas_linux$ touch privado.txt
user@user-dani:~/practicas_linux$ ls -l privado.txt
-rw------- 1 user user 0 oct 27 09:10 privado.txt
user@user-dani:~/practicas_linux$ umask 0002
```

Bloque 4: Gestión de Servicios con systemctl

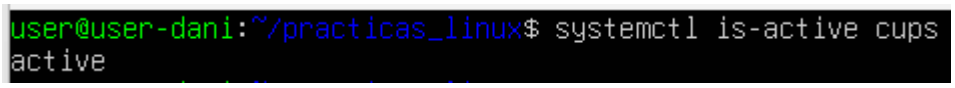
1. Estado Detallado de un Servicio: Comprueba el estado completo del servicio cups. Analiza la salida: ¿está activo (active), cargado (loaded) y habilitado (enabled)? Anota las últimas líneas de su registro (log) que aparecen.

En mi Ubuntu no venía instalado el servicio “cups”, por lo tanto, tuve que instalarlo. Una vez instalado muestra lo siguiente



Se puede ver como el servicio está habilitado (enable), también aparece como cargado (loaded) y finalmente está activo (active)

2. Comprobación Rápida: Utiliza un comando más directo para verificar si el servicio cups está actualmente en ejecución (activo). La salida de este comando debería ser simplemente active o inactive.



3. Ver Archivo de Unidad: Muestra el contenido del archivo de unidad del servicio cups (cups.service). Esto te permitirá ver cómo está definido el servicio.

Podemos usar un “cat” del archivo que muestra el contenido o simplemente utilizar el “cat” junto al nombre del servicio

```
user@user-dani:~/practicas_linux$ systemctl cat cups
# /usr/lib/systemd/system/cups.service
[Unit]
Description=CUPS Scheduler
Documentation=man:cupsd(8)
After=network.target nss-user-lookup.target nslcd.service
Requires=cups.socket

[Service]
ExecStart=/usr/sbin/cupsd -l
Type=notify
Restart=on-failure


[Install]
Also=cups.socket cups.path
WantedBy=printer.target multi-user.target
```

4. Detener un Servicio: Detén la ejecución del servicio cups. Comprueba su estado de nuevo para confirmar que está inactive (dead).

```
user@user-dani:~/practicas_linux$ sudo systemctl stop cups
user@user-dani:~/practicas_linux$ systemctl status cups
● cups.service - CUPS Scheduler
     Loaded: loaded (/usr/lib/systemd/system/cups.service; enabled; preset: enabled)
     Active: inactive (dead) since Wed 2025-10-29 09:58:17 UTC; 7s ago
   Duration: 6min 12.125s
TriggeredBy: ● cups.path
             ● cups.socket
       Docs: man:cupsd(8)
    Process: 2553 ExecStart=/usr/sbin/cupsd -l (code=exited, status=0/SUCCESS)
   Main PID: 2553 (code=exited, status=0/SUCCESS)
     Status: "Scheduler is running..."
        CPU: 860ms

oct 29 09:52:05 user-dani systemd[1]: Starting cups.service - CUPS Scheduler...
oct 29 09:52:05 user-dani systemd[1]: Started cups.service - CUPS Scheduler.
oct 29 09:58:17 user-dani systemd[1]: Stopping cups.service - CUPS Scheduler...
oct 29 09:58:17 user-dani systemd[1]: cups.service: Deactivated successfully.
oct 29 09:58:17 user-dani systemd[1]: Stopped cups.service - CUPS Scheduler.
```

5. Iniciar un Servicio: Vuelve a iniciar el servicio cups. Verifica una vez más que ha vuelto al estado active (running).

```
user@user-dani:~/practicas_linux$ sudo systemctl start cups
user@user-dani:~/practicas_linux$ systemctl status cups
● cups.service - CUPS Scheduler
     Loaded: loaded (/usr/lib/systemd/system/cups.service; enabled; preset: enabled)
     Active: active (running) since Wed 2025-10-29 10:01:22 UTC; 3s ago
TriggeredBy: ● cups.path
             ● cups.socket
       Docs: man:cupsd(8)
   Main PID: 3459 (cupsd)
     Status: "Scheduler is running..."
      Tasks: 1 (limit: 4605)
     Memory: 1.7M (peak: 1.9M)
        CPU: 12ms
     CGroup: /system.slice/cups.service
             └─3459 /usr/sbin/cupsd -l

oct 29 10:01:22 user-dani systemd[1]: Starting cups.service - CUPS Scheduler...
oct 29 10:01:22 user-dani systemd[1]: Started cups.service - CUPS Scheduler.
```

6. Reiniciar un Servicio: El comando restart es muy común tras un cambio de configuración. Ejecútalo para el servicio cups.

```
user@user-dani:~/practicas_linux$ sudo systemctl restart cups
user@user-dani:~/practicas_linux$ systemctl status cups
● cups.service - CUPS Scheduler
     Loaded: loaded (/usr/lib/systemd/system/cups.service; enabled; preset: enabled)
     Active: active (running) since Wed 2025-10-29 10:02:25 UTC; 2s ago
TriggeredBy: ● cups.path
             ● cups.socket
       Docs: man:cupsd(8)
   Main PID: 3473 (cupsd)
     Status: "Scheduler is running..."
      Tasks: 1 (limit: 4605)
     Memory: 1.7M (peak: 1.9M)
        CPU: 12ms
     CGroup: /system.slice/cups.service
             └─3473 /usr/sbin/cupsd -l

oct 29 10:02:25 user-dani systemd[1]: Starting cups.service - CUPS Scheduler...
oct 29 10:02:25 user-dani systemd[1]: Started cups.service - CUPS Scheduler.
```

7. Habilitar para el Arranque: Asegúrate de que el servicio cups esté configurado para iniciarse automáticamente cada vez que el sistema arranque.

```
user@user-dani:~/practicas_linux$ sudo systemctl enable cups
Synchronizing state of cups.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable cups
```

8. Verificar si está Habilitado: Usa un comando específico para preguntar si cups está habilitado. La salida debería ser enabled o disabled .

```
user@user-dani:~/practicas_linux$ systemctl is-enabled cups
enabled
```

9. Deshabilitar para el Arranque: Ahora, desactiva el servicio cups para que no se inicie automáticamente.

```
user@user-dani:~/practicas_linux$ sudo systemctl disable cups
Synchronizing state of cups.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install disable cups
Removed "/etc/systemd/system/printer.target.wants/cups.service".
Removed "/etc/systemd/system/multi-user.target.wants/cups.path".
Removed "/etc/systemd/system/multi-user.target.wants/cups.service".
Removed "/etc/systemd/system/sockets.target.wants/cups.socket".
Disabling 'cups.service', but its triggering units are still active:
cups.socket
user@user-dani:~/practicas_linux$ systemctl is-enabled cups
disabled
```

10. Enmascarar un Servicio: El enmascaramiento es una forma más contundente de deshabilitar, ya que impide cualquier tipo de inicio (manual o automático). Enmascara el servicio cups. Intenta iniciarlo después. Debería fallar. No olvides desenmascararlo (unmask) al terminar el ejercicio.

Primero vamos a enmascarar el servicio

```
user@user-dani:~/practicas_linux$ sudo systemctl mask cups
Created symlink /etc/systemd/system/cups.service → /dev/null.
Masking 'cups.service', but its triggering units are still active:
cups.socket
```

Ahora si intentamos iniciarlo nos va a salir un error

```
user@user-dani:~/practicas_linux$ sudo systemctl start cups
Failed to start cups.service: Unit cups.service is masked.
```

Y finalmente vamos a desenmascararlo

```
user@user-dani:~/practicas_linux$ sudo systemctl unmask cups
Removed "/etc/systemd/system/cups.service".
```

Bloque 5: Gestión de ufw

1. Comprobar Estado y Activar UFW: * Primero, ejecuta un comando para verificar el estado actual del firewall. Probablemente estará inactivo. * A continuación, activa UFW. Presta atención al mensaje de advertencia, especialmente si estás conectado por SSH.

Como nunca lo inicie pues me sale inactivo por defecto

```
user@user-dani:~/practicas_linux$ sudo ufw status
Status: inactive
```

Ahora vamos a iniciarlo

```
user@user-dani:~/practicas_linux$ sudo ufw enable
Firewall is active and enabled on system startup
```

2. Permitir un Servicio Web (HTTP): * Imagina que tu servidor necesita alojar una página web. Añade una regla para permitir todas las conexiones entrantes para el servicio http. * Verifica el estado del firewall de nuevo para confirmar que la regla (y el puerto 80) se ha añadido correctamente.

Primero, vamos a permitir todas las conexiones entrantes

```
user@user-dani:~/practicas_linux$ sudo ufw allow http
Rule added
Rule added (v6)
```

Y ahora vamos a verificar el estado del firewall

```
user@user-dani:~/practicas_linux$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
80/tcp                     ALLOW       Anywhere
80/tcp (v6)                ALLOW       Anywhere (v6)
```

3. Abrir un Puerto Específico: * Imagina que estás ejecutando un servidor de aplicaciones web en el puerto 8080. Añade una regla para permitir las conexiones entrantes TCP a ese puerto.

Para abrir un puerto en específico, usamos el siguiente comando

```
user@user-dani:~/practicas_linux$ sudo ufw allow 8080/tcp
Rule added
Rule added (v6)
user@user-dani:~/practicas_linux$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
80/tcp                     ALLOW       Anywhere
8080/tcp                   ALLOW       Anywhere
80/tcp (v6)                ALLOW       Anywhere (v6)
8080/tcp (v6)              ALLOW       Anywhere (v6)
```

4. Permitir un Rango de Puertos: * Supón que una aplicación FTP necesita un rango de puertos pasivos. Añade una regla para permitir las conexiones TCP en el rango de puertos desde el 3000 al 3100.

Para permitir un rango de puertos utilizamos este comando

```
user@user-dani:~/practicas_linux$ sudo ufw allow 3000:3100/tcp
Rule added
Rule added (v6)
user@user-dani:~/practicas_linux$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
80/tcp                     ALLOW       Anywhere
8080/tcp                   ALLOW       Anywhere
3000:3100/tcp              ALLOW       Anywhere
80/tcp (v6)                ALLOW       Anywhere (v6)
8080/tcp (v6)              ALLOW       Anywhere (v6)
3000:3100/tcp (v6)         ALLOW       Anywhere (v6)
```

5. Bloquear una Dirección IP: * Por seguridad, has detectado actividad sospechosa desde la IP 192.168.100.50. Añade una regla para denegar todas las conexiones provenientes de esa dirección IP.

Para bloquear una dirección IP especifica hacemos lo siguiente

```
user@user-dani:~/practicas_linux$ sudo ufw deny from 192.168.100.50
Rule added
user@user-dani:~/practicas_linux$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
80/tcp                     ALLOW       Anywhere
8080/tcp                   ALLOW       Anywhere
3000:3100/tcp              ALLOW       Anywhere
Anywhere                   DENY        192.168.100.50
80/tcp (v6)                ALLOW       Anywhere (v6)
8080/tcp (v6)              ALLOW       Anywhere (v6)
3000:3100/tcp (v6)         ALLOW       Anywhere (v6)
```

6. Listar Reglas para Borrar: * Muestra todas las reglas activas del firewall, pero esta vez de forma numerada, para prepararte para eliminar una de ellas.

Para listarlas usamos el comando “numbered”

```
user@user-dani:~/practicas_linux$ sudo ufw status numbered
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] 80/tcp                     ALLOW IN    Anywhere
[ 2] 8080/tcp                   ALLOW IN    Anywhere
[ 3] 3000:3100/tcp              ALLOW IN    Anywhere
[ 4] Anywhere                   DENY IN     192.168.100.50
[ 5] 80/tcp (v6)                ALLOW IN    Anywhere (v6)
[ 6] 8080/tcp (v6)              ALLOW IN    Anywhere (v6)
[ 7] 3000:3100/tcp (v6)         ALLOW IN    Anywhere (v6)
```

7. Eliminar una Regla: * Basándote en la lista del ejercicio anterior, elimina la regla que creaste para el puerto 8080. * Vuelve a listar las reglas (de forma normal o numerada) para confirmar que la regla ha sido eliminada correctamente.

Para eliminar una regla utilizamos el “delete” junto al número de la lista numerada

```
user@user-dani:~/practicas_linux$ sudo ufw delete 2
Deleting:
 allow 8080/tcp
Proceed with operation (y|n)? y
Rule deleted
user@user-dani:~/practicas_linux$ sudo ufw status numbered
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] 80/tcp                     ALLOW IN    Anywhere
[ 2] 3000:3100/tcp              ALLOW IN    Anywhere
[ 3] Anywhere                   DENY IN     192.168.100.50
[ 4] 80/tcp (v6)                ALLOW IN    Anywhere (v6)
[ 5] 8080/tcp (v6)              ALLOW IN    Anywhere (v6)
[ 6] 3000:3100/tcp (v6)         ALLOW IN    Anywhere (v6)
```