

# Modulo 1 –Tema 2. Terminal y línea de comandos de Linux.

---

Comandos para la navegación básica en la terminal  
de Linux

# Objetivos:

Proporcionar las habilidades y conocimientos necesarios para utilizar la terminal de Linux de forma eficiente en sus proyectos de análisis de datos.

# Contenido

## 1.- Introducción a la terminal de Linux:

- Conceptos básicos: shell, comandos, argumentos, opciones
- Navegación por el sistema de archivos: cd, ls, pwd, mkdir, rmdir
- Edición de archivos: nano, vim
- Permisos de archivos y directorios: chmod, chown

## 2.- Comandos básicos de la terminal:

- grep: búsqueda de texto en archivos
- sort: ordenar líneas de un archivo
- uniq: eliminar líneas duplicadas
- wc: contar líneas, palabras y caracteres
- head/tail: mostrar las primeras/últimas líneas de un archivo
- tee: enviar la salida de un comando a un archivo y a la pantalla
- pipe: combinar la salida de dos comandos

# Conceptos básicos. Shell de comandos

Linux dispone de una aplicación denominada intérprete de comandos o shell de comandos que permite al usuario interactuar con el sistema operativo mediante la ejecución de comandos o sentencias de texto. Hay diferentes tipos de intérpretes que se diferencian en la sintaxis de los comandos y en la forma de interaccionar con el usuario.

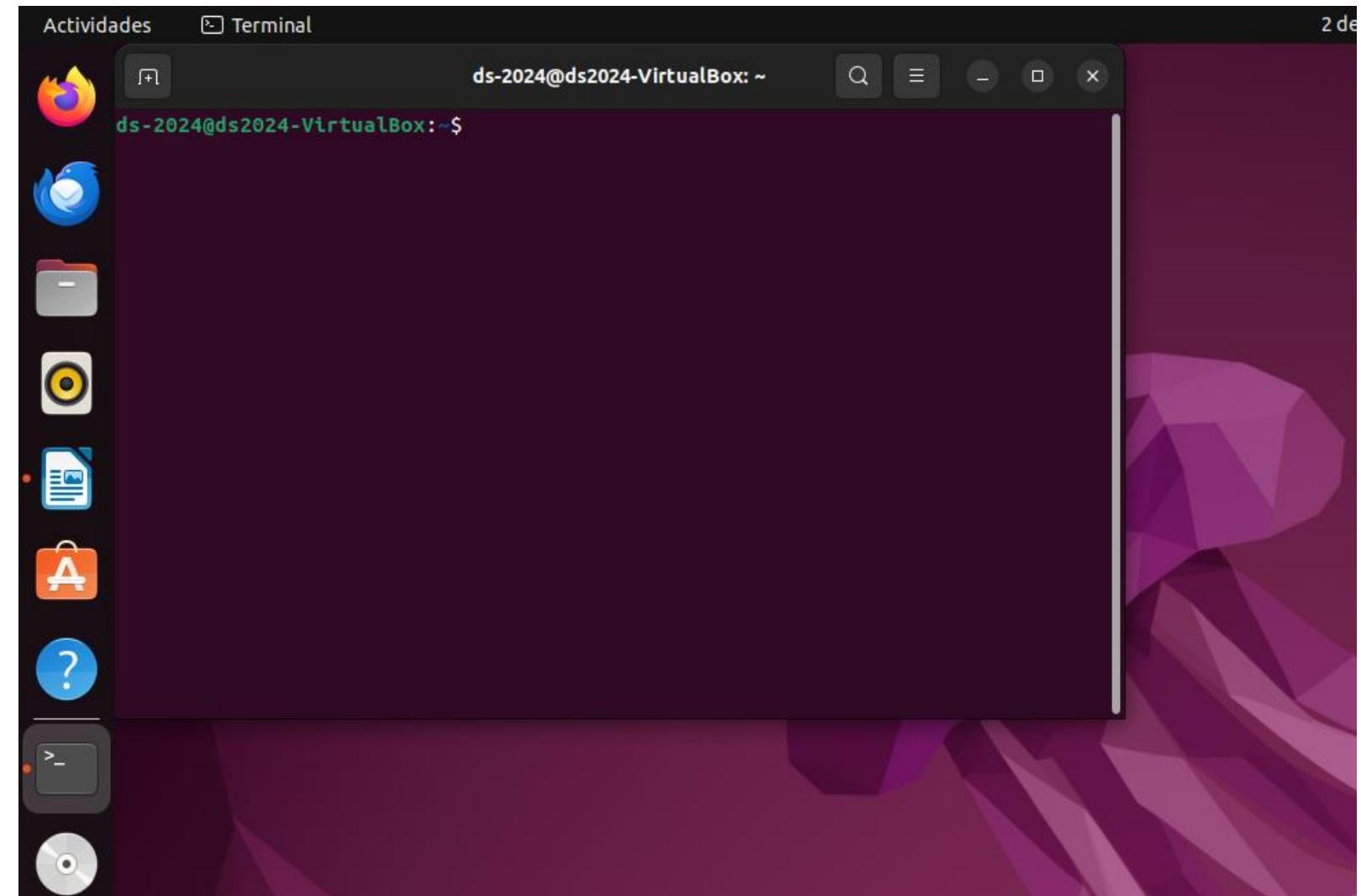
# Conceptos básicos. Shell de comandos

Según el sistema Linux utilizado, habrá diferentes tipos de shell. La más utilizada es Bash (Bourne Again Shell), que no solo permite ejecutar comandos puntuales, sino que dispone de un sistema de sentencias donde se pueden componer scripts o automatismos de cierto grado de complejidad.

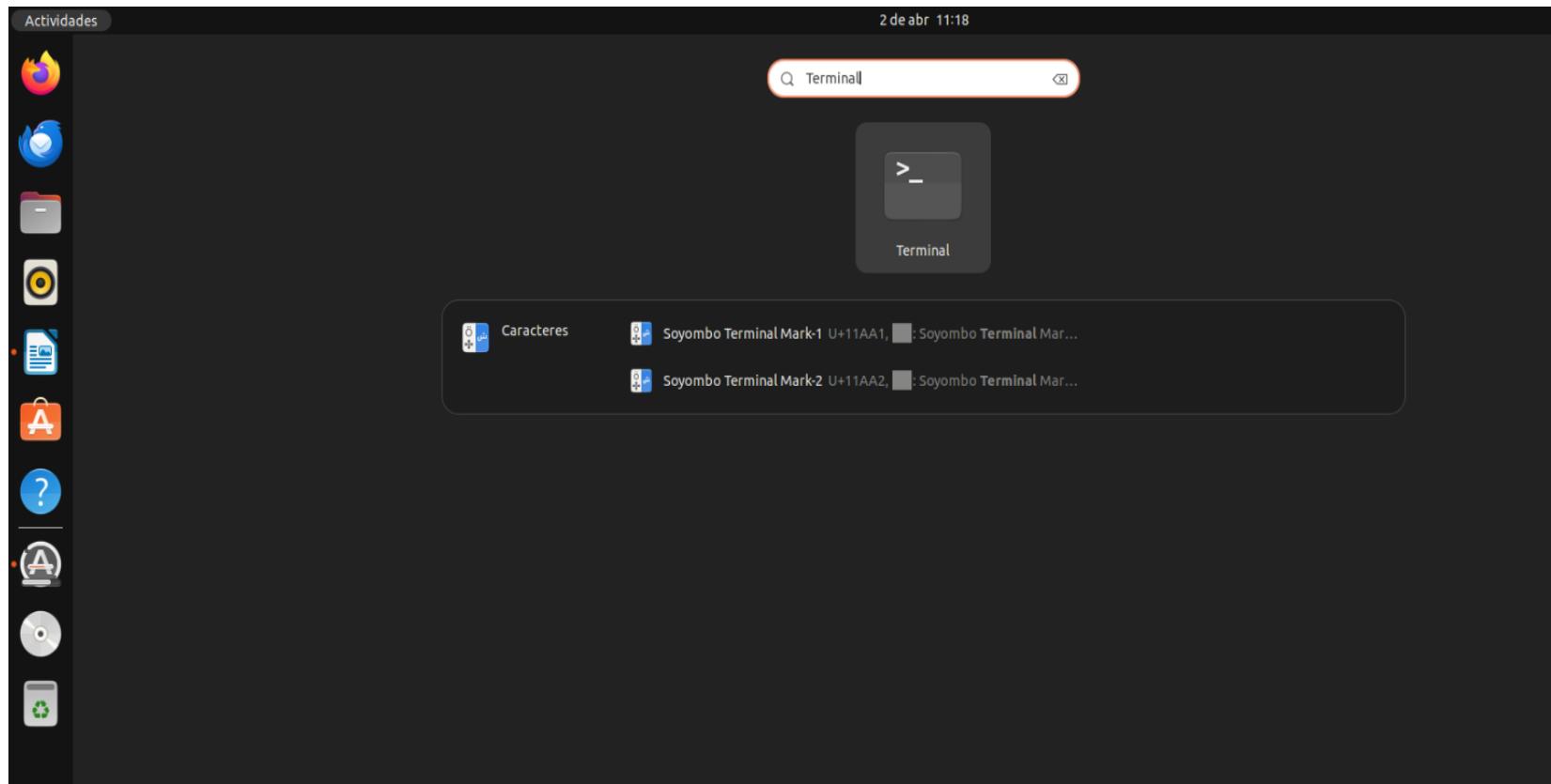
# Conceptos básicos. Shell de comandos

Para acceder al intérprete de comandos, se pulsa Ctrl+Alt+T o bien se pulsa en un ícono que aparece en la parte inferior izquierda, dentro de la barra de herramientas inferior del escritorio de ubuntu

Para acceder al intérprete de comandos, se pulsa Ctrl+Alt+T, o bien se pulsa en un ícono que aparece en la parte inferior izquierda, dentro de la barra de herramientas inferior del escritorio de ubuntu

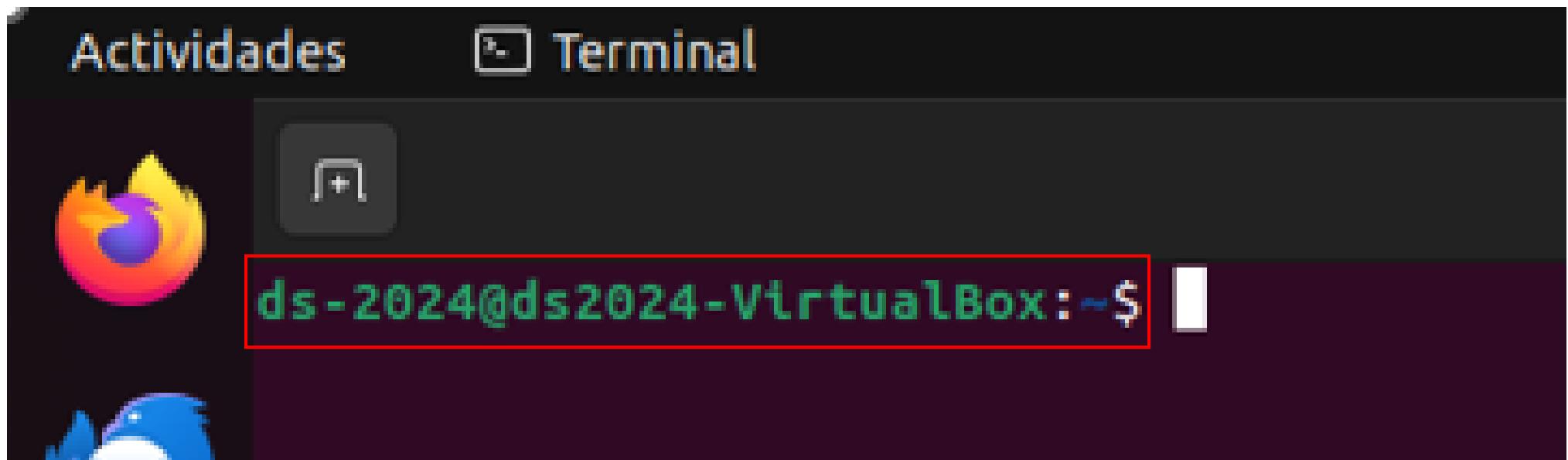


o bien se pulsa en un icono  
que aparece en la parte  
inferior izquierda,  
de la barra de  
herramientas inferior del  
escritorio de Ubuntu



# Algunas características generales son:

El terminal muestra en pantalla un indicador de línea de órdenes, denominado “prompt”, para que el usuario introduzca una. El indicador finaliza con un carácter \$ en el caso de usuarios normales y con # en el caso del superusuario.



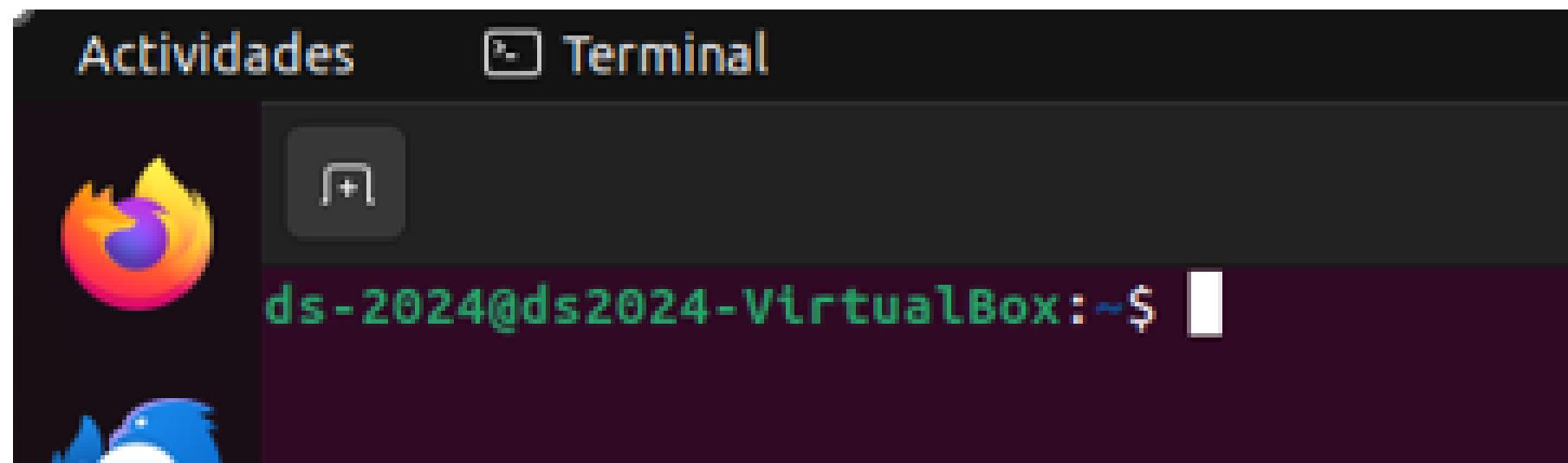
Al comienzo de la línea de órdenes aparece el usuario en la forma **usuario@máquina:directorio\$**, donde:

“usuario” es el nombre del usuario logado.

“máquina” es el nombre asignado a la máquina.

“directorio” es la ruta, dentro del sistema de ficheros, en la que se encuentra el usuario en ese momento.

El carácter especial “~” se utiliza para referirse al “HOME” del usuario, que normalmente se corresponde con la ruta “/home/<usuario>”. Es el espacio de ficheros dentro del sistema global asignado a ese usuario

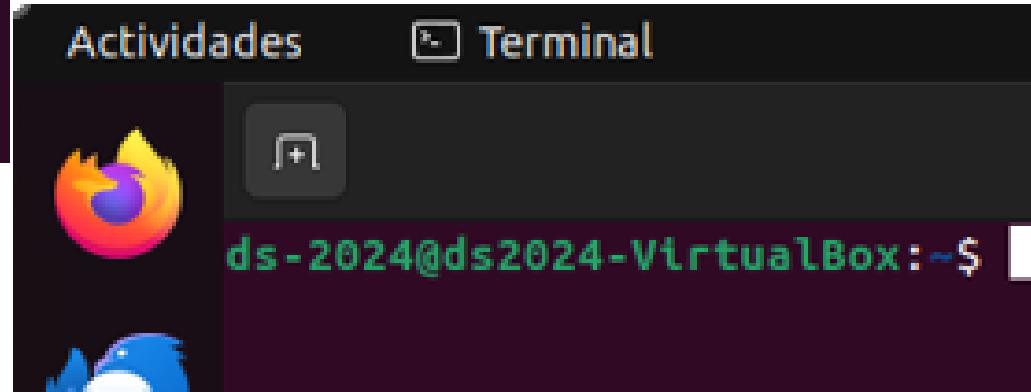
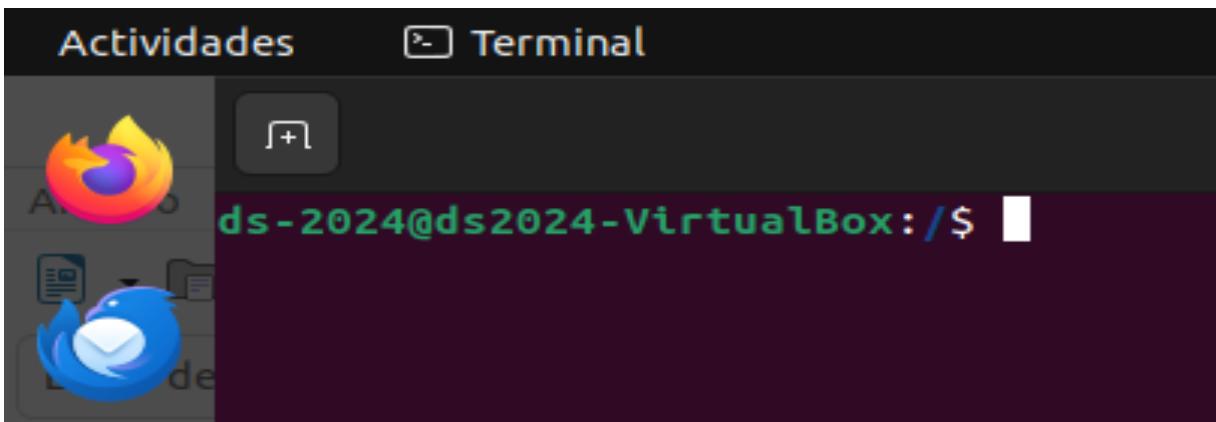


# Algunas características generales son:

- Cuando se escribe un comando para que se ejecute, hay que pulsar la tecla Enter.
- Los comandos hay que teclearlos exactamente. En este sentido, las letras mayúsculas y minúsculas se consideran caracteres diferentes.
- Un amplio número de comandos de la shell está pensado para trabajar directamente con el sistema de ficheros (o sistema de archivos). Por tanto, es conveniente entender cómo se organiza este sistema y cómo la shell da información sobre él.

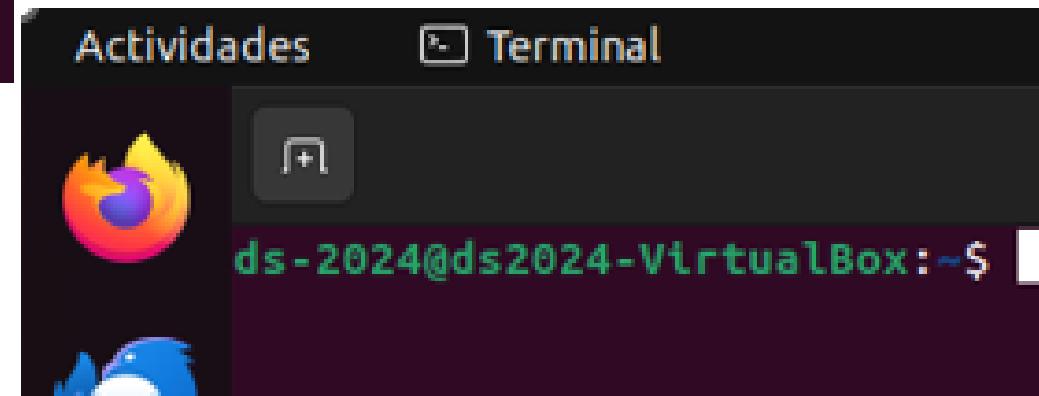
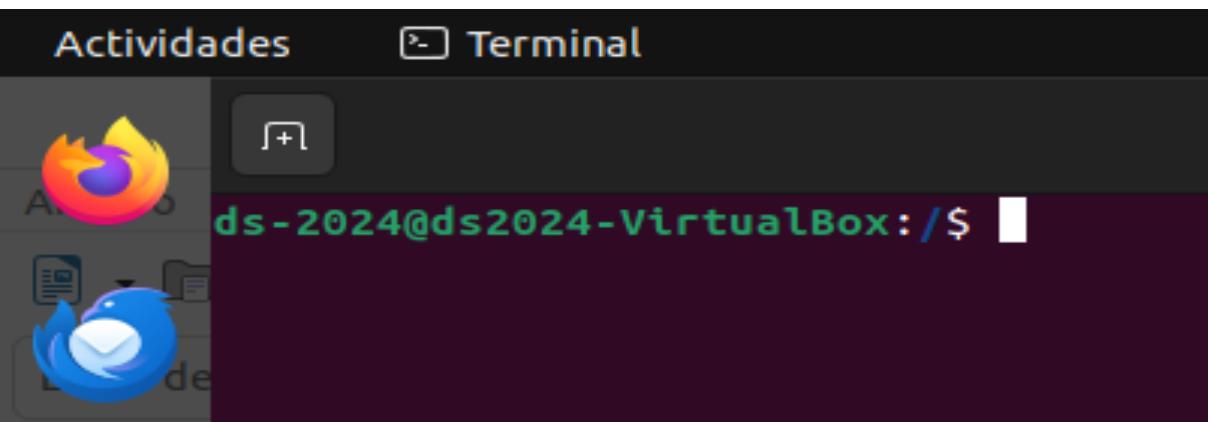
# Algunas características generales son:

- El directorio raíz es “/” y es el punto de partida hacia cualquier otro fichero o directorio del sistema.
- La jerarquía de directorios se visualiza mediante rutas, donde se muestra el camino, de izquierda a derecha, que se ha seguido para llegar a un directorio. Por ejemplo, si hay un directorio “prueba1” en el HOME, el camino o “PATH” seguido será: “/home/usuario/prueba1”.



# Algunas características generales son:

- El directorio raíz es “/” y es el punto de partida hacia cualquier otro fichero o directorio del sistema.
- La jerarquía de directorios se visualiza mediante rutas, donde se muestra el camino, de izquierda a derecha, que se ha seguido para llegar a un directorio. Por ejemplo, si hay un directorio “prueba1” en el HOME, el camino o “PATH” seguido será: “/home/usuario/prueba1”.



# Estructura de los directorios

- \* En el sistema de ficheros de UNIX (y similares, como GNU/Linux), existen varias sub-jerarquías de directorios que poseen múltiples y diferentes funciones de almacenamiento y organización en todo el sistema. Estos directorios pueden clasificarse en:
- \* ° Estáticos: Contiene archivos que no cambian sin la intervención del administrador (root), sin embargo, pueden ser leídos por cualquier otro usuario. (/bin, /sbin, /opt, /boot, /usr/bin...)

# Estructura

- \* <<sup>o</sup> Dinámicos: Contiene archivos que son cambiantes, y pueden leerse y escribirse (algunos sólo por su respectivo usuario y el root). Contienen configuraciones, documentos, etc. (/var/mail, /var/spool, /var/run, /var/lock, /home...)
- \* <<sup>o</sup> Compartidos: Contiene archivos que se pueden encontrar en un ordenador y utilizarse en otro, o incluso compartirse entre usuarios.

# Estructura

- \* <<sup>o</sup> Restringidos: Contiene ficheros que no se pueden compartir, solo son modificables por el administrador. (/etc, /boot, /var/run, /var/lock...)

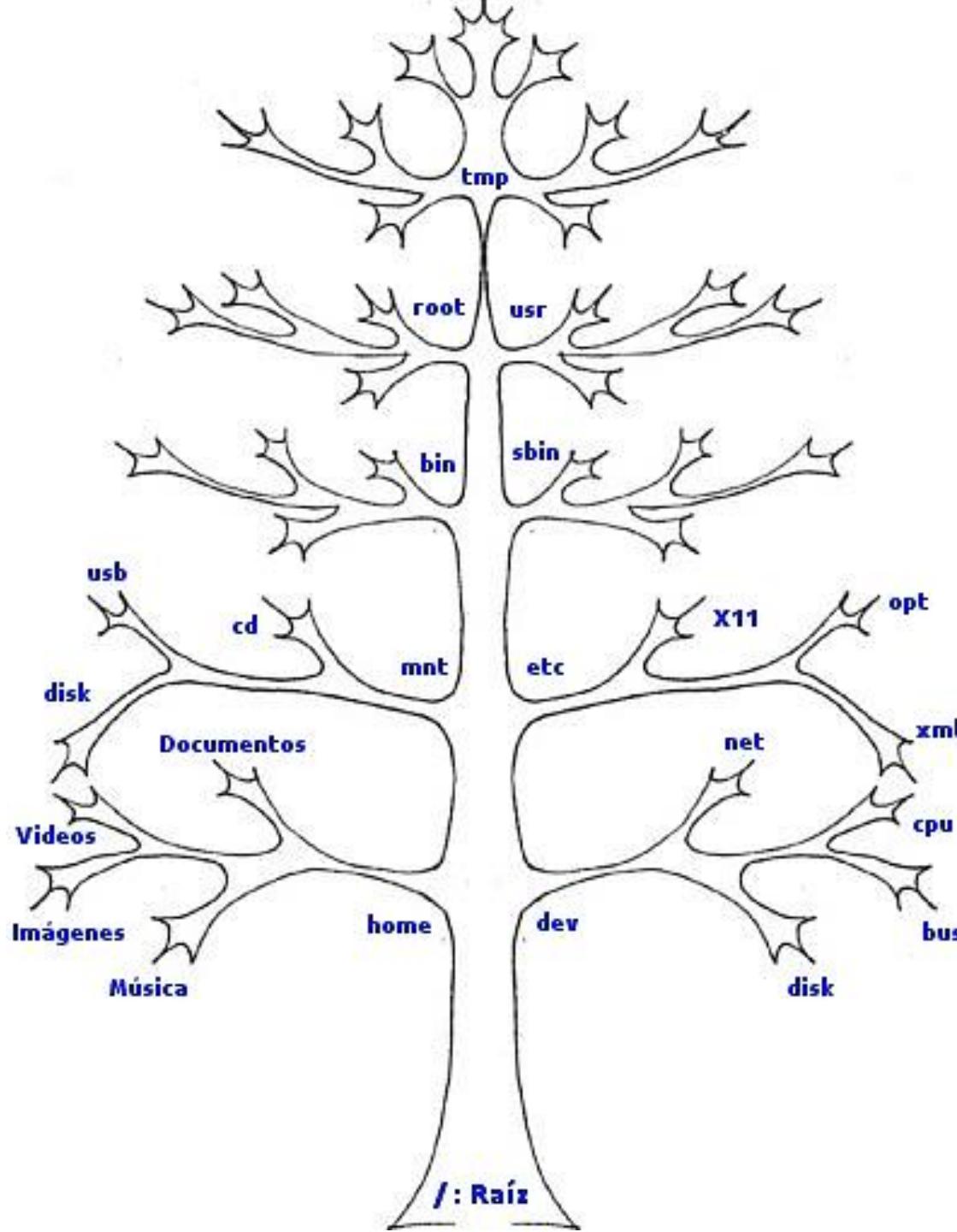
# Estructura

- \* root: es el nombre convencional de la cuenta de usuario que posee todos los derechos en todos los modos (mono o multi usuario). root es también llamado superusuario. Normalmente esta es la cuenta de administrador. El usuario root puede hacer muchas cosas que un usuario común no puede, tales como cambiar el dueño o permisos de archivos y enlazar a puertos de numeración pequeña.

# Estructura

- \* No es recomendable utilizar el usuario root para una simple sesión de uso habitual, ya que pone en riesgo el sistema al garantizar acceso privilegiado a cada programa en ejecución. Es preferible utilizar una cuenta de usuario normal y utilizar el comando su para acceder a los privilegios de root de ser necesario

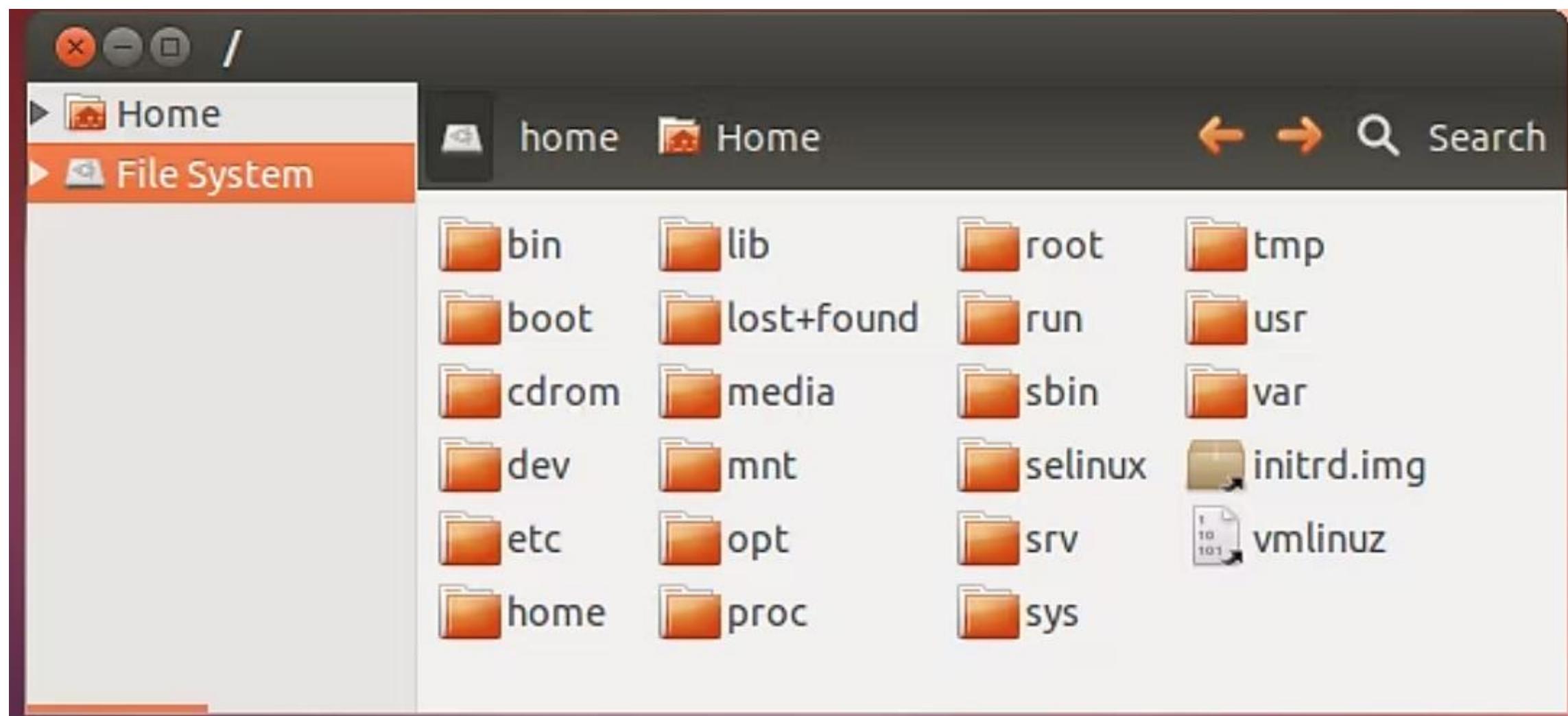
Donde la raíz del árbol (/) es la base de toda la estructura de directorios y las ramas (directorios y archivos) surgen o cuelgan de dicha base.



# Estructura Standard

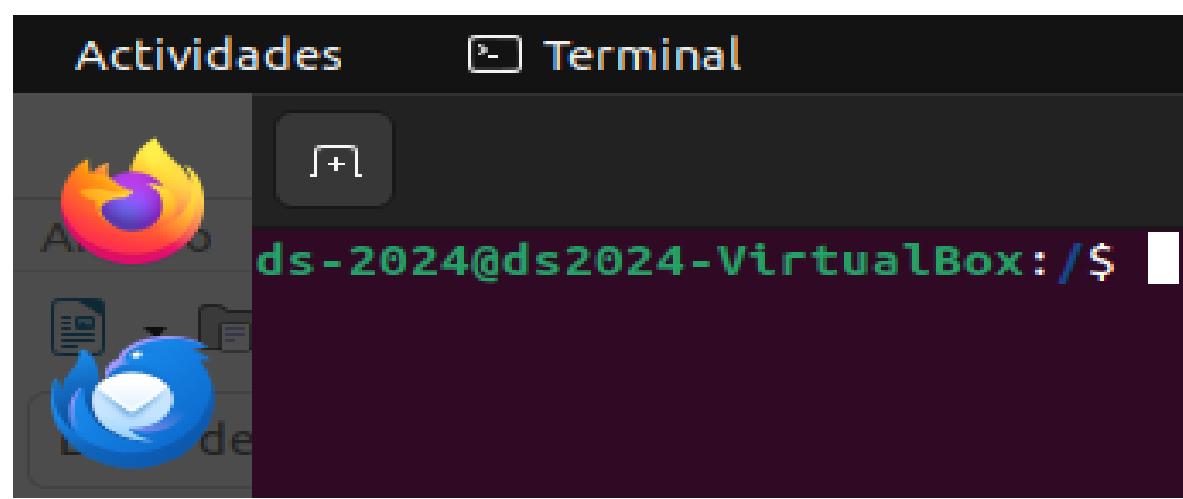


# Sistema de archivos de Linux



# Estructura Raíz ‘/’

/ (raíz): Parecido a el directorio raíz “C:\” de los sistemas operativos DOS y Windows. Es el nivel más alto dentro de la jerarquía de directorios, es el contenedor de todo el sistema (accesos al sistema de archivos, incluyendo los discos extraíbles [CD’s, DVD’s, pendrives, etc.]).



# Estructura bin

/bin (binarios): Los binarios son los ejecutables de Linux (similar a los archivos.exe de Windows). Aquí tendremos los ejecutables de los programas propios del sistema operativo.

/bin/

Comandos binarios esenciales  
de usuario

# Estructura boot

El directorio `/boot` contiene los archivos necesarios para arrancar el sistema, por ejemplo, los archivos del gestor de arranque GRUB.

Por lo tanto, en `/boot` se almacenan datos que se utilizan antes de que el kernel comience a ejecutar programas en modo usuario. Esto puede incluir sectores de arranque maestro guardados.

`/boot/`

Archivos estáticos  
del selector de arranque

# Estructura boot

Núcleo o kernel: es un software que constituye la parte más importante del sistema operativo. Es el principal responsable de facilitar a los distintos programas acceso seguro al hardware de la computadora o en forma básica, es el encargado de gestionar recursos, a través de servicios de llamada al sistema

/boot/

Archivos estáticos  
del selector de arranque

# Estructura dev

/dev (dispositivos): Esta carpeta contiene los dispositivos del sistema, incluso los que no se les ha asignado (montado) un directorio, por ejemplo: micrófonos, impresoras, pendrives (memorias USB) y dispositivos especiales (por ejemplo, /dev/null). Linux trata los dispositivos como si fueran un fichero más para facilitar el flujo de la información.

/dev/

Archivos de unidades

# Estructura etc

/etc (etcétera): Aquí se guardan los ficheros de configuración de los programas instalados, así como ciertos scripts que se ejecutan en el inicio del sistema. Los valores de estos ficheros de configuración pueden ser complementados o sustituidos por los ficheros de configuración de usuario que cada uno tiene en su respectivo “home” (carpeta personal).

/etc/

Configuración de sistema  
de Host específico  
Directorios requeridos: opt, X11, sgml, xml

# Estructura etc

Ejemplos:

/etc/opt/ Archivos de configuración para los programas alojados dentro del directorio /opt.

/etc/X11/ Archivos de configuración para el X Window System, versión 11.

/etc/

Configuración de sistema  
de Host específico  
Directorios requeridos: opt, X11, sgml, xml

# Estructura lib

/lib (bibliotecas): Contiene las bibliotecas (mal conocidas como librerías) esenciales compartidas de los programas alojados, es decir, para los binarios en /bin/ y /sbin/, las bibliotecas para el núcleo, así como módulos y controladores (drivers).

/lib/

Librerías esenciales compartidas  
y módulos de Kernel

# Estructura opt

/opt (opcionales): Contiene paquetes de programas opcionales de aplicaciones estáticas, es decir, que pueden ser compartidas entre los usuarios. Dichas aplicaciones no guardan sus configuraciones en este directorio; de esta manera, cada usuario puede tener una configuración diferente de una misma aplicación, de manera que se comparte la aplicación pero no las configuraciones de los usuarios, las cuales se guardan en su respectivo directorio en /home.

# Estructura usr

/usr: El directorio /usr es la segunda sección más importante del sistema de archivos, cuyos datos se pueden compartir y son de solo lectura. Contiene aplicaciones y archivos utilizados por los usuarios, a diferencia de las aplicaciones y archivos utilizados por el sistema.



Utilidades y aplicaciones de (Multi-)usuario  
Jerarquía secundaria  
Directorios requeridos: bin, include, lib, local, sbin, share

# Estructura usr

Por ejemplo, las aplicaciones no fundamentales se encuentran dentro del directorio /usr/bin en lugar del directorio /bin y los binarios de administración del sistema no fundamentales se encuentran en el directorio /usr/sbin en lugar del directorio /sbin.

/usr/

Utilidades y aplicaciones de (Multi-)usuario  
Jerarquía secundaria  
Directorios requeridos: bin, include, lib, local, sbin, share

# Estructura usr

Las bibliotecas para cada uno se encuentran dentro del directorio /usr/lib. El directorio /usr a su vez contiene otros directorios, por ejemplo /usr/share. El directorio /usr/local es donde se instalan las aplicaciones compiladas localmente de forma predeterminada, esto evita que pueda afectar al resto del sistema.



Utilidades y aplicaciones de (Multi-)usuario  
Jerarquía secundaria  
Directarios requeridos: bin, include, lib, local, sbin, share

# Estructura usr

usr/share: Archivos compartidos como ficheros de configuración, imágenes, iconos, themes, etc.

/usr/X11R6/ Sistema X Window System, Versión 11, Release 6.  
Este directorio se relaciona con el entorno gráfico



# Estructura usr

/usr/src: Códigos fuente de algunas aplicaciones y del kernel Linux. Al igual que /mnt, esta carpeta es manejada por los usuarios directamente para que éstos puedan guardar en él el código fuente de programas y bibliotecas y así puedan accesarlo fácilmente, sin problemas con permisos. Permite que el código fuente tenga un espacio propio, accesible pero apartado de todos los usuarios.



Utilidades y aplicaciones de (Multi-)usuario  
Jerarquía secundaria  
Directorio requerido: bin, include, lib, local, sbin, share

# Estructura var

/var (variables): Archivos variables, tales como logs, archivos spool, bases de datos, archivos de e-mail temporales, y algunos archivos temporales en general. Generalmente actúa como un registro del sistema. Ayuda a encontrar los orígenes de un problema.

**/var/**

**Variables de archivo**

# Estructura var

/var/crash/ Se depositan datos e información, referentes a las caídas o errores del sistema operativo. Es más específico que /var en general.

/var/lib: Información sobre el estado actual de las aplicaciones, modificable por las propias aplicaciones.

/var/

Variables de archivo

# Estructura var

/var/log: Es uno de los subdirectorios más importantes ya que aquí se guardan todo tipo de logs del sistema.

/var/

Variables de archivo

# Parámetros

Los comandos de la Shell de Linux pueden recibir parámetros que afectan a la manera de ejecutarse:  
comando parámetro1 parámetro2 ... parámetroN.

Los parámetros son valores o argumentos que se pueden pasar a los comandos o scripts para modificar su comportamiento o salida.

Permiten a los usuarios interactuar con los comandos y scripts de manera más dinámica y flexible.

Por ejemplo, en el comando `ls -l /home/usuario`

`-l` es un parámetro que le indica al comando `ls` que liste los archivos y directorios con información detallada, y  
`/home/usuario` es un parámetro que especifica el directorio cuyo contenido se debe listar.

# Características generales: Modificadores

Además de parámetros, los comandos pueden recibir modificadores propios, que van siempre precedidos de los símbolos “-” y “--”.

El primero se utiliza para indicar modificadores cuyo nombre es un único carácter, mientras que el segundo precede siempre a nombres completos de modificadores.

Los modificadores, también conocidos como opciones o flags, son parámetros especiales que se utilizan para cambiar el comportamiento de los comandos

# Modificadores

Ejemplo con el comando ls:

ls -l: Muestra información detallada sobre los archivos, como permisos, propietario, tamaño y fecha de modificación.

ls -a: Muestra todos los archivos, incluidos los archivos ocultos.

ls -h: Muestra los tamaños de los archivos en formato legible por humanos (por ejemplo, "10K" para 10 kilobytes).

ls -t: Ordena los archivos por fecha de modificación, con los archivos más recientes al principio.

# Modificadores

Ejemplo con el comando **ls** pero con el doble guión ‘--’:

**ls --all**: Es equivalente a **ls -a**, muestra todos los archivos, incluidos los archivos ocultos.

**ls --color**: Muestra los nombres de los archivos con diferentes colores para indicar su tipo.

# Modificadores

Asimismo, es habitual que coexistan ambas versiones de un mismo modificador. Por ejemplo, un buen número de comandos ofrecen estos dos modificadores, que hacen lo mismo: -v y --verbose (hacen que la salida del comando contenga información adicional). Se podría utilizar uno u otro, indistintamente.

Ejemplo `grep -i --color=auto 'ciclo' archivo.txt`

En este caso, se utilizan los siguientes modificadores:

**-i: Es el modificador corto que indica que la búsqueda no es sensible a mayúsculas/minúsculas**

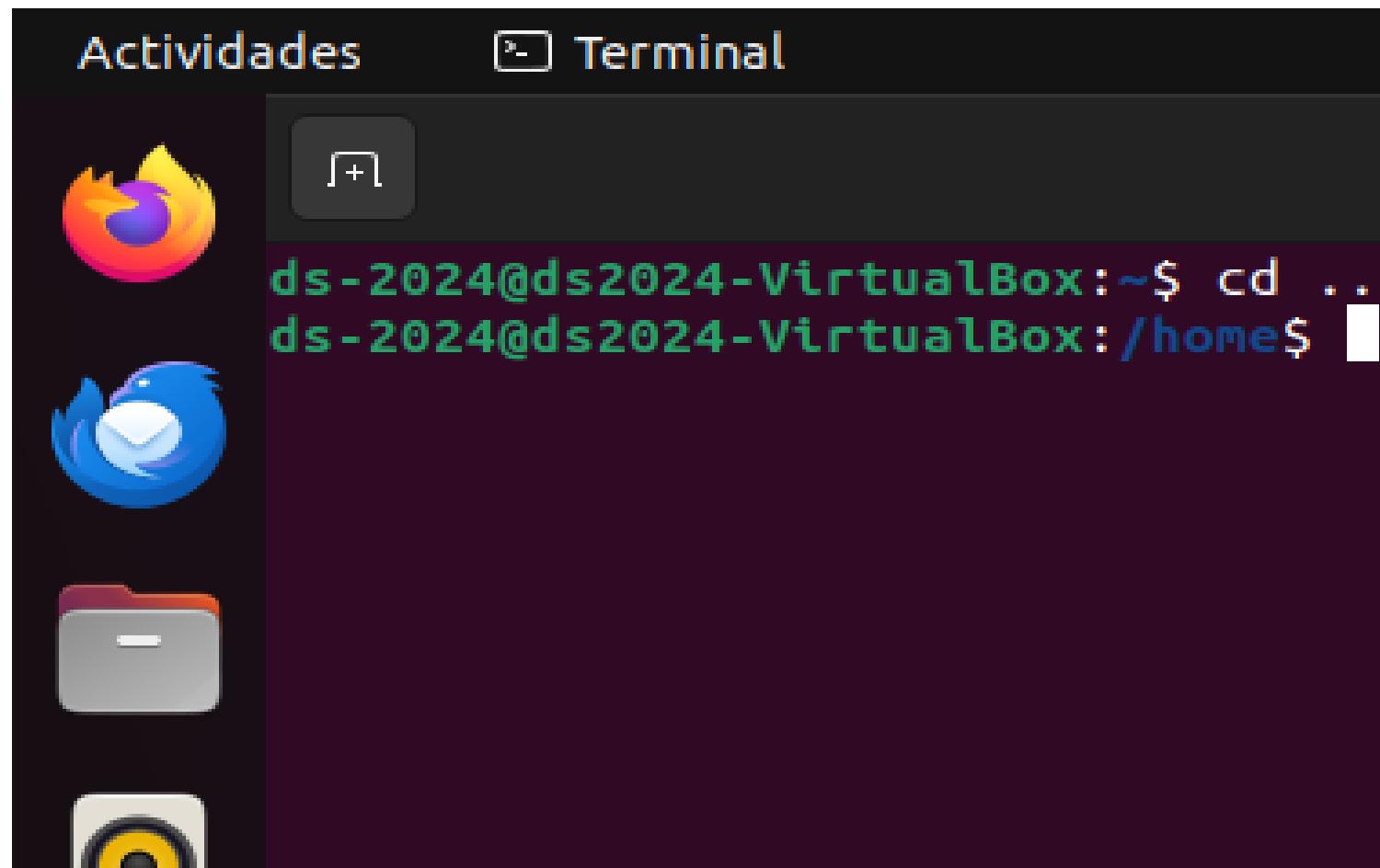
**--color=auto: Es el modificador largo que resalta los resultados coincidentes en color**

## Actividad 5.

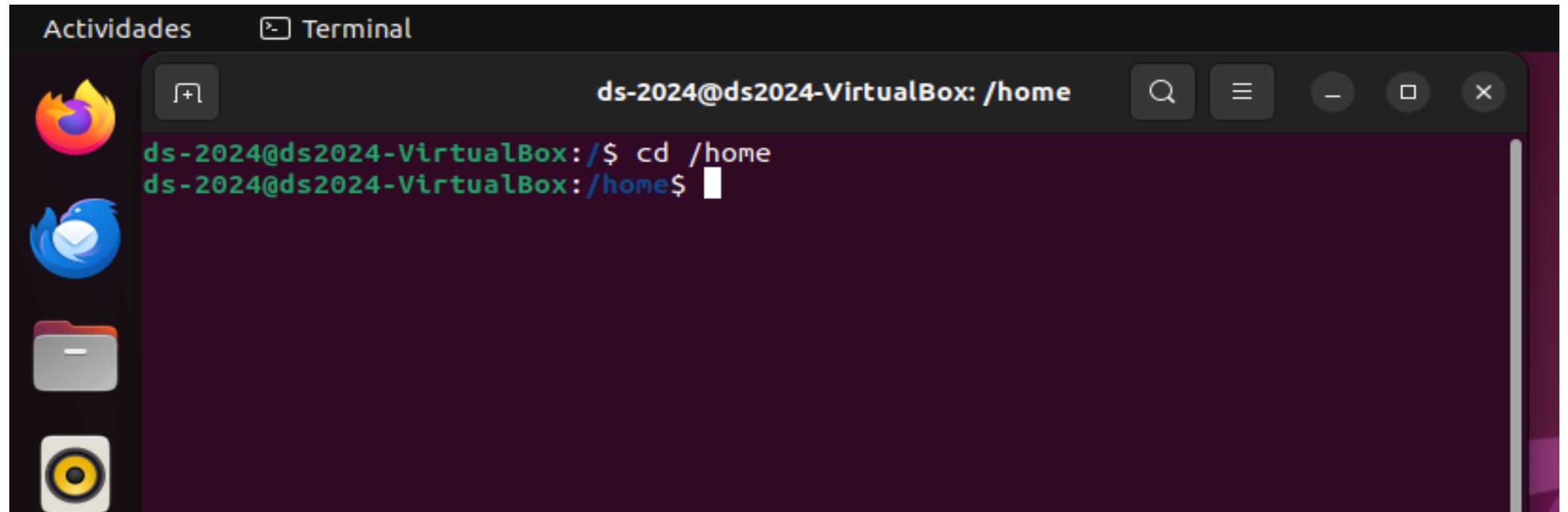
A partir de las siguiente diapositiva hasta la página 117 debe repetir cada uno de los comandos que se explican con tus propios ficheros. Una vez terminado, exportar el histórico de comandos utilizado en un fichero .txt u otro similar.

# Comandos básicos de navegación por el sistema de archivos

Cambio de directorio: `cd ..`

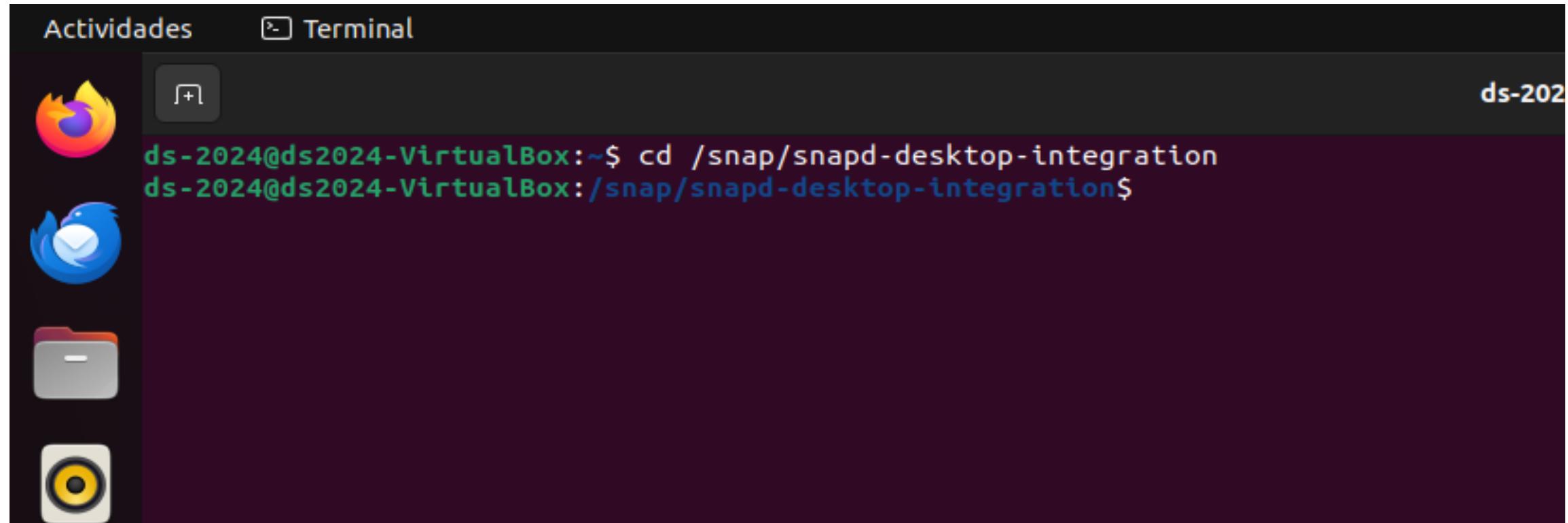


También se puede escribir cd con la ruta a llegar



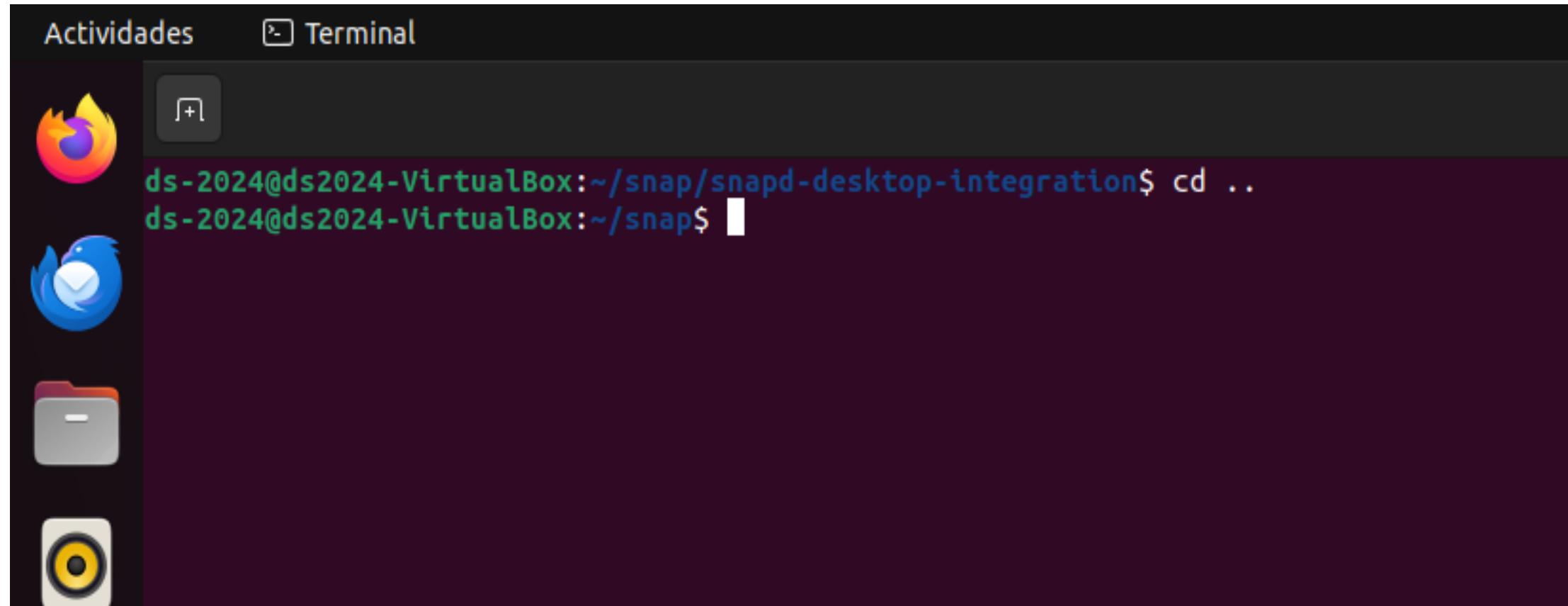
También se puede escribir cd con la ruta a llegar.

Otro ejemplo:



Asimismo, dentro de cada directorio existen dos directorios especiales: “.” y “..”

“.” hace referencia al directorio actual, mientras que  
“..” se refiere al directorio padre del actual.

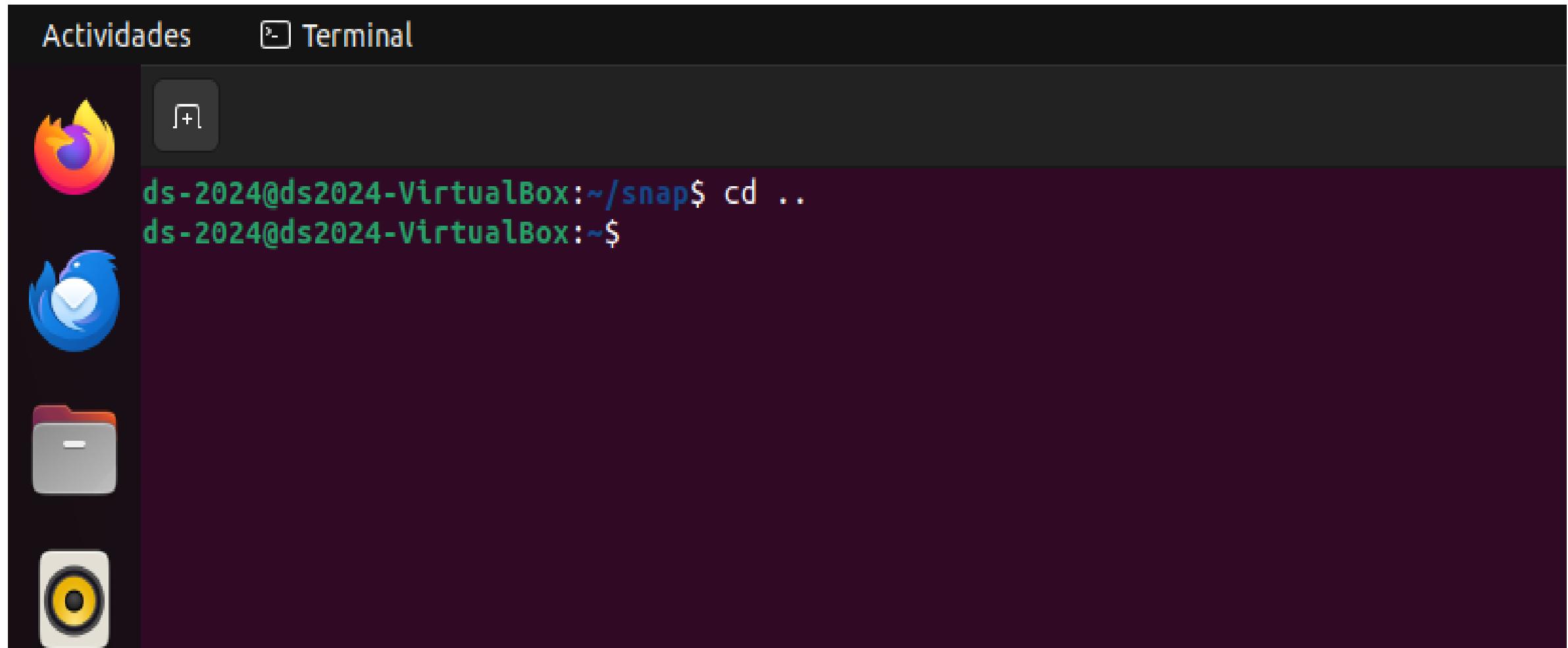
A screenshot of a Linux desktop environment. On the left, there's a dock with icons for a web browser (Firefox), a file manager (Nautilus), and other applications. In the center, a terminal window is open with the following text:

```
ds-2024@ds2024-VirtualBox:~/snap/snapd-desktop-integration$ cd ..  
ds-2024@ds2024-VirtualBox:~/snap$
```

The terminal window has a dark background and light-colored text. The desktop background is also dark.

Vuelve a escribir cd ..

Actividades Terminal

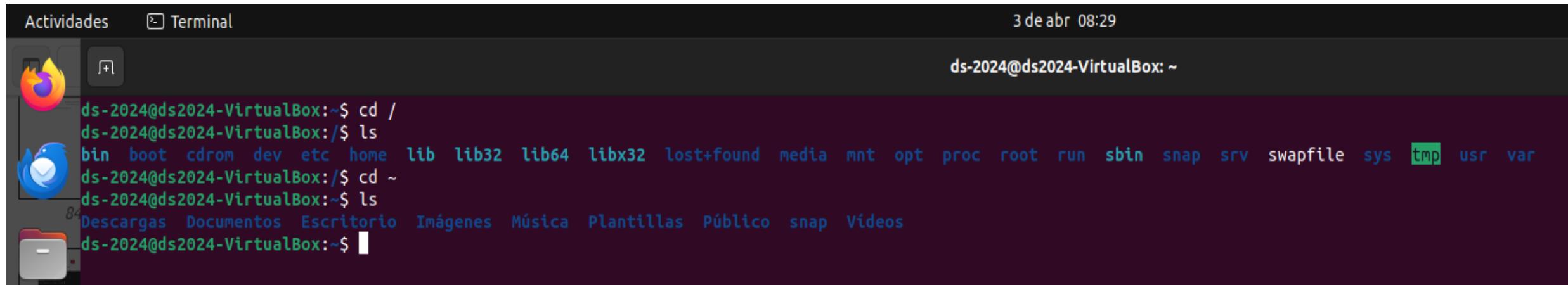


The image shows the Ubuntu Dash interface. On the left, there is a vertical sidebar with icons for various applications: a yellow fox (Firefox), a blue bird (Thunderbird), a grey folder (File Manager), and a speaker (Sound). To the right of the sidebar is a dark purple terminal window titled "Terminal". The terminal window has a title bar with the word "Terminal" and a close button. The main area of the terminal shows the following command-line session:

```
ds-2024@ds2024-VirtualBox:~/snap$ cd ..  
ds-2024@ds2024-VirtualBox:~$
```

cd ~ : Me lleva al directorio home

cd / : Me lleva al directorio raíz



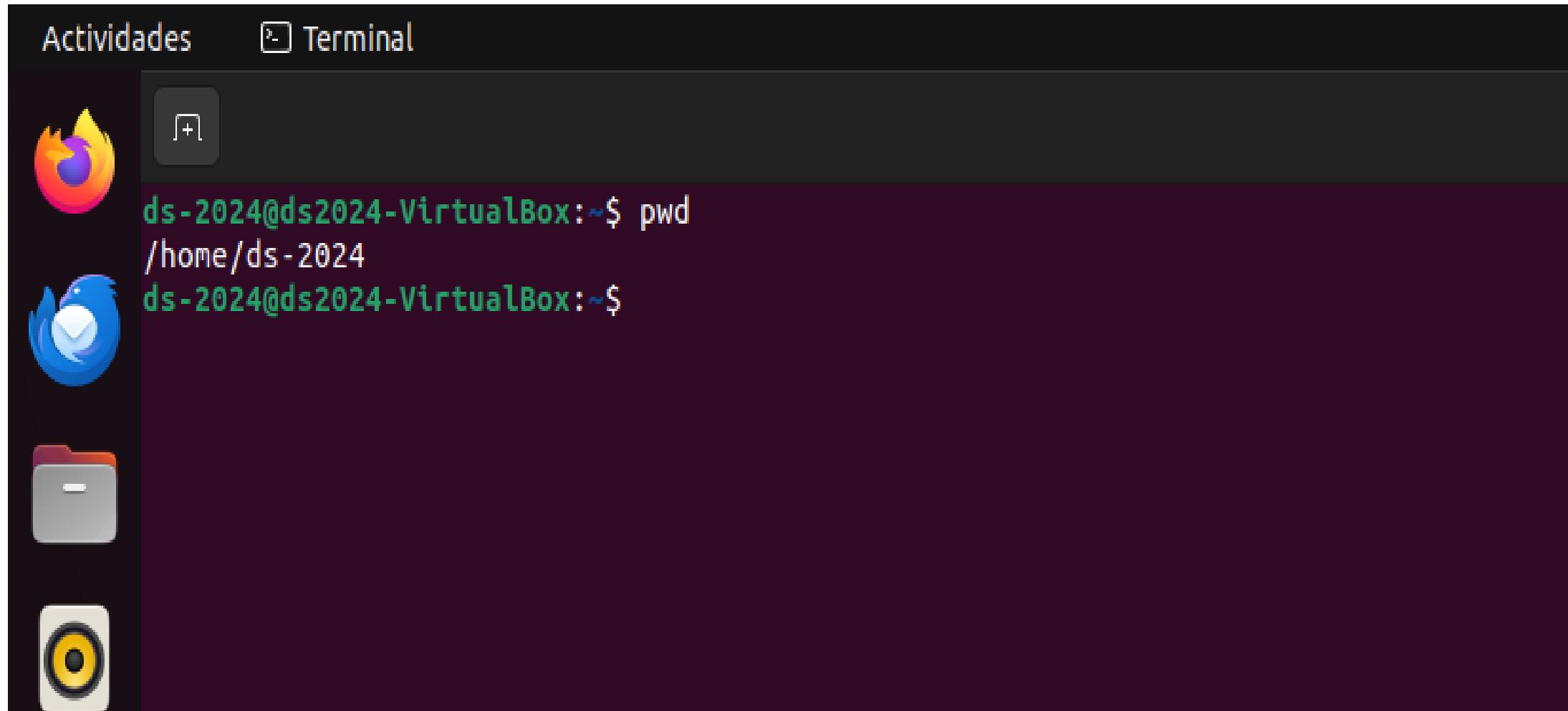
The image shows a screenshot of a Linux desktop environment, likely Ubuntu, with a dark theme. At the top, there is a dock with icons for the Dash (home), Terminal, and other applications. The main window is a terminal window titled "Terminal". The terminal window has a dark background and light-colored text. It displays the following command-line session:

```
Actividades Terminal 3 de abr 08:29
ds-2024@ds2024-VirtualBox:~$ cd /
ds-2024@ds2024-VirtualBox:/$ ls
bin boot cdrom dev etc home lib lib32 lib64 libx32 lost+found media mnt opt proc root run sbin snap srv swapfile sys tmp usr var
ds-2024@ds2024-VirtualBox:/$ cd ~
ds-2024@ds2024-VirtualBox:~$ ls
Descargas Documentos Escritorio Imágenes Música Plantillas Público snap Vídeos
ds-2024@ds2024-VirtualBox:~$ █
```

Para saber mi ubicación actual : **pwd**

El comando **pwd** imprime la ruta del directorio actual

**pwd** es la abreviación de: print working directory

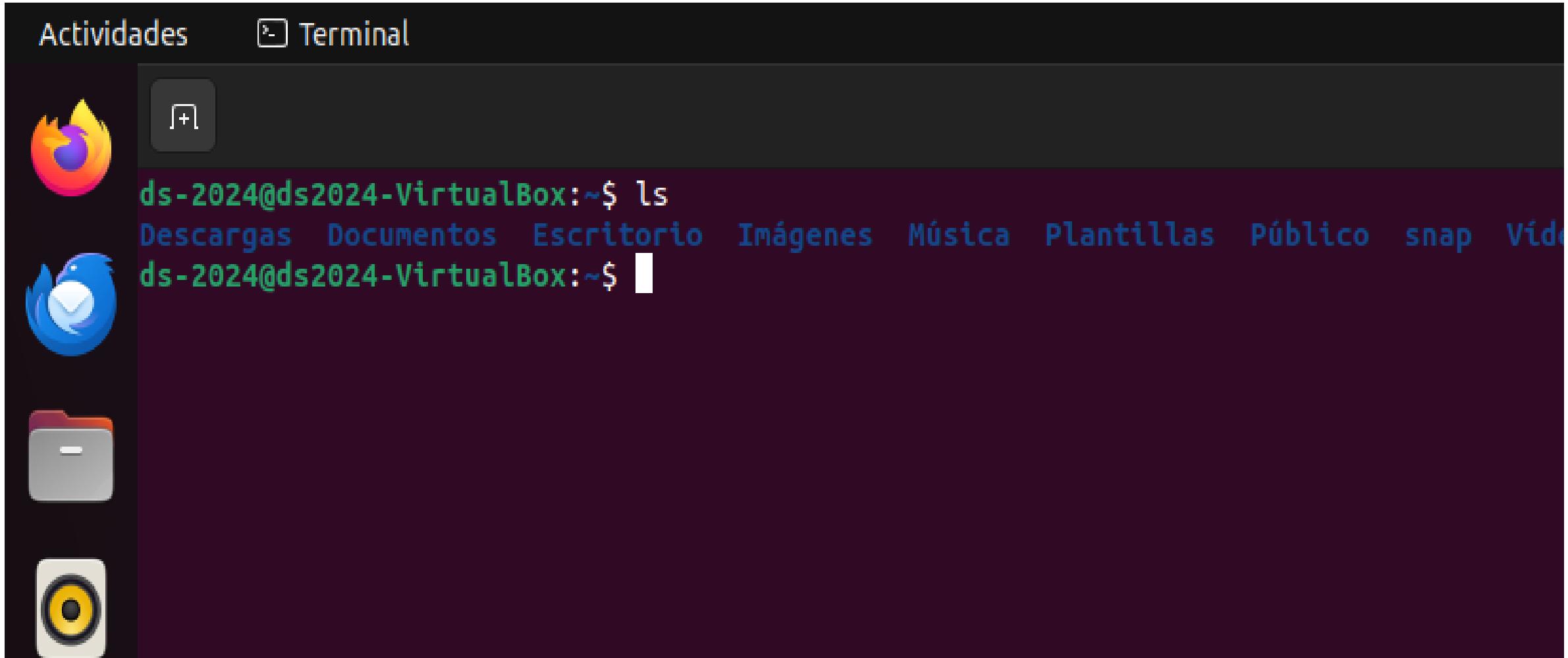


The screenshot shows a Linux desktop interface with a dark theme. At the top, there is a dock with several icons: a red and orange flame (Firefox), a blue bird (Thunderbird), a folder (File Manager), and a speaker (Speaker). To the right of the dock is a terminal window titled "Terminal". The terminal window has a dark background and contains the following text:

```
ds-2024@ds2024-VirtualBox:~$ pwd
/home/ds-2024
ds-2024@ds2024-VirtualBox:~$
```

## Listado del contenido de un directorio : ls

El comando ls muestra los nombres de los archivos y subdirectorios contenidos en el directorio en el que se está. Solo se obtienen los nombres de los archivos, sin ninguna otra información.

A screenshot of a Linux desktop environment. At the top, there is a dark header bar with the text "Actividades" and "Terminal". The terminal window is open and displays the following command and output:

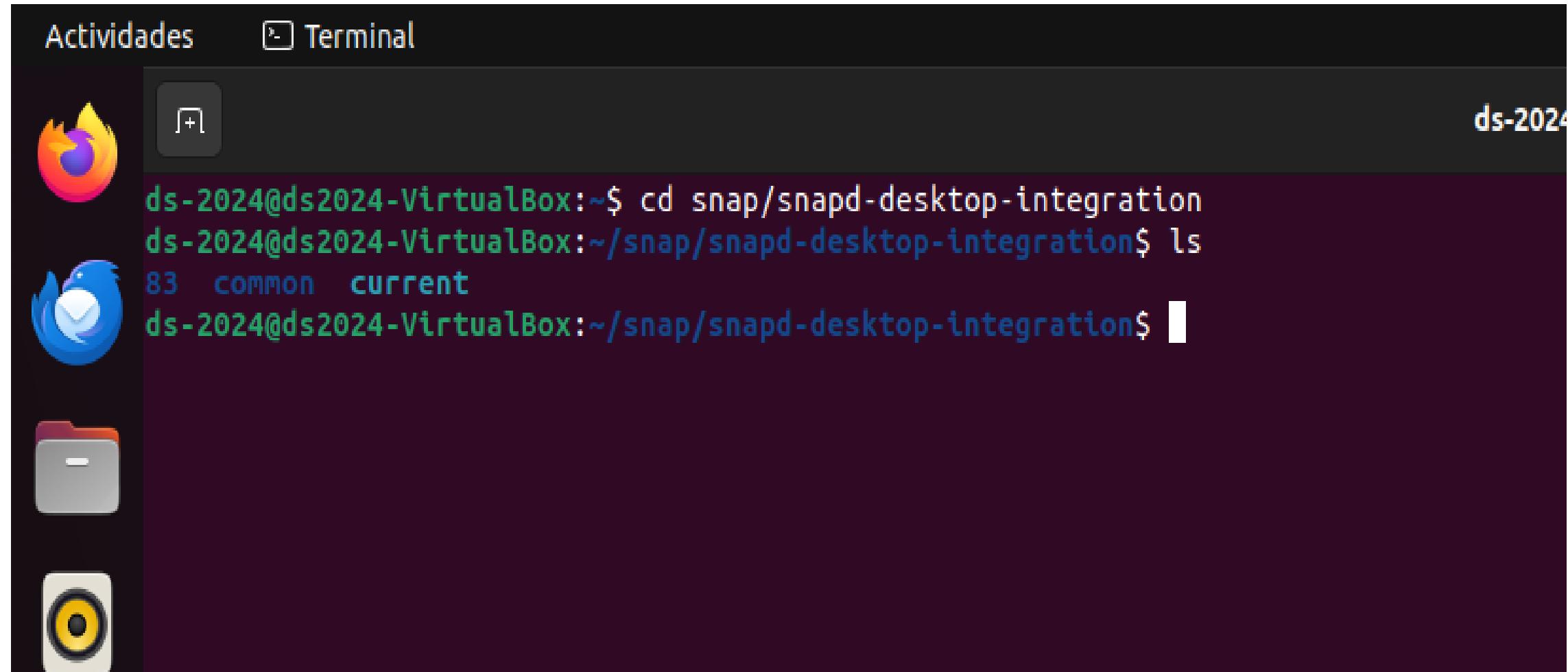
```
ds-2024@ds2024-VirtualBox:~$ ls
Descargas Documentos Escritorio Imágenes Música Plantillas Público snap Vídeos
ds-2024@ds2024-VirtualBox:~$ █
```

The desktop background is dark, and there are icons for the Dash, Home, and Sound settings in the bottom left corner. The terminal has a dark purple background and white text.

Actividades Terminal

```
ds-2024@ds2024-VirtualBox:~$ ls
Descargas Documentos Escritorio Imágenes Música Plantillas Público snap Vídeos
ds-2024@ds2024-VirtualBox:~$ █
```

Otro ejemplo:



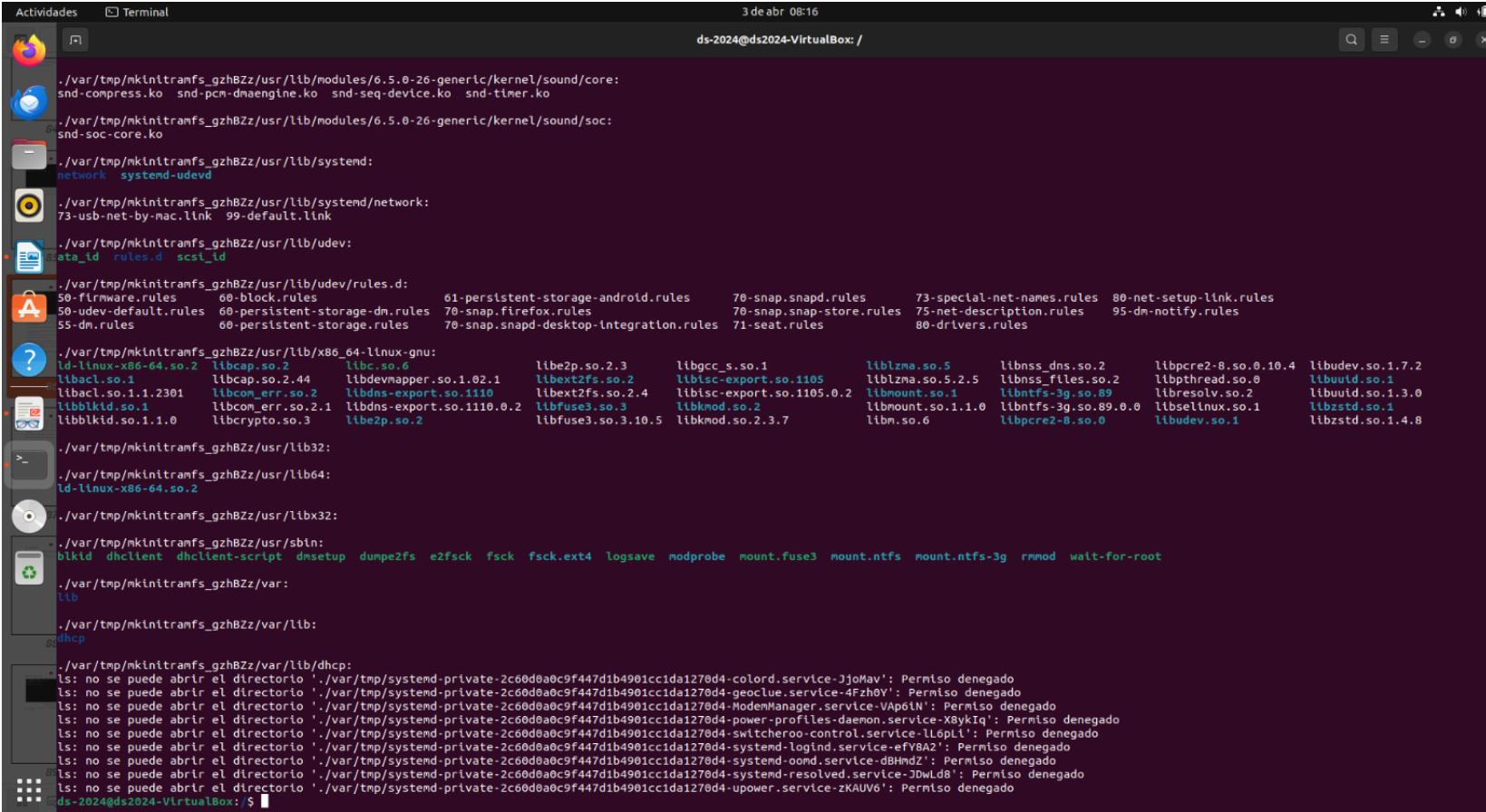
The screenshot shows a dark-themed desktop environment with a terminal window open. The terminal window has a dark background and light-colored text. It displays the following command-line session:

```
ds-2024@ds2024-VirtualBox:~$ cd snap/snapd-desktop-integration
ds-2024@ds2024-VirtualBox:~/snap/snapd-desktop-integration$ ls
83 common current
ds-2024@ds2024-VirtualBox:~/snap/snapd-desktop-integration$
```

The desktop interface includes a dock with icons for a browser (Firefox), a file manager, and a terminal. The top bar features a search icon and the user's name, "ds-2024".

# El comando ls admite diferentes modificadores que alterarán la información mostrada en pantalla.

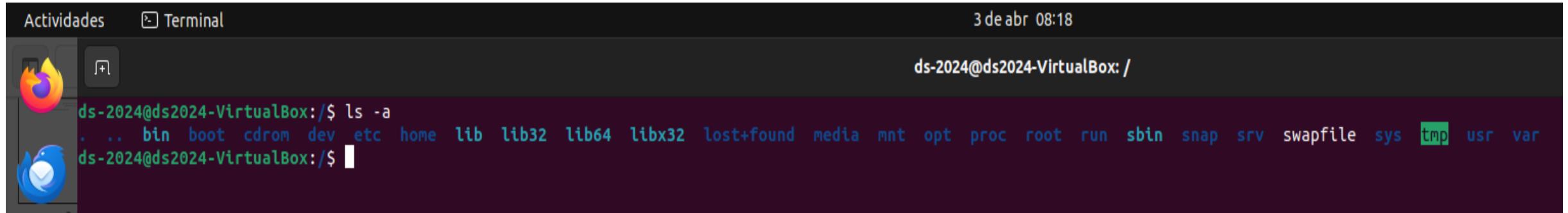
**ls -R** lista recursivamente los subdirectorios encontrados. Es decir, muestra los ficheros y directorios del actual, así como subdirectorios y su contenido:



The screenshot shows a terminal window titled "Terminal" with the command "ls -R" run on the root directory. The output lists all files and subdirectories in a recursive manner. The terminal interface includes a title bar with the date and time (3 de abr 08:16), a menu bar with "Actividades" and "Terminal", and a toolbar with icons for file operations. The background shows a dark desktop environment with various icons.

```
3 de abr 08:16
ds-2024@ds2024-VirtualBox: / [1]
.
./var/tmp/mkinitramfs_gzhBZz/usr/lib/modules/6.5.0-26-generic/kernel/sound/core:
snd-compress.ko snd-pcm-dmaengine.ko snd-seq-device.ko snd-timer.ko
snd-soc-core.ko
.
./var/tmp/mkinitramfs_gzhBZz/usr/lib/modules/6.5.0-26-generic/kernel/sound/soc:
.
./var/tmp/mkinitramfs_gzhBZz/usr/lib/systemd:
network systemd-udevd
.
./var/tmp/mkinitramfs_gzhBZz/usr/lib/systemd/network:
73-usb-net-by-mac.link 99-default.link
.
./var/tmp/mkinitramfs_gzhBZz/usr/lib/udev:
data_id.rules.d scsi_id
.
./var/tmp/mkinitramfs_gzhBZz/usr/lib/udev/rules.d:
50-firmware.rules 60-block.rules 61-persistent-storage-android.rules 70-snap.snapd.rules 73-special-net-names.rules 80-net-setup-link.rules
50-udev-default.rules 60-persistent-storage-dm.rules 70-snap.firefox.rules 70-snap.snap-store.rules 75-net-description.rules 95-dm-notify.rules
55-dm.rules 60-persistent-storage.rules 70-snap.snapd-desktop-integration.rules 71-seat.rules 80-drivers.rules
.
./var/tmp/mkinitramfs_gzhBZz/usr/lib/x86_64-linux-gnu:
ld-linux-x86-64.so.2 libcap.so.6 libgcc_s.so.1 liblzlma.so.5 libnss_dns.so.2 libpcre2-8.so.0.10.4 libudev.so.1.7.2
libacl.so.1 libcap.so.2.44 libdevmapper.so.1.02.1 libext2fs.so.2 libisc-export.so.1105 liblzma.so.5.2.5 libnss_files.so.2 libpthread.so.0 libbuid.so.1
libacl.so.1.2301 libcom_err.so.2 libdns-export.so.1110 libext2fs.so.2.4 libisc-export.so.1105.0.2 libmount.so.1 libnfs-3g.so.89 libresolv.so.2 libuvid.so.1.3.0
libblkid.so.1 libcom_err.so.2.1 libdns-export.so.1110.0.2 libfuse3.so.3 libkmod.so.2 libmount.so.1.1.0 libnfs-3g.so.89.0.0 libselinux.so.1 libzstd.so.1
libblkid.so.1.1.0 libcrypto.so.3 libe2p.so.2 libfuse3.so.3.10.5 libkmod.so.2.3.7 libm.so.6 libpcre2-8.so.0 libudev.so.1 libzstd.so.1.4.8
.
./var/tmp/mkinitramfs_gzhBZz/usr/lib32:
.
./var/tmp/mkinitramfs_gzhBZz/usr/lib64:
ld-linux-x86-64.so.2
.
./var/tmp/mkinitramfs_gzhBZz/usr/libx32:
.
./var/tmp/mkinitramfs_gzhBZz/usr/sbin:
blkid dhclient_dhclient-script dmsetup dumpe2fs e2fsck fsck fsck.ext4 logsave modprobe mount.fuse3 mount.ntfs mount.ntfs-3g rmmod wait-for-root
.
./var/tmp/mkinitramfs_gzhBZz/var:
lib
.
./var/tmp/mkinitramfs_gzhBZz/var/lib:
dhclient
.
./var/tmp/mkinitramfs_gzhBZz/var/lib/dhcp:
ls: no se puede abrir el directorio './var/tmp/systemd-private-2c60d0a0c9f447dib4901cc1da1270d4-colord.service-JjoMav': Permiso denegado
ls: no se puede abrir el directorio './var/tmp/systemd-private-2c60d0a0c9f447dib4901cc1da1270d4-geoclue.service-4Fzh0Y': Permiso denegado
ls: no se puede abrir el directorio './var/tmp/systemd-private-2c60d0a0c9f447dib4901cc1da1270d4-ModemManager.service-Vap0tN': Permiso denegado
ls: no se puede abrir el directorio './var/tmp/systemd-private-2c60d0a0c9f447dib4901cc1da1270d4-power-profiles-daemon.service-XBykIq': Permiso denegado
ls: no se puede abrir el directorio './var/tmp/systemd-private-2c60d0a0c9f447dib4901cc1da1270d4-switcheroo-control.service-L6pL1': Permiso denegado
ls: no se puede abrir el directorio './var/tmp/systemd-private-2c60d0a0c9f447dib4901cc1da1270d4-systemd-logind.service-eFyBA2': Permiso denegado
ls: no se puede abrir el directorio './var/tmp/systemd-private-2c60d0a0c9f447dib4901cc1da1270d4-systemd-service-eFyBA2': Permiso denegado
ls: no se puede abrir el directorio './var/tmp/systemd-private-2c60d0a0c9f447dib4901cc1da1270d4-systemd-oomd.service-dBHamZ': Permiso denegado
ls: no se puede abrir el directorio './var/tmp/systemd-private-2c60d0a0c9f447dib4901cc1da1270d4-systemd-resolved.service-JDwId8': Permiso denegado
ls: no se puede abrir el directorio './var/tmp/systemd-private-2c60d0a0c9f447dib4901cc1da1270d4-upower.service-zKAUV6': Permiso denegado
.
ds-2024@ds2024-VirtualBox: $
```

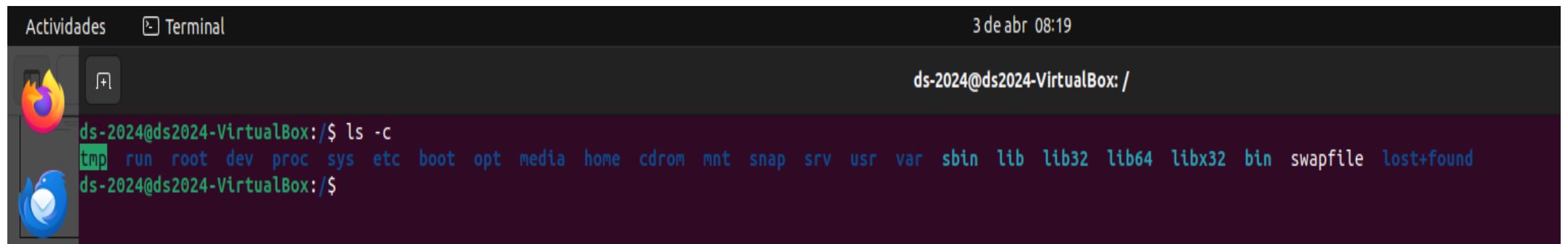
**ls -a** muestra todos los archivos, incluyendo algunos que están ocultos para el usuario , son aquellos que comienzan por un punto.



A screenshot of a Linux desktop environment. At the top, there's a dark header bar with the word "Actividades" and a "Terminal" icon. On the right side of the header, it shows the date "3 de abr 08:18". Below the header is a terminal window with a dark background. In the terminal, the command "ls -a" is run, and it lists numerous files and directories starting with a dot, such as ". . . bin boot cdrom dev etc home lib lib32 lib64 libx32 lost+found media mnt opt proc root run sbin snap srv swapfile sys tmp usr var". The terminal window also has a small icon of a blue bird in the bottom-left corner.

```
Actividades Terminal 3 de abr 08:18
ds-2024@ds2024-VirtualBox:~$ ls -a
. . . bin boot cdrom dev etc home lib lib32 lib64 libx32 lost+found media mnt opt proc root run sbin snap srv swapfile sys tmp usr var
ds-2024@ds2024-VirtualBox:~$
```

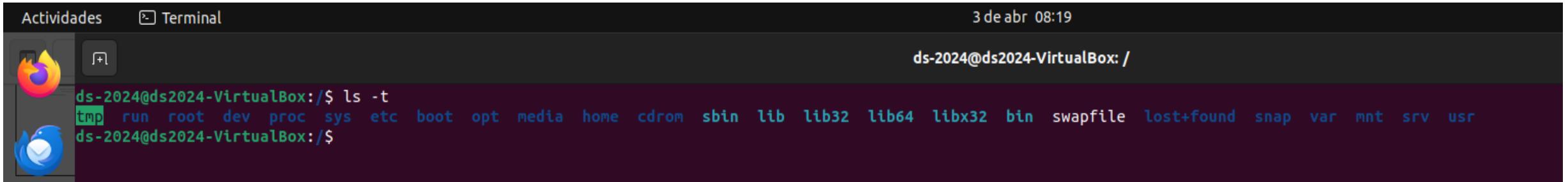
**ls -c** muestra el contenido del directorio ordenado por día y hora de creación.



A screenshot of a Linux desktop environment, similar to the one above. It shows a terminal window with the command "ls -c" run, which lists the same set of files and directories as the previous screenshot, but they are ordered by creation time. The terminal window has a small icon of a blue bird in the bottom-left corner.

```
Actividades Terminal 3 de abr 08:19
ds-2024@ds2024-VirtualBox:~$ ls -c
tmp run root dev proc sys etc boot opt media home cdrom mnt snap srv usr var sbin lib lib32 lib64 libx32 bin swapfile lost+found
ds-2024@ds2024-VirtualBox:~$
```

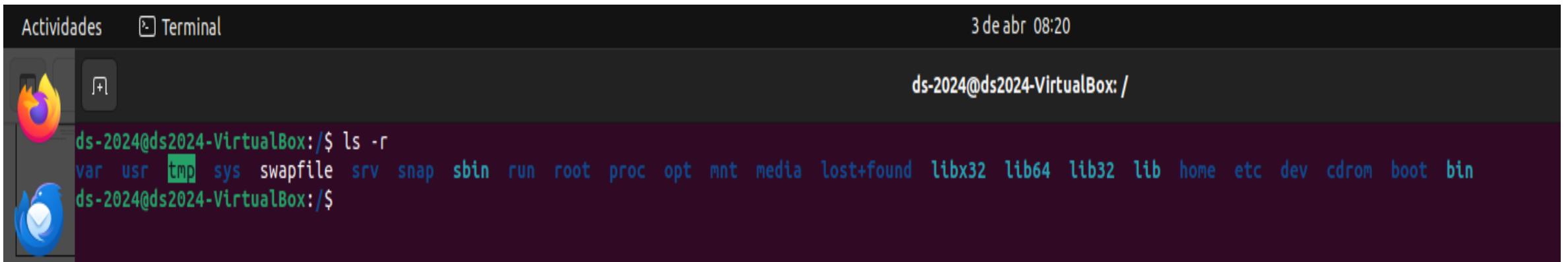
**ls -t** muestra el contenido del directorio ordenado por día y hora de modificación.



A screenshot of a Linux desktop environment. At the top, there's a dark header bar with the text "Actividades" and "Terminal". On the right side of the header, it shows the date and time: "3 de abr 08:19". Below the header is a terminal window with a dark background. The terminal window has a title bar with the text "ds-2024@ds2024-VirtualBox: /". Inside the terminal, the command "ls -t" is run, displaying a list of directory contents sorted by modification time. The output includes: tmp, run, root, dev, proc, sys, etc, boot, opt, media, home, cdrom, sbin, lib, lib32, lib64, libx32, bin, swapfile, lost+found, snap, var, mnt, srv, usr. The terminal prompt "ds-2024@ds2024-VirtualBox: /\$" is visible at the bottom.

```
ds-2024@ds2024-VirtualBox:/$ ls -t
tmp run root dev proc sys etc boot opt media home cdrom sbin lib lib32 lib64 libx32 bin swapfile lost+found snap var mnt srv usr
ds-2024@ds2024-VirtualBox:/$
```

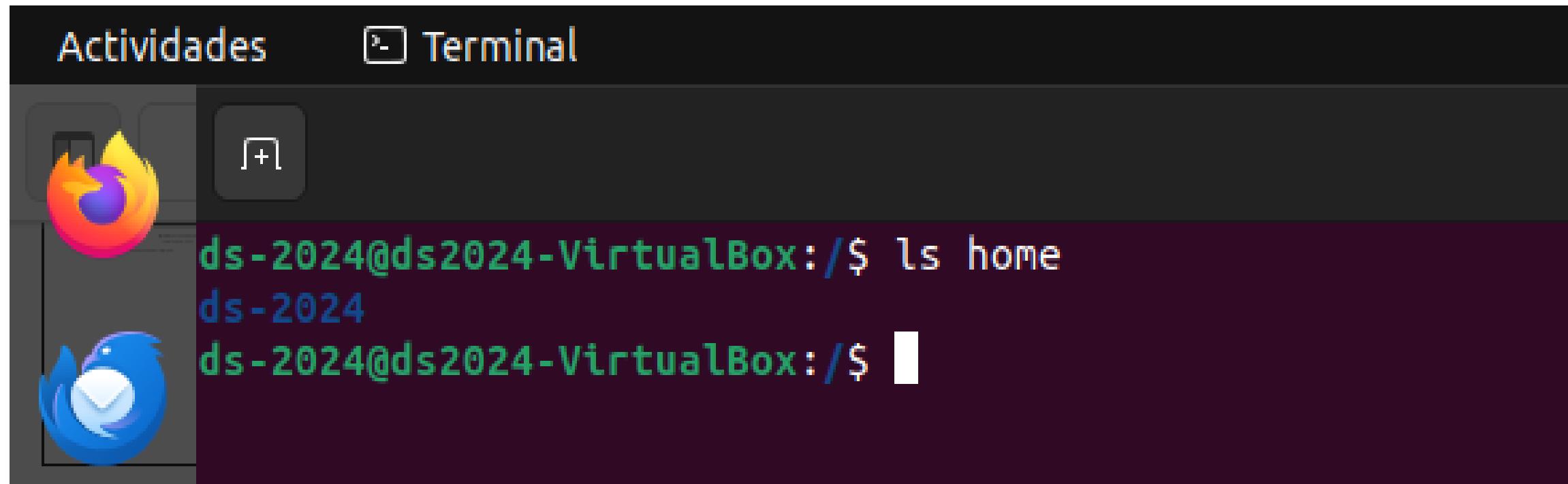
**ls -r** muestra el contenido del directorio y lo ordena alfabéticamente en orden inverso.



A screenshot of a Linux desktop environment, similar to the one above. It features a dark header bar with "Actividades" and "Terminal" and a date/time indicator "3 de abr 08:20". A terminal window is open with the title "ds-2024@ds2024-VirtualBox: /". The command "ls -r" is executed, listing the directory contents in reverse alphabetical order. The output shows the same set of files and directories as the previous screenshot, but in a different order. The terminal prompt "ds-2024@ds2024-VirtualBox: /\$" is at the bottom.

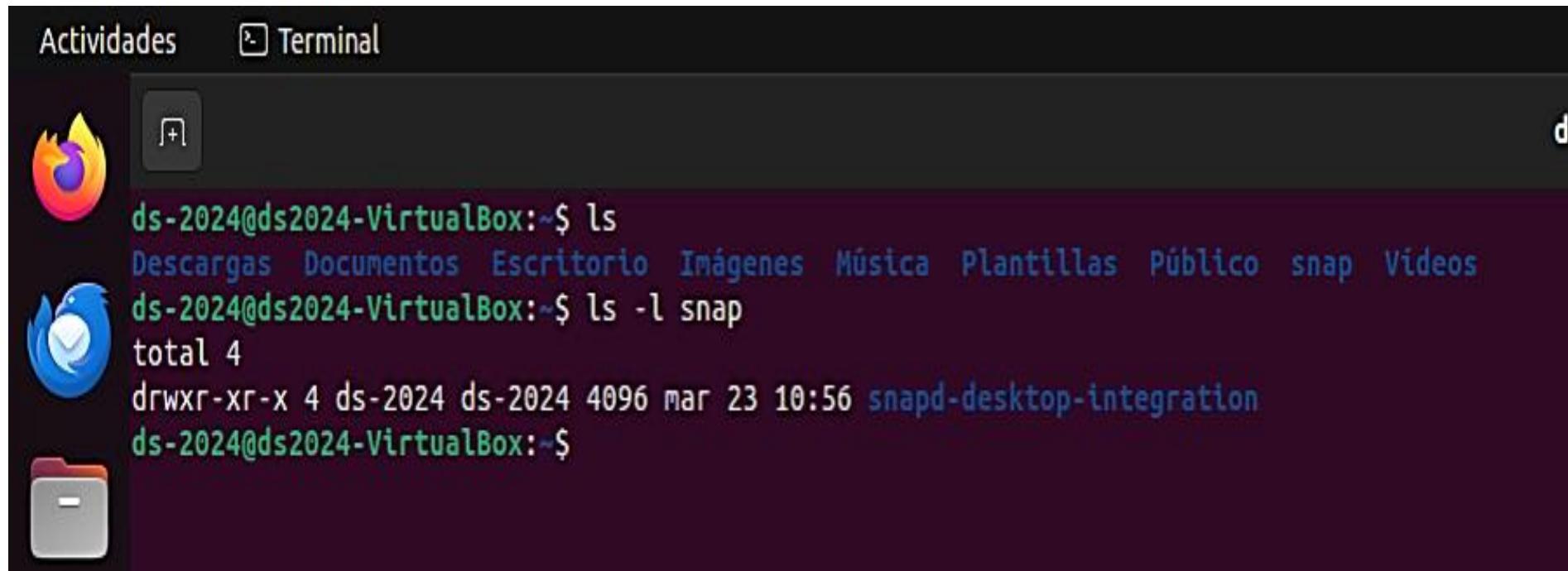
```
ds-2024@ds2024-VirtualBox:/$ ls -r
var usr tmp sys swapfile srv snap sbin run root proc opt mnt media lost+found libx32 lib64 lib32 lib home etc dev cdrom boot bin
ds-2024@ds2024-VirtualBox:/$
```

`ls subdir` muestra el contenido del subdirectorio “subdir”.

A screenshot of a dark-themed Ubuntu desktop environment. At the top, there is a dock with icons for the Dash, Home, and Terminal. The Terminal window is open and shows the command `ls home` being run. The output of the command, `ds-2024`, is displayed in the terminal window. The desktop background is a solid dark color.

```
ds-2024@ds2024-VirtualBox:/$ ls home
ds-2024
ds-2024@ds2024-VirtualBox:/$
```

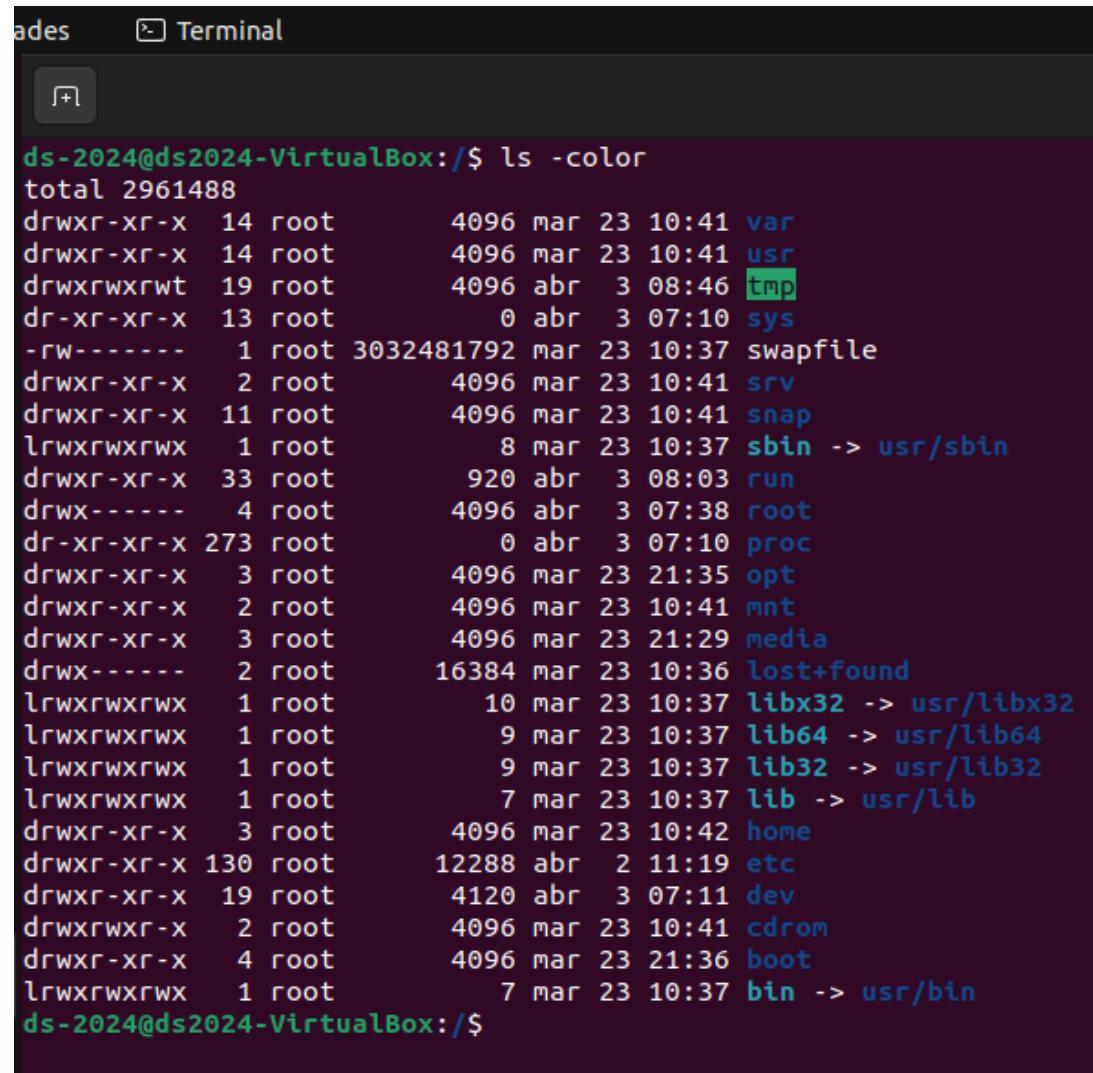
**ls -l filename** muestra toda la información sobre el archivo “filename”, también puede usarse para subdirectorios.



The screenshot shows a dark-themed desktop environment with a dock at the bottom containing icons for the Dash, Home, and a few other applications. A terminal window is open in the center, displaying the following command-line session:

```
Actividades Terminal
ds-2024@ds2024-VirtualBox:~$ ls
Descargas Documentos Escritorio Imágenes Música Plantillas Público snap Videos
ds-2024@ds2024-VirtualBox:~$ ls -l snap
total 4
drwxr-xr-x 4 ds-2024 ds-2024 4096 mar 23 10:56 snapd-desktop-integration
ds-2024@ds2024-VirtualBox:~$
```

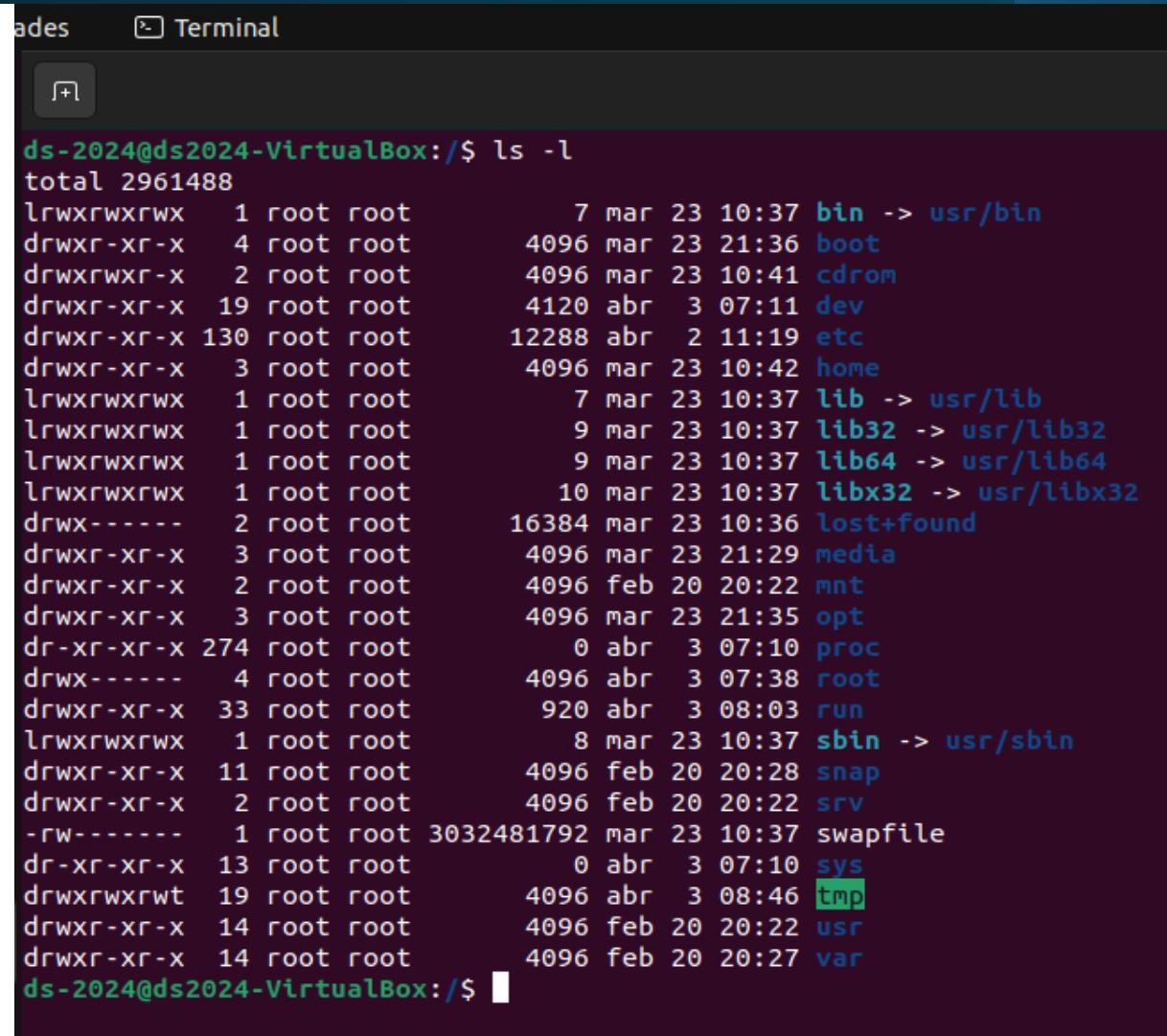
**ls --color** muestra el contenido del directorio coloreado: verde para los ejecutables, azul para las carpetas, fucsia para las imágenes, rojo para los comprimidos, etc.



The screenshot shows a terminal window titled "Terminal" with the command "ls -color" run by the user "ds-2024" on a system named "ds2024-VirtualBox". The output lists numerous system directories and files, each colored according to its type: green for executables, blue for directories, magenta for images, and red for compressed files. The output includes standard directories like /var, /usr, /tmp, /sys, and various system files such as swapfile, snap, and lib\*. The terminal interface includes a title bar, a menu bar with "File", "Edit", "View", "Search", "Help", and "Terminal", and a status bar at the bottom.

```
ades Terminal
ds-2024@ds2024-VirtualBox:~$ ls -color
total 2961488
drwxr-xr-x 14 root      4096 mar 23 10:41 var
drwxr-xr-x 14 root      4096 mar 23 10:41 usr
drwxrwxrwt 19 root      4096 abr  3 08:46 tmp
dr-xr-xr-x 13 root      0 abr  3 07:10 sys
-rw-----  1 root 3032481792 mar 23 10:37 swapfile
drwxr-xr-x  2 root      4096 mar 23 10:41 srv
drwxr-xr-x 11 root      4096 mar 23 10:41 snap
lrwxrwxrwx  1 root      8 mar 23 10:37 sbin -> usr/sbin
drwxr-xr-x 33 root      920 abr  3 08:03 run
drwx----- 4 root      4096 abr  3 07:38 root
dr-xr-xr-x 273 root      0 abr  3 07:10 proc
drwxr-xr-x  3 root      4096 mar 23 21:35 opt
drwxr-xr-x  2 root      4096 mar 23 10:41 mnt
drwxr-xr-x  3 root      4096 mar 23 21:29 media
drwx----- 2 root     16384 mar 23 10:36 lost+found
lrwxrwxrwx  1 root      10 mar 23 10:37 libx32 -> usr/libx32
lrwxrwxrwx  1 root      9 mar 23 10:37 lib64 -> usr/lib64
lrwxrwxrwx  1 root      9 mar 23 10:37 lib32 -> usr/lib32
lrwxrwxrwx  1 root      7 mar 23 10:37 lib -> usr/lib
drwxr-xr-x  3 root      4096 mar 23 10:42 home
drwxr-xr-x 130 root    12288 abr  2 11:19 etc
drwxr-xr-x 19 root     4120 abr  3 07:11 dev
drwxrwxr-x  2 root      4096 mar 23 10:41 cdrom
drwxr-xr-x  4 root      4096 mar 23 21:36 boot
lrwxrwxrwx  1 root      7 mar 23 10:37 bin -> usr/bin
ds-2024@ds2024-VirtualBox:~$
```

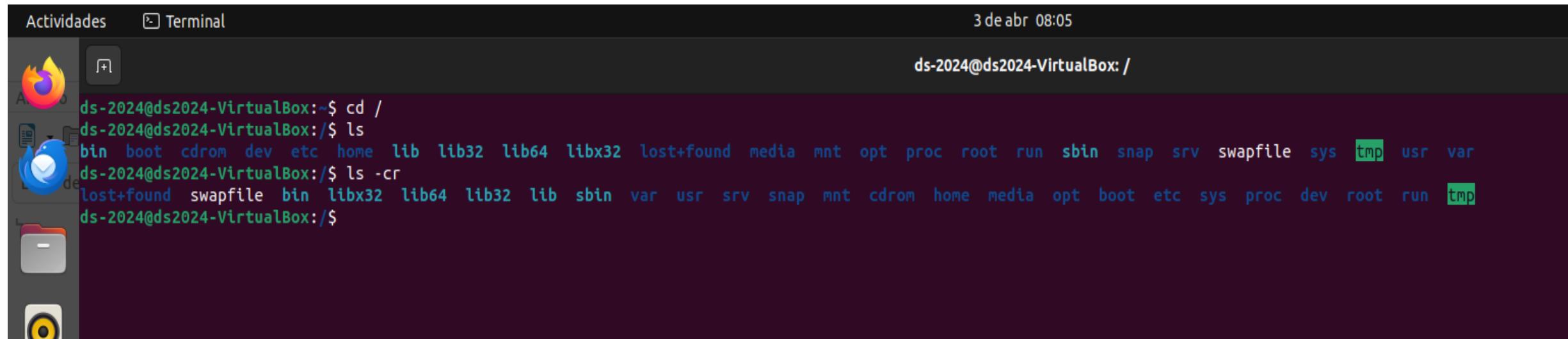
**ls -l** muestra toda la información de cada archivo, incluyendo permisos, tipo de archivo, tamaño y fecha de creación o del último cambio introducido, etc.



The screenshot shows a terminal window with the title "Terminal". The command "ls -l" is run, displaying a detailed listing of files and directories. The output includes columns for permissions, owner, group, size, date modified, and file name. Some entries are symbolic links pointing to other paths. The terminal prompt "ds-2024@ds2024-VirtualBox:" is visible at the bottom.

```
total 2961488
lrwxrwxrwx  1 root root      7 mar 23 10:37 bin -> usr/bin
drwxr-xr-x  4 root root    4096 mar 23 21:36 boot
drwxrwxr-x  2 root root    4096 mar 23 10:41 cdrom
drwxr-xr-x 19 root root   4120 abr  3 07:11 dev
drwxr-xr-x 130 root root  12288 abr  2 11:19 etc
drwxr-xr-x  3 root root   4096 mar 23 10:42 home
lrwxrwxrwx  1 root root      7 mar 23 10:37 lib -> usr/lib
lrwxrwxrwx  1 root root     9 mar 23 10:37 lib32 -> usr/lib32
lrwxrwxrwx  1 root root     9 mar 23 10:37 lib64 -> usr/lib64
lrwxrwxrwx  1 root root    10 mar 23 10:37 libx32 -> usr/libx32
drwx----- 2 root root  16384 mar 23 10:36 lost+found
drwxr-xr-x  3 root root   4096 mar 23 21:29 media
drwxr-xr-x  2 root root   4096 feb 20 20:22 mnt
drwxr-xr-x  3 root root   4096 mar 23 21:35 opt
dr-xr-xr-x 274 root root     0 abr  3 07:10 proc
drwx----- 4 root root   4096 abr  3 07:38 root
drwxr-xr-x 33 root root    920 abr  3 08:03 run
lrwxrwxrwx  1 root root      8 mar 23 10:37 sbin -> usr/sbin
drwxr-xr-x 11 root root   4096 feb 20 20:28 snap
drwxr-xr-x  2 root root   4096 feb 20 20:22 srv
-rw-------  1 root root 3032481792 mar 23 10:37 swapfile
dr-xr-xr-x 13 root root     0 abr  3 07:10 sys
drwxrwxrwt 19 root root   4096 abr  3 08:46 tmp
drwxr-xr-x 14 root root   4096 feb 20 20:22 usr
drwxr-xr-x 14 root root   4096 feb 20 20:27 var
```

Las opciones anteriores pueden combinarse. Por ejemplo, `ls -cr` muestra el directorio ordenando inversamente por fechas



The screenshot shows a Linux desktop environment with a dark theme. A terminal window is open in the center, displaying a command-line session. The terminal title bar says "Terminal". The date and time "3 de abr 08:05" are shown above the terminal window. The terminal window contains the following text:

```
Actividades Terminal 3 de abr 08:05
ds-2024@ds2024-VirtualBox: ~
ds-2024@ds2024-VirtualBox: ~$ cd /
ds-2024@ds2024-VirtualBox: /$ ls
bin boot cdrom dev etc home lib lib32 lib64 libx32 lost+found media mnt opt proc root run sbin snap srv swapfile sys tmp usr var
ds-2024@ds2024-VirtualBox: /$ ls -cr
lost+found swapfile bin libx32 lib64 lib32 lib sbin var usr srv snap mnt cdrom home media opt boot etc sys proc dev root run tmp
ds-2024@ds2024-VirtualBox: /$
```

The terminal window has a dark background with light-colored text. The file names are color-coded: "tmp" and "var" are in green, while "lost+found" and "swapfile" are in red. The rest of the file names are in white. The desktop interface includes a dock on the left with icons for the terminal, a file manager, and other applications.

La shell de comandos admite el uso de caracteres denominados “comodín” (o wildcards), que permiten abarcar conjuntos de ficheros o directorios cuyo nombre encaje con la expresión que definen:

El carácter “\*” se utiliza para representar cualquier carácter un número indefinido de veces.

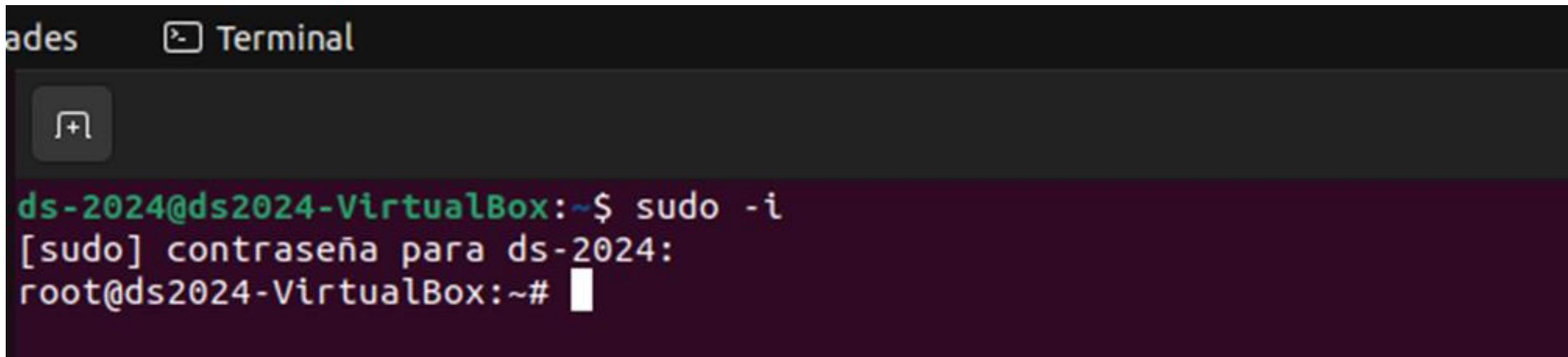
El carácter “?” se utiliza para representar cualquier carácter una única vez.

Si hay dos ficheros, “bar.txt” y “baz.txt”, se podrían listar ambos (sin considerar el resto de los ficheros del directorio) mediante el comando **ls ba?.txt**.

Asimismo, si se quisieran listar todos los ficheros con extensión .gif, podría hacerse mediante el comando **ls \*.gif**.

Existe otro comando, denominado **dir**, que tiene la misma función que **ls**, pero sin mostrar tanta información.

Para cambiar al modo superusuario: **sudo -i** .  
Luego introducir la contraseña de root

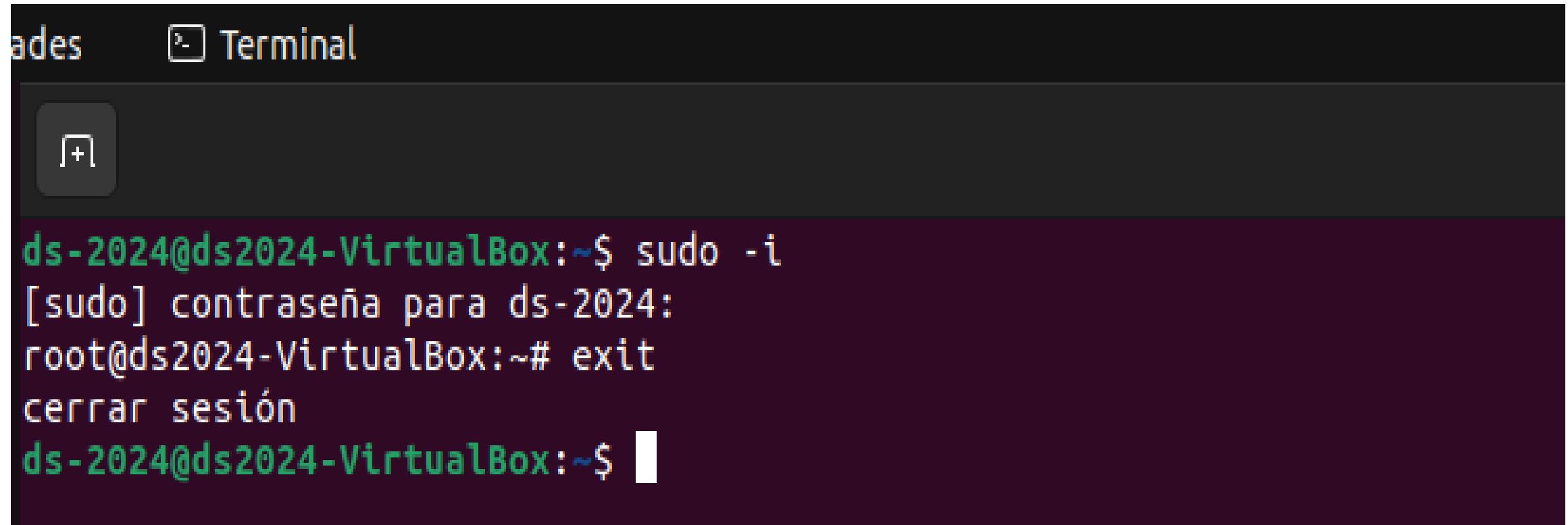


The screenshot shows a terminal window titled "Terminal". The window has a dark background and light-colored text. At the top, there is a menu bar with some icons and the word "Terminal". Below the menu bar, there is a toolbar with a single icon. The main area of the terminal contains the following text:

```
ades      □ Terminal
[+]
ds-2024@ds2024-VirtualBox:~$ sudo -i
[sudo] contraseña para ds-2024:
root@ds2024-VirtualBox:~# █
```

The terminal prompt shows the user has successfully become root.

para salir del modo superusuario escribimos: **exit**



The screenshot shows a terminal window titled "Terminal". The window has a dark theme with a light gray header bar. In the header bar, there is a small icon with a plus sign and the word "Terminal". The main area of the terminal shows the following command-line session:

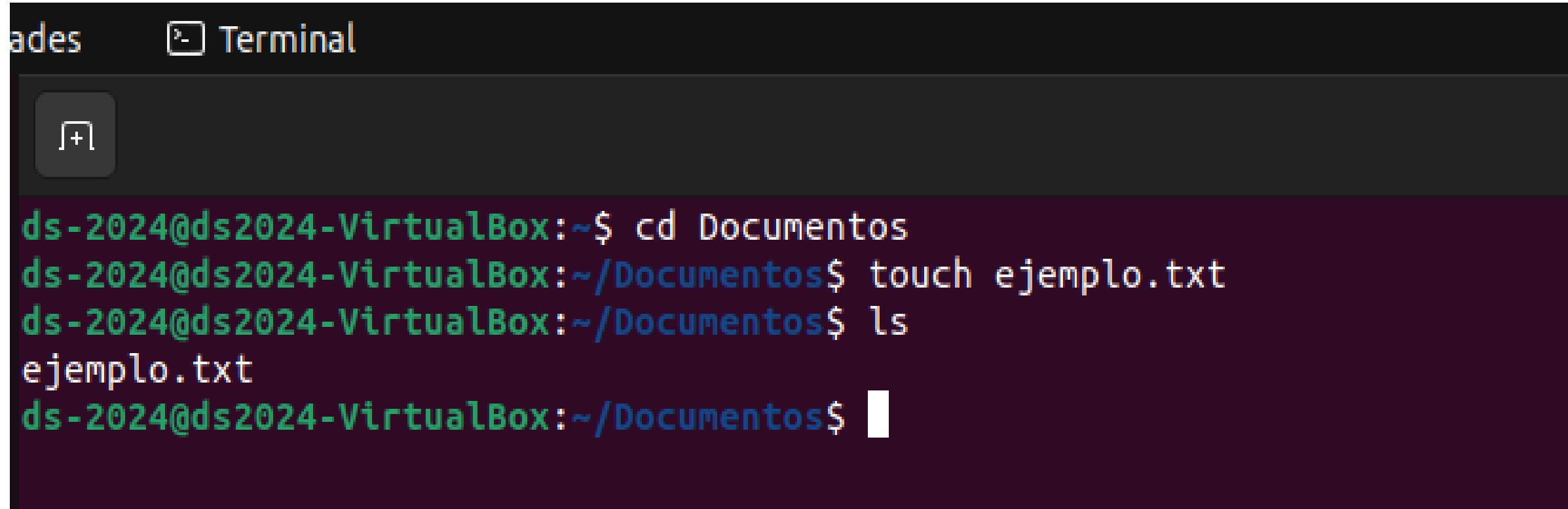
```
ades Terminal
[+]
ds-2024@ds2024-VirtualBox:~$ sudo -i
[sudo] contraseña para ds-2024:
root@ds2024-VirtualBox:~# exit
cerrar sesión
ds-2024@ds2024-VirtualBox:~$
```

para limpiar pantalla escribimos: **clear**

# Creación de ficheros

El comando **touch** actualiza los registros de fecha y hora con la fecha y hora actual de los ficheros indicados como argumento. Si el fichero no existe, el comando **touch** lo crea. Su uso más frecuente es para crear archivos.

La sintaxis del comando **touch** sigue la forma: **touch nombre\_fichero**.



The screenshot shows a terminal window with a dark background and light-colored text. The title bar says "Terminal". The window contains the following command-line session:

```
ds-2024@ds2024-VirtualBox:~$ cd Documentos
ds-2024@ds2024-VirtualBox:~/Documentos$ touch ejemplo.txt
ds-2024@ds2024-VirtualBox:~/Documentos$ ls
ejemplo.txt
ds-2024@ds2024-VirtualBox:~/Documentos$
```

Al escribir **touch ejemplo.txt** se crea el archivo “ejemplo.txt” en el directorio actual si este no existiera. Mas ejemplos:

ades  Terminal

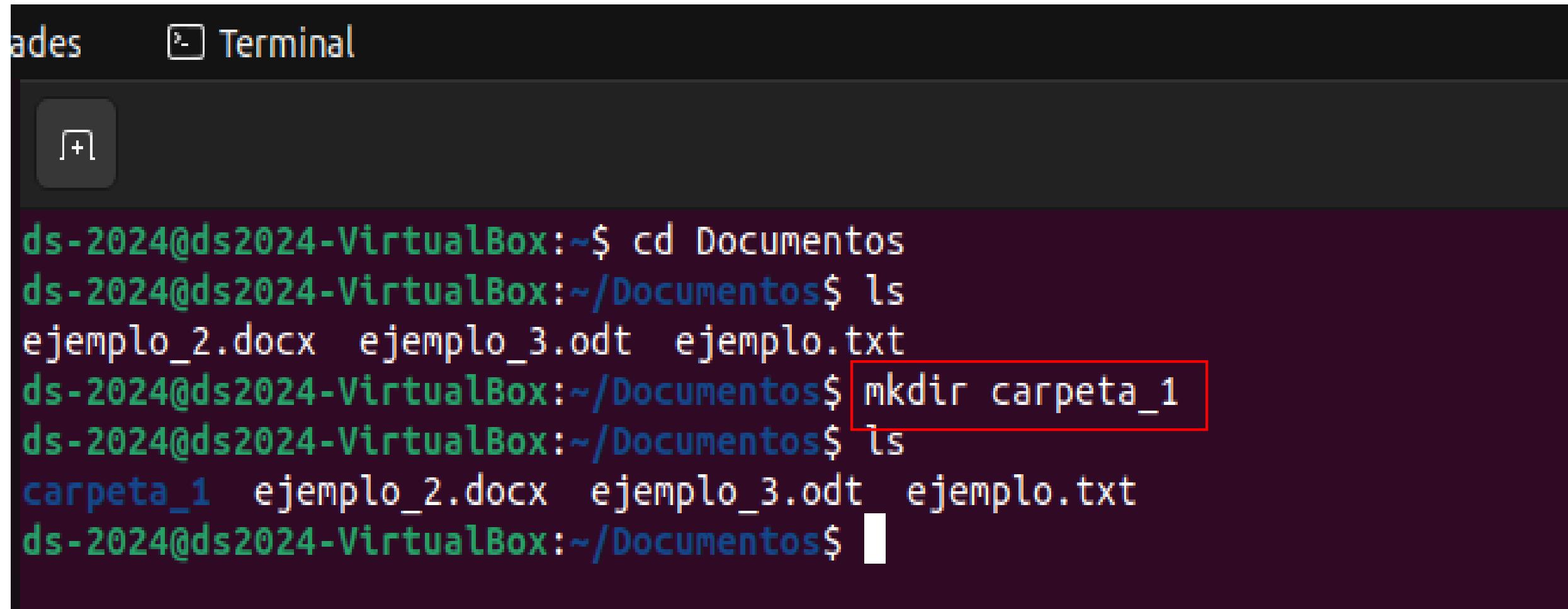
```
ds-2024@ds2024-VirtualBox:~$ cd Documentos
ds-2024@ds2024-VirtualBox:~/Documentos$ touch ejemplo.txt
ds-2024@ds2024-VirtualBox:~/Documentos$ ls
ejemplo.txt
ds-2024@ds2024-VirtualBox:~/Documentos$ touch ejemplo_2.docx
ds-2024@ds2024-VirtualBox:~/Documentos$ touch ejemplo_3.odt
ds-2024@ds2024-VirtualBox:~/Documentos$ ls
ejemplo_2.docx  ejemplo_3.odt  ejemplo.txt
```

## Creación de subdirectorios: **mkdir**

**mkdir:** make directory

El comando **mkdir** permite crear un nuevo subdirectorio (subcarpeta). La sintaxis es **mkdir subdir1** donde **subdir1** es el nombre del directorio (carpeta) que se va a crear

# Ejemplo:



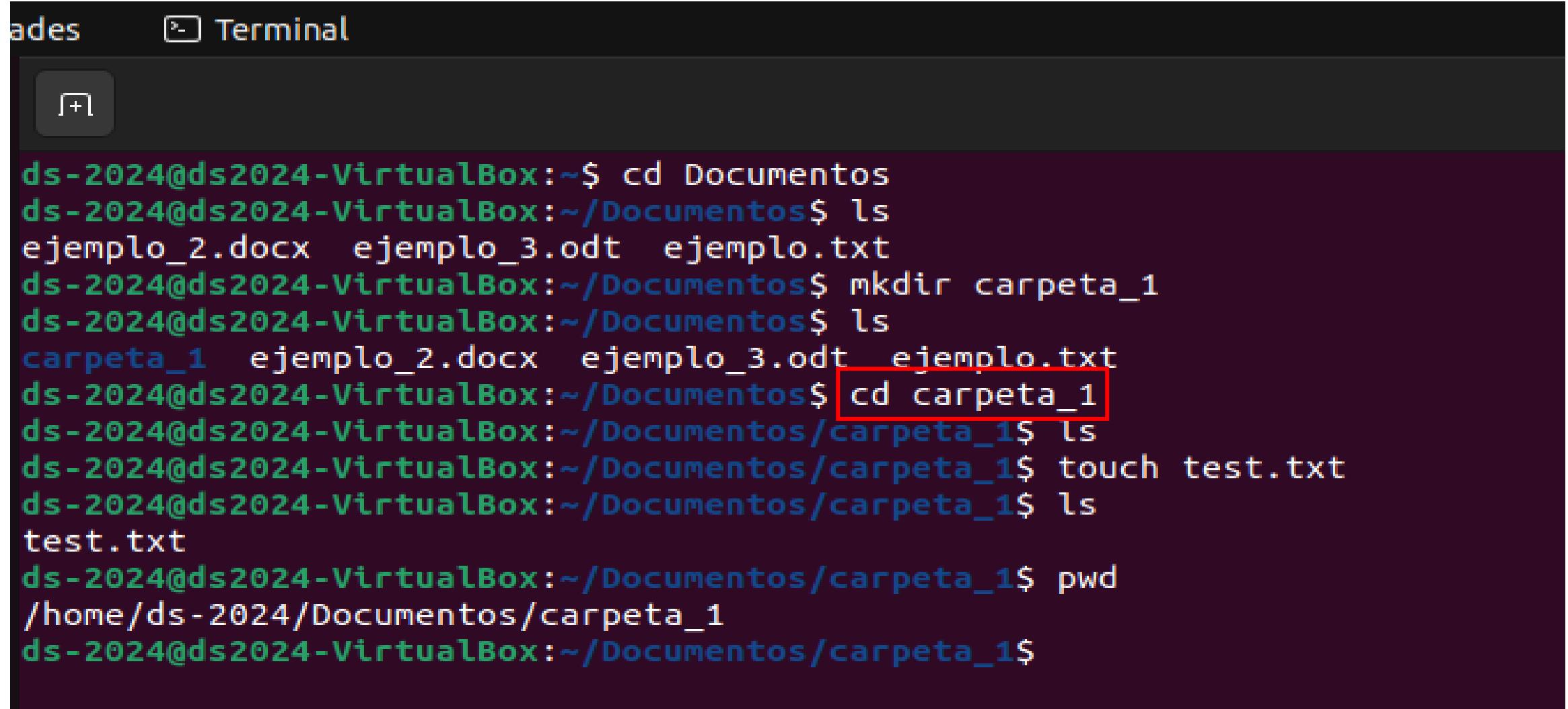
The screenshot shows a terminal window titled "Terminal" with a dark theme. The terminal window has a header bar with the title and a small icon. Below the header is a toolbar with a single button containing a plus sign. The main area of the terminal displays a command-line session:

```
ades      Terminal

ds-2024@ds2024-VirtualBox:~$ cd Documentos
ds-2024@ds2024-VirtualBox:~/Documentos$ ls
ejemplo_2.docx  ejemplo_3.odt  ejemplo.txt
ds-2024@ds2024-VirtualBox:~/Documentos$ mkdir carpeta_1
ds-2024@ds2024-VirtualBox:~/Documentos$ ls
carpeta_1  ejemplo_2.docx  ejemplo_3.odt  ejemplo.txt
ds-2024@ds2024-VirtualBox:~/Documentos$
```

The command `mkdir carpeta_1` is highlighted with a red rectangular box.

Entramos al subdirectorio y creamos un archivo llamado test.txt



The screenshot shows a terminal window titled "Terminal" with the following session history:

```
ades Terminal
[+]
ds-2024@ds2024-VirtualBox:~$ cd Documentos
ds-2024@ds2024-VirtualBox:~/Documentos$ ls
ejemplo_2.docx  ejemplo_3.odt  ejemplo.txt
ds-2024@ds2024-VirtualBox:~/Documentos$ mkdir carpeta_1
ds-2024@ds2024-VirtualBox:~/Documentos$ ls
carpeta_1  ejemplo_2.docx  ejemplo_3.odt  ejemplo.txt
ds-2024@ds2024-VirtualBox:~/Documentos$ cd carpeta_1
ds-2024@ds2024-VirtualBox:~/Documentos/carpeta_1$ ls
ds-2024@ds2024-VirtualBox:~/Documentos/carpeta_1$ touch test.txt
ds-2024@ds2024-VirtualBox:~/Documentos/carpeta_1$ ls
test.txt
ds-2024@ds2024-VirtualBox:~/Documentos/carpeta_1$ pwd
/home/ds-2024/Documentos/carpeta_1
ds-2024@ds2024-VirtualBox:~/Documentos/carpeta_1$
```

The command `cd carpeta_1` is highlighted with a red box.

# Borrado de subdirectorios: **rmdir**

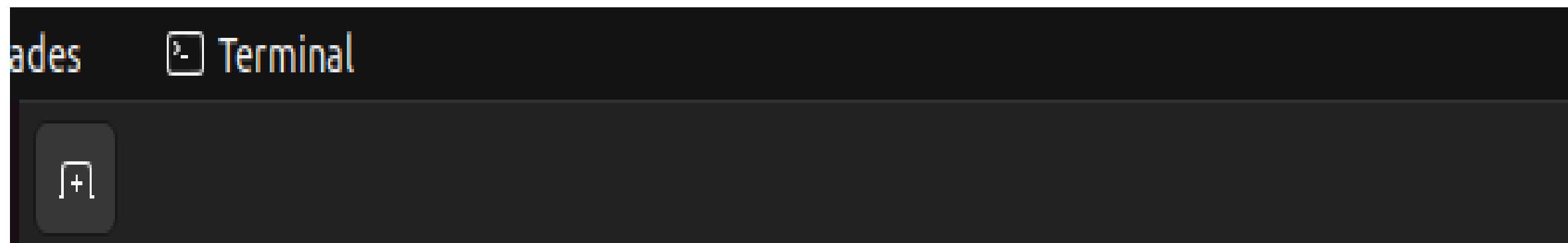
El comando **rmdir** borra uno o más directorios del sistema siempre que estos subdirectorios estén vacíos. La sintaxis es **rmdir subdir1** donde **subdir1** es el nombre del directorio que se va a eliminar.

# Ejemplo:

```
ds-2024@ds2024-VirtualBox:~/Documentos$ rmdir carpeta_1
rmdir: fallo al borrar 'carpeta_1': El directorio no está vacío
ds-2024@ds2024-VirtualBox:~/Documentos$ █
```

Para borrar subdirectorios (carpetas) que contienen archivos se escribe: **rm -r nombre\_directorio**

**-r** significa: remove



```
ades Terminal
+ ds-2024@ds2024-VirtualBox:~/Documentos$ rm -r carpeta_1
ds-2024@ds2024-VirtualBox:~/Documentos$ ls
carpeta_2  ejemplo_2.docx  ejemplo_3.odt  ejemplo.txt
ds-2024@ds2024-VirtualBox:~/Documentos$
```

A screenshot of a macOS desktop environment. In the top-left corner, the Dock shows the 'Finder' icon (a blue folder) and the 'Terminal' icon (a white terminal window with a blue border). The main window is a terminal session with a dark background and light-colored text. The user has run the command 'rm -r carpeta\_1' to delete a directory named 'carpeta\_1'. After the deletion, they run 'ls' to list the contents of the current directory, which includes 'carpeta\_2', 'ejemplo\_2.docx', 'ejemplo\_3.odt', and 'ejemplo.txt'. The terminal prompt ends with a '\$' sign.

# Copia de archivos: cp

## cp: copy

Sintaxis: cp [opciones] origen destino

**[opciones]**: Son opciones adicionales que modifican el comportamiento del comando cp. Algunas opciones comunes son:

-v: Muestra información detallada sobre el proceso de copia.

-r: Copia directorios de forma recursiva (incluidos todos los archivos y subdirectorios).

-n: No sobrescribe archivos existentes.

-b: Crea una copia de seguridad del archivo original.

**origen**: Es la ubicación del archivo o directorio que deseas copiar. Puede ser un nombre de archivo, una ruta completa o un comodín.

**destino**: Es la ubicación donde deseas copiar el archivo o directorio. Puede ser un nombre de archivo, una ruta completa o un directorio existente.

# Ejemplo

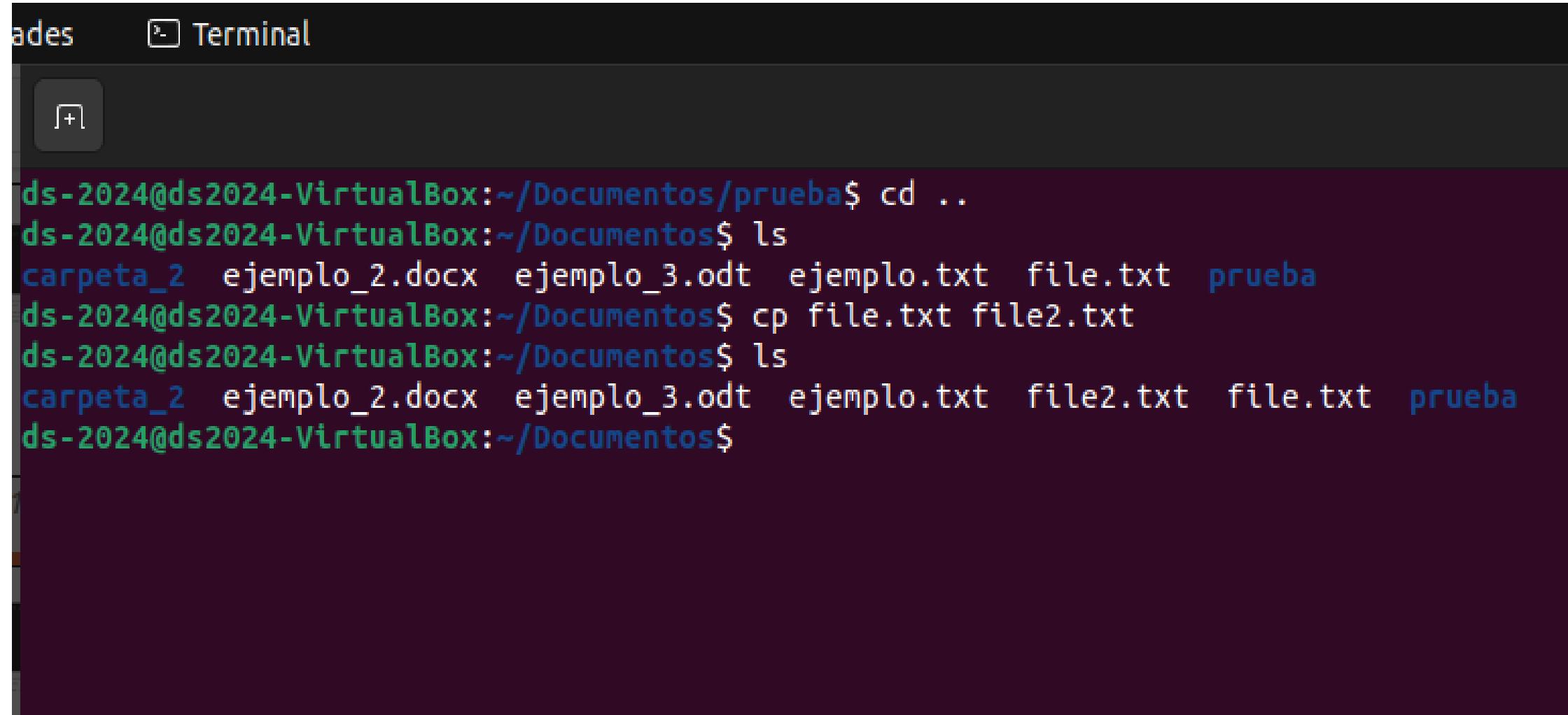
```
andes Terminal
+
ds-2024@ds2024-VirtualBox:~/Documentos$ mkdir prueba
ds-2024@ds2024-VirtualBox:~/Documentos$ ls
carpeta_2 ejemplo_2.docx ejemplo_3.odt ejemplo.txt prueba
ds-2024@ds2024-VirtualBox:~/Documentos$ touch file.txt
ds-2024@ds2024-VirtualBox:~/Documentos$ ls
carpeta_2 ejemplo_2.docx ejemplo_3.odt ejemplo.txt file.txt prueba
ds-2024@ds2024-VirtualBox:~/Documentos$ cp file.txt prueba
ds-2024@ds2024-VirtualBox:~/Documentos$ ls
carpeta_2 ejemplo_2.docx ejemplo_3.odt ejemplo.txt file.txt prueba
ds-2024@ds2024-VirtualBox:~/Documentos$ cd prueba
ds-2024@ds2024-VirtualBox:~/Documentos/prueba$ ls
file.txt
ds-2024@ds2024-VirtualBox:~/Documentos/prueba$
```

# Copiar un archivo y cambiarle el nombre:

Copia el archivo original y lo guarda con un nuevo nombre.

Sintaxis: **cp archivo.algo nuevo\_archivo.algo**

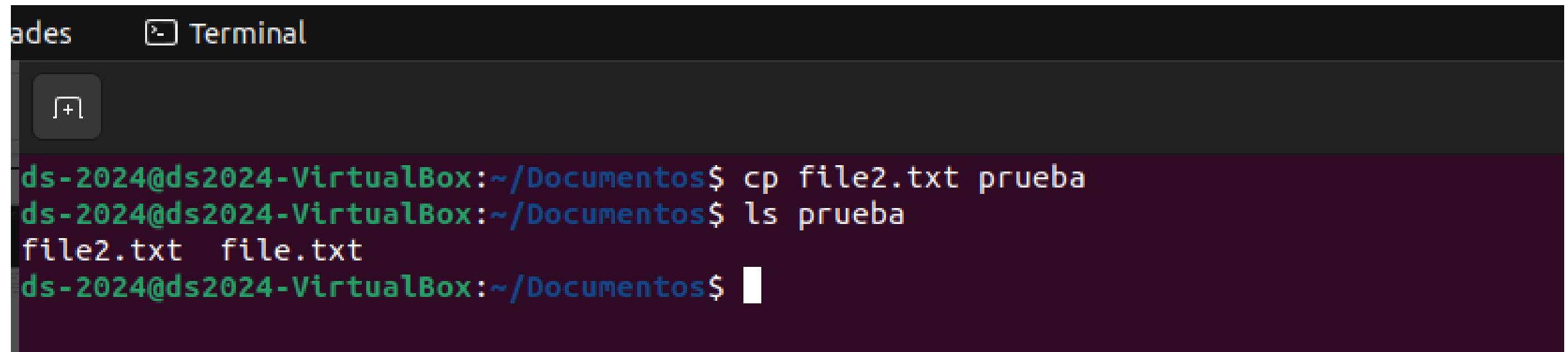
# Copiar un archivo y cambiarle el nombre:



The screenshot shows a terminal window titled "Terminal" with a dark theme. The window contains the following command-line session:

```
ades Terminal
[+]
ds-2024@ds2024-VirtualBox:~/Documentos/prueba$ cd ..
ds-2024@ds2024-VirtualBox:~/Documentos$ ls
carpeta_2 ejemplo_2.docx ejemplo_3.odt ejemplo.txt file.txt prueba
ds-2024@ds2024-VirtualBox:~/Documentos$ cp file.txt file2.txt
ds-2024@ds2024-VirtualBox:~/Documentos$ ls
carpeta_2 ejemplo_2.docx ejemplo_3.odt ejemplo.txt file2.txt file.txt prueba
ds-2024@ds2024-VirtualBox:~/Documentos$
```

# Otro ejemplo especificando ruta:



The screenshot shows a terminal window titled "Terminal" with a dark theme. The window has a header bar with tabs for "ades" and "Terminal". Below the header is a toolbar with a "+" icon. The terminal window displays the following command-line session:

```
ds-2024@ds2024-VirtualBox:~/Documentos$ cp file2.txt prueba
ds-2024@ds2024-VirtualBox:~/Documentos$ ls prueba
file2.txt  file.txt
ds-2024@ds2024-VirtualBox:~/Documentos$
```

# ¿Qué otras cosas se hacen con el comando cp? Escribir en el terminal: cp --help

```
andes Terminal ds-2024@ds2024-VirtualBox:~/Documentos$ cp --help
Modo de empleo: cp [OPCIÓN]... ORIGEN DESTINO
    o bien: cp [OPCIÓN]... ORIGEN... DIRECTORIO
    o bien: cp [OPCIÓN]... -t DIRECTORIO ORIGEN...
Copia ORIGEN a DESTINO, o varios ORIGEN(es) a DIRECTORIO.

Los argumentos obligatorios para las opciones largas son también obligatorios
para las opciones cortas.
-a, --archive           lo mismo que -dR --preserve=all
--attributes-only      no copia los datos del fichero, solamente los
                       atributos
--backup[=CONTROL]     crea una copia de seguridad de cada fichero de
                       destino que existe
-b                      como --backup pero no acepta ningún argumento
--copy-contents        copia el contenido de los ficheros especiales
                       cuando opera recursivamente
-d                      lo mismo que --no-dereference --preserve=link
-f, --force              si un fichero de destino no se puede abrir, lo
                         borra y lo intenta de nuevo (no se tiene en
                         cuenta si se utiliza también la opción -n)
-i, --interactive       pide confirmación antes de sobreescibir
-H                      sigue los enlaces simbólicos de la línea
                       de órdenes
-l, --link               crea enlaces duros de los ficheros en vez de copiarlos
-L, --dereference        siempre sigue los enlaces simbólicos en ORIGEN
-n, --no-clobber         no sobreescribe un fichero que existe
                         (tiene prioridad sobre una opción -i anterior)
-P, --no-dereference   nunca sigue los enlaces simbólicos en ORIGEN
-P, --preserve[=LISTA_ATTR] conserva si puede los atributos especificados,
                         (por omisión: mode,ownership,timestamps)
                         atributos adicionales: context, links, xattr,
                         all
--no-preserve=LISTA_ATTR no conserva los atributos especificados
--parents añade el directorio de origen a DIRECTORIO
-R, -r, --recursive     copia directorios recursivamente
--reflink[=CUÁNDO]       controla copias clonadas/Cow. Ver más abajo.
--remove-destination   borra cada fichero de destino que exista antes
                         de intentar abrirlo (compárese con --force).
--sparse=CUÁNDO         controla la creación de ficheros dispersos.
                         Véase más abajo.
--strip-trailing-slashes elimina todas las barras finales de cada
                         argumento ORIGEN
-s, --symbolic-link     crea enlaces simbólicos en lugar de copiarlos
-S, --suffix=SUFijo     reemplaza el sufijo de respaldo habitual
-t, --target-directory=DIRECTORIO copia todos los argumentos ORIGEN al
                         directorio DIRECTORIO
-T, --no-target-directory considera DEST como un archivo normal
-u, --update             copia solamente cuando el fichero ORIGEN es
                         más moderno que el fichero de destino,
                         o cuando falta el fichero de destino
-v, --verbose            da detalles sobre lo que se va haciendo
-x, --one-file-system   permanece en este sistema de ficheros
-Z, --context[=CTX]      establece el contexto de seguridad SELinux del fichero de
                         destino al tipo predeterminado
                           cosa_Z si se especifica CTX entonces establece
```

El parámetro **--help** se puede utilizar para todos los comandos:

Escribir: **ls --help**

# Mover Archivos: mv origen destino

Mueve los archivos de un *directorio a otro*, o *del directorio actual a otro directorio*.

# Ejemplo 1:

ades 2 Terminal



```
ds-2024@ds2024-VirtualBox:~/Documentos$ ls
carpeta_2 ejemplo_2.docx ejemplo_3.odt ejemplo.txt file2.txt file.txt prueba
ds-2024@ds2024-VirtualBox:~/Documentos$ mv ejemplo.txt prueba
ds-2024@ds2024-VirtualBox:~/Documentos$ ls
carpeta_2 ejemplo_2.docx ejemplo_3.odt file2.txt file.txt prueba
ds-2024@ds2024-VirtualBox:~/Documentos$ ls prueba
ejemplo.txt file2.txt file.txt
ds-2024@ds2024-VirtualBox:~/Documentos$ 
```

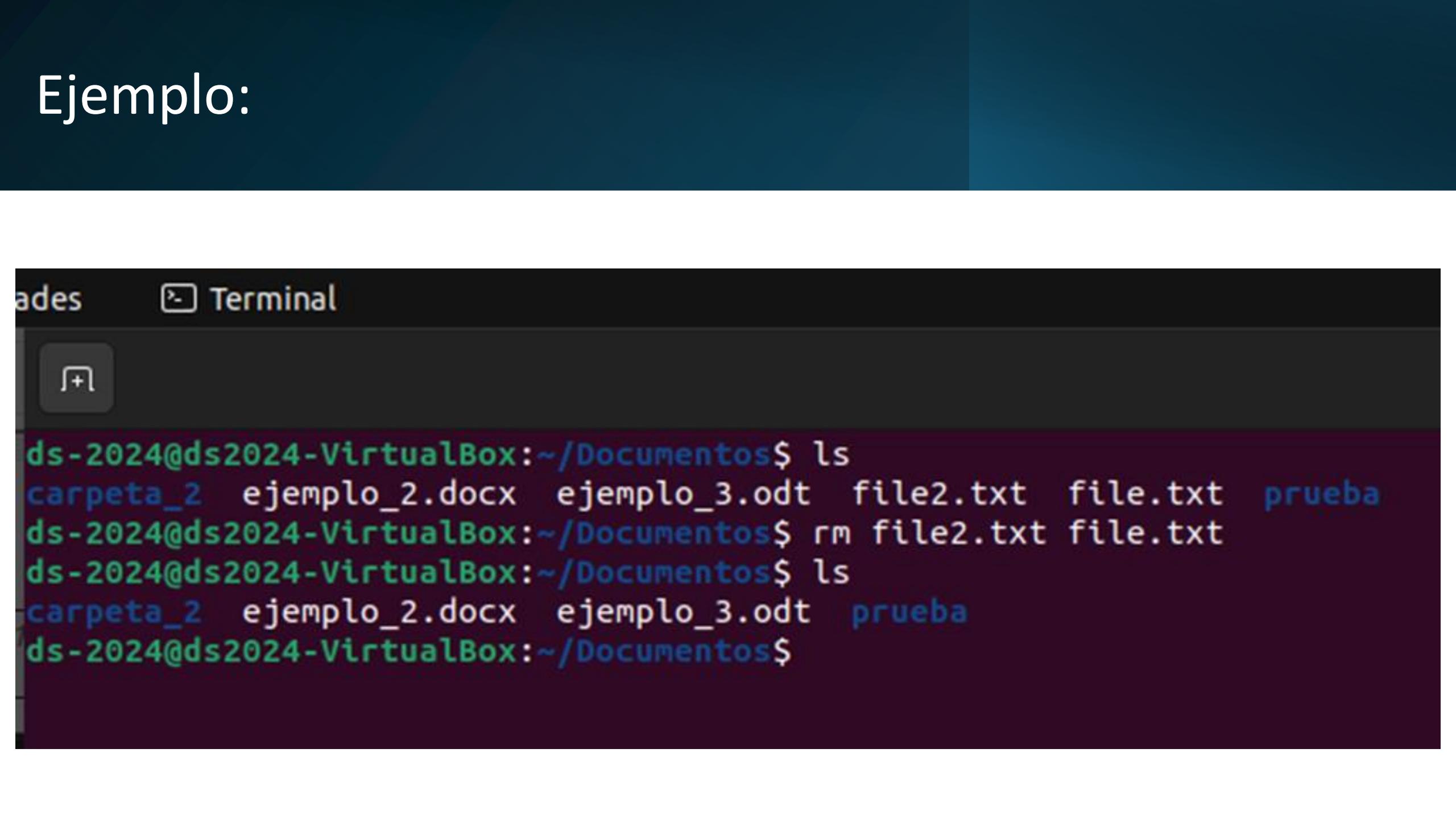
## Ejemplo 2:

```
ades Terminal 3 d
+ ds-2024@ds2024-VirtualBox:~/Documentos$ touch archivo1.txt
ds-2024@ds2024-VirtualBox:~/Documentos$ ls
archivo1.txt  carpeta_2  ejemplo_2.docx  ejemplo_3.odt  file2.txt  file.txt  prueba
ds-2024@ds2024-VirtualBox:~/Documentos$ mv archivo1.txt carpeta_2
ds-2024@ds2024-VirtualBox:~/Documentos$ ls carpeta_2
archivo1.txt
ds-2024@ds2024-VirtualBox:~/Documentos$ pwd
/home/ds-2024/Documentos
ds-2024@ds2024-VirtualBox:~/Documentos$ mv /home/ds-2024/Documentos/carpeta_2/archivo1.txt prueba
ds-2024@ds2024-VirtualBox:~/Documentos$ ls prueba
archivo1.txt  ejemplo.txt  file2.txt  file.txt
ds-2024@ds2024-VirtualBox:~/Documentos$
```

# Borrado de archivos: **rm**

El comando **rm** elimina uno o más archivos de un directorio en el cual se tenga permiso de escritura. La sintaxis es **rm file1 file2** . En el comando, se pueden utilizar los caracteres comodín \* y ?.

# Ejemplo:



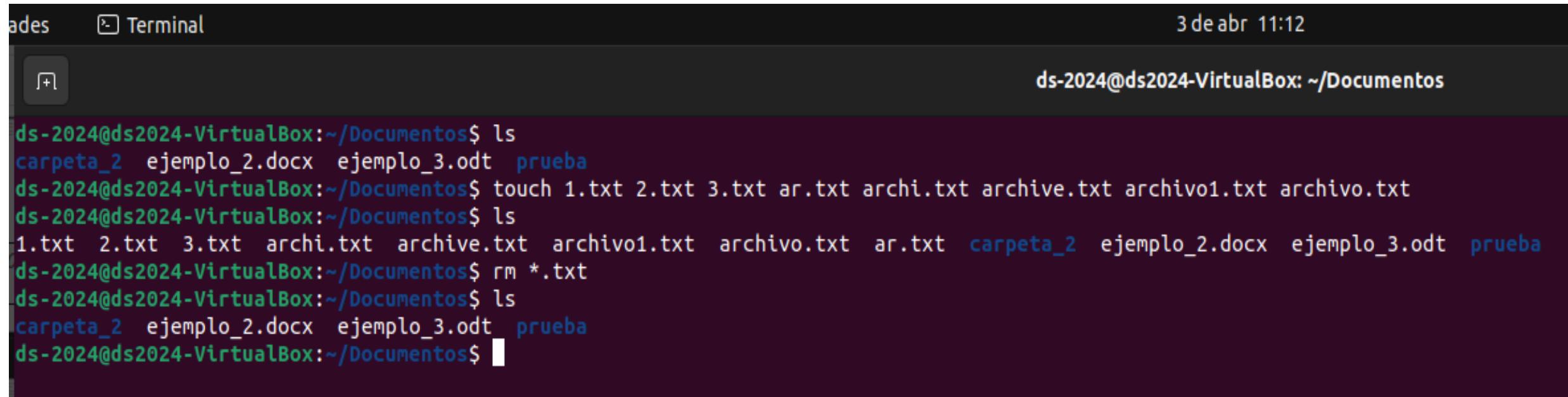
The screenshot shows a dark-themed macOS desktop environment. At the top, the Dock contains two icons: 'Alfred' (with a search field) and 'Terminal'. The main window is a terminal application with a dark background. It displays the following command-line session:

```
ades      [-] Terminal

+ 

ds-2024@ds2024-VirtualBox:~/Documentos$ ls
carpeta_2  ejemplo_2.docx  ejemplo_3.odt  file2.txt  file.txt  prueba
ds-2024@ds2024-VirtualBox:~/Documentos$ rm file2.txt file.txt
ds-2024@ds2024-VirtualBox:~/Documentos$ ls
carpeta_2  ejemplo_2.docx  ejemplo_3.odt  prueba
ds-2024@ds2024-VirtualBox:~/Documentos$
```

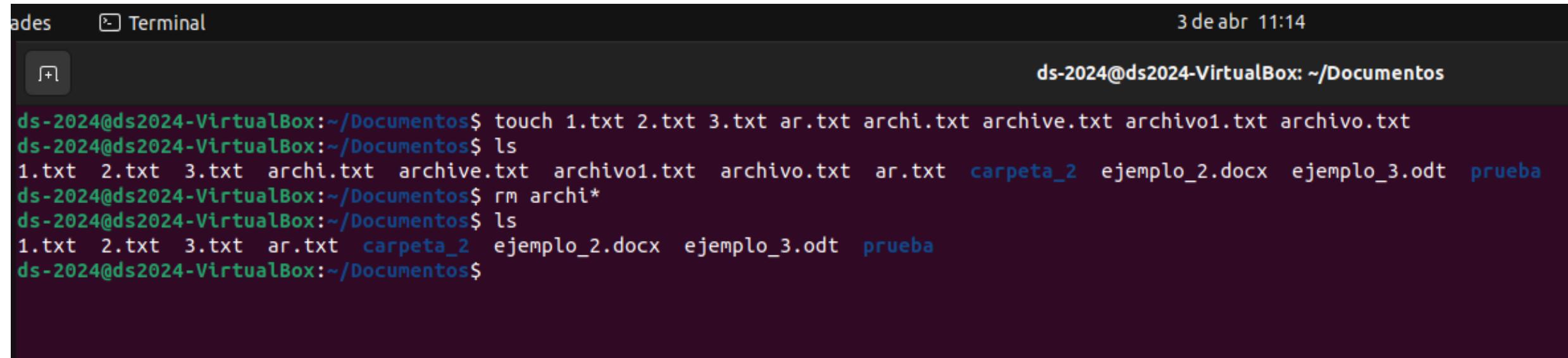
Para eliminar todos los archivos con una extensión específica: **rm \*.txt**



The screenshot shows a terminal window with the following session:

```
ades Terminal 3 de abr 11:12
ds-2024@ds2024-VirtualBox: ~/Documentos
ds-2024@ds2024-VirtualBox:~/Documentos$ ls
carpeta_2 ejemplo_2.docx ejemplo_3.odt prueba
ds-2024@ds2024-VirtualBox:~/Documentos$ touch 1.txt 2.txt 3.txt ar.txt archi.txt archive.txt archivo1.txt archivo.txt
ds-2024@ds2024-VirtualBox:~/Documentos$ ls
1.txt 2.txt 3.txt archi.txt archive.txt archivo1.txt archivo.txt ar.txt carpeta_2 ejemplo_2.docx ejemplo_3.odt prueba
ds-2024@ds2024-VirtualBox:~/Documentos$ rm *.txt
ds-2024@ds2024-VirtualBox:~/Documentos$ ls
carpeta_2 ejemplo_2.docx ejemplo_3.odt prueba
ds-2024@ds2024-VirtualBox:~/Documentos$
```

## Eliminar archivos que comienzan con un nombre específico: **rm nombre\***



The screenshot shows a terminal window with the following details:

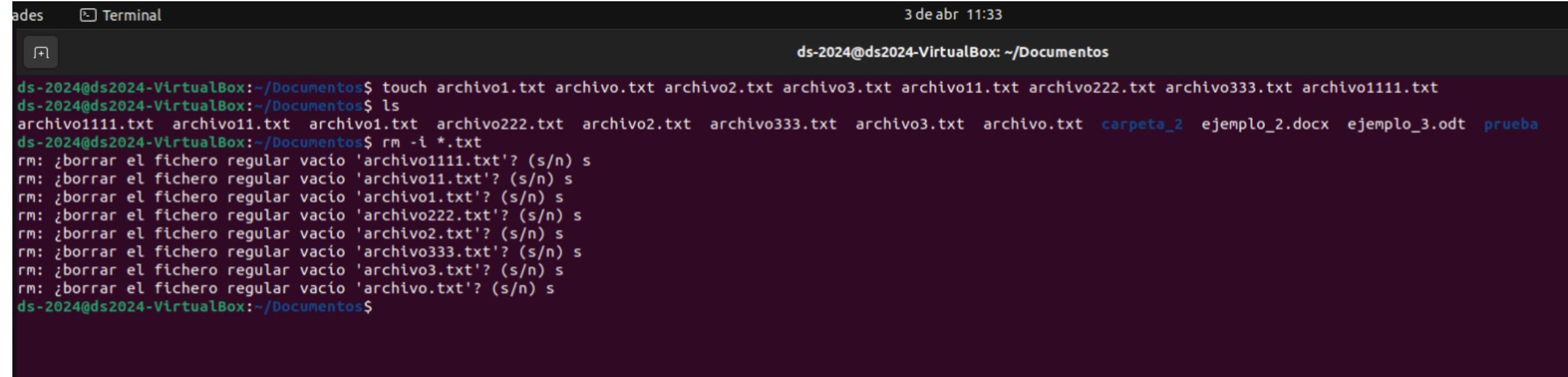
- Top bar: "ades" (partially visible), "Terminal", and the date/time "3 de abr 11:14".
- User information: "ds-2024@ds2024-VirtualBox: ~/Documentos".
- Terminal content:

```
ds-2024@ds2024-VirtualBox:~/Documentos$ touch 1.txt 2.txt 3.txt ar.txt archi.txt archive.txt archivo1.txt archivo.txt
ds-2024@ds2024-VirtualBox:~/Documentos$ ls
1.txt 2.txt 3.txt archi.txt archive.txt archivo1.txt archivo.txt ar.txt carpeta_2 ejemplo_2.docx ejemplo_3.odt prueba
ds-2024@ds2024-VirtualBox:~/Documentos$ rm archi*
ds-2024@ds2024-VirtualBox:~/Documentos$ ls
1.txt 2.txt 3.txt ar.txt carpeta_2 ejemplo_2.docx ejemplo_3.odt prueba
ds-2024@ds2024-VirtualBox:~/Documentos$
```

# Eliminar archivos que contienen una letra específica: rm archivo?.algo

```
ades Terminal 3 de abr 11:26
ds-2024@ds2024-VirtualBox:~/Documentos
ds-2024@ds2024-VirtualBox:~/Documentos$ touch archivo1.txt archivo.txt archivo2.txt archivo3.txt archivo11.txt archivo222.txt archivo333.txt archivo1111.txt
ds-2024@ds2024-VirtualBox:~/Documentos$ ls
archivo1111.txt archivo11.txt archivo1.txt archivo222.txt archivo2.txt archivo333.txt archivo3.txt archivo.txt carpeta_2 ejemplo_2.docx ejemplo_3.odt prueba
ds-2024@ds2024-VirtualBox:~/Documentos$ rm archivo?.txt
ds-2024@ds2024-VirtualBox:~/Documentos$ ls
archivo1111.txt archivo11.txt archivo222.txt archivo333.txt archivo.txt carpeta_2 ejemplo_2.docx ejemplo_3.odt prueba
ds-2024@ds2024-VirtualBox:~/Documentos$ rm archivo??.txt
ds-2024@ds2024-VirtualBox:~/Documentos$ ls
archivo1111.txt archivo222.txt archivo333.txt archivo.txt carpeta_2 ejemplo_2.docx ejemplo_3.odt prueba
ds-2024@ds2024-VirtualBox:~/Documentos$ rm archivo???.txt
ds-2024@ds2024-VirtualBox:~/Documentos$ ls
archivo1111.txt archivo.txt carpeta_2 ejemplo_2.docx ejemplo_3.odt prueba
ds-2024@ds2024-VirtualBox:~/Documentos$ rm archivo????.txt
ds-2024@ds2024-VirtualBox:~/Documentos$ ls
archivo.txt carpeta_2 ejemplo_2.docx ejemplo_3.odt prueba
ds-2024@ds2024-VirtualBox:~/Documentos$
```

Para solicitar confirmación antes de eliminar archivos se puede escribir: **rm -i \*.txt**



The screenshot shows a terminal window with the following details:

- Top bar: "ades" and "Terminal".
- Time: "3 de abr 11:33".
- User and location: "ds-2024@ds2024-VirtualBox: ~/Documentos".
- Command history:
  - "touch archivo1.txt archivo.txt archivo2.txt archivo3.txt archivo11.txt archivo222.txt archivo333.txt archivo1111.txt"
  - "ls"
  - "rm -i \*.txt"
  - Confirmation loop:
    - rm: ¿borrar el fichero regular vacío 'archivo1111.txt'? (s/n) s
    - rm: ¿borrar el fichero regular vacío 'archivo11.txt'? (s/n) s
    - rm: ¿borrar el fichero regular vacío 'archivo1.txt'? (s/n) s
    - rm: ¿borrar el fichero regular vacío 'archivo222.txt'? (s/n) s
    - rm: ¿borrar el fichero regular vacío 'archivo2.txt'? (s/n) s
    - rm: ¿borrar el fichero regular vacío 'archivo333.txt'? (s/n) s
    - rm: ¿borrar el fichero regular vacío 'archivo3.txt'? (s/n) s
    - rm: ¿borrar el fichero regular vacío 'archivo.txt'? (s/n) s
  - "ds-2024@ds2024-VirtualBox:~/Documentos\$"

# Eliminar todos los archivos en el directorio actual: rm \*



The screenshot shows a terminal window titled "Terminal" with the following session history:

```
ades Terminal 3 de abr 11:36
ds-2024@ds2024-VirtualBox: ~/Documentos/prueba1
[+]
ds-2024@ds2024-VirtualBox:~/Documentos$ mkdir prueba1
ds-2024@ds2024-VirtualBox:~/Documentos$ cd prueba1
ds-2024@ds2024-VirtualBox:~/Documentos/prueba1$ touch archivo1.txt archivo.txt archivo2.txt archivo3.txt archivo11.txt archivo222.txt archivo333.txt archivo1111.txt
ds-2024@ds2024-VirtualBox:~/Documentos/prueba1$ ls
archivo1111.txt archivo11.txt archivo1.txt archivo222.txt archivo2.txt archivo333.txt archivo3.txt archivo.txt
ds-2024@ds2024-VirtualBox:~/Documentos/prueba1$ rm *
ds-2024@ds2024-VirtualBox:~/Documentos/prueba1$ ls
ds-2024@ds2024-VirtualBox:~/Documentos/prueba1$
```

Para ‘ver’ el contenido de un archivo se puede usar el comando **cat**.  
La sintaxis es: **cat nombre\_archivo.algo**

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ cat contenido.txt
Una de las principales dificultades que surge en el ámbito del desarrollo de software es la incompatibilidad de las herramientas que se van a utilizar con respecto a los diferentes sistemas operativos. Así, en muchas ocasiones, el software no es multiplataforma, es decir, se ha creado para ejecutarse en un sistema operativo concreto. Esto genera problemas si el sistema operativo sobre el que se va a trabajar no coincide con el que precisa el software que se quiere utilizar, por lo que en estos casos sería necesario instalar el sistema operativo requerido por el softwareds-2024@ds2024-VirtualBox:~/Documentos/practica$
```

Para numerar las líneas podemos escribir: **cat -n contenido.txt**

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ cat -n contenido.txt
1 Una de las principales dificultades que surge en el ámbito del desarrollo de software es la incompatibilidad de las herramientas
2 se van a utilizar con respecto a los diferentes sistemas operativos. Así, en muchas ocasiones, el software no es multiplataforma
3 es decir, se ha creado para ejecutarse en un sistema operativo concreto. Esto genera problemas si el sistema operativo sobre el
4 que se va a trabajar no coincide con el que precisa el software que se quiere utilizar, por lo que en estos casos sería necesario
5 instalar el sistema operativo requerido por el softwareds-2024@ds2024-VirtualBox:~/Documentos/practica$
```

# Ahora, vamos a visualizar el archivo contenido.txt usando el comando: cat contenido.txt

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ cat contenido.txt
Una de las principales dificultades que surge en el ámbito del desarrollo de software es la incompatibilidad de las herramientas que se van a utilizar con respecto a los diferentes sistemas operativos. Así, en muchas ocasiones, el software no es multiplataforma, es decir, se ha creado para ejecutarse en un sistema operativo concreto. Esto genera problemas si el sistema operativo sobre el que se va a trabajar no coincide con el que precisa el software que se quiere utilizar, por lo que en estos casos sería necesario instalar el sistema operativo requerido por el softwareds-2024@ds2024-VirtualBox:~/Documentos/practica$
```

Para numerar las líneas podemos escribir: **cat -n contenido.txt**

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ cat -n contenido.txt
1 Una de las principales dificultades que surge en el ámbito del desarrollo de software es la incompatibilidad de las herramientas
2 se van a utilizar con respecto a los diferentes sistemas operativos. Así, en muchas ocasiones, el software no es multiplataforma
3 es decir, se ha creado para ejecutarse en un sistema operativo concreto. Esto genera problemas si el sistema operativo sobre el
4 que se va a trabajar no coincide con el que precisa el software que se quiere utilizar, por lo que en estos casos sería necesario
5 instalar el sistema operativo requerido por el softwareds-2024@ds2024-VirtualBox:~/Documentos/practica$
```

# Hay varias formas de utilizar el comando cat:

**cat > filen.txt** : crea un nuevo archivo.

**cat archivo1.txt archivo2.txt > archivo3.txt** :

fusiona el archivo1.txt con el archivo2.txt y almacena el resultado en el archivo3.txt.

**tac archivo.txt**: muestra el contenido en orden inverso.

cat > filen.txt: crea un nuevo archivo y se puede escribir sobre él.



ds-2024@ds2024-Virtual

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ cat > nuevo.txt
```

```
Esto es un nuevo archivo txt  
para probar el editor cat, una vez que has terminado  
de escribir, presiona CTRL + D para salir del editor
```

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$
```

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$
```

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$
```

# Vamos a visualizar el archivo nuevo.txt:



ds-2024@ds2024-Virtual

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ cat nuevo.txt
```

Esto es un nuevo archivo txt  
para probar el editor cat, una vez que has terminado  
de escribir, presiona CTRL + D para salir del editor

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$
```

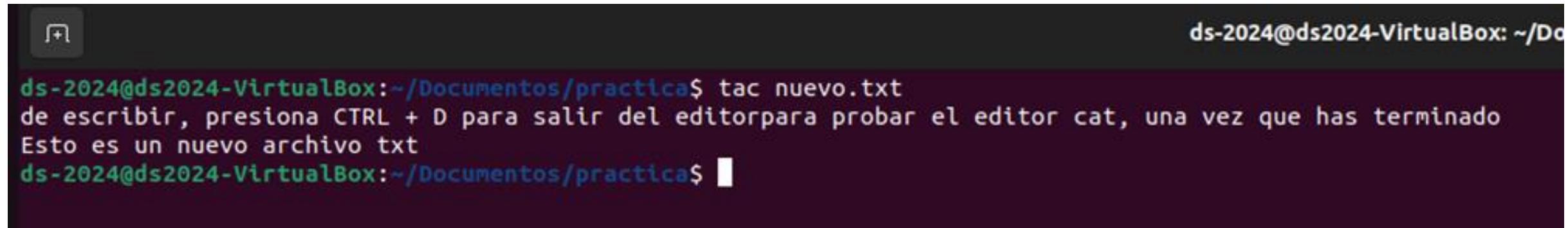
```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$
```

**cat nuevo.txt contenido.txt > nuevo2.txt :**

Combina el contenido de los archivos nuevo.txt y contenido.txt y los guarda en un nuevo fichero(archivo) llamado nuevo2.txt

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ cat nuevo.txt contenido.txt > nuevo2.txt
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ ls
contenido1.txt contenido2.txt contenido.txt nuevo2.txt nuevo.txt
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ cat nuevo2.txt
Esto es un nuevo archivo txt.
para probar el editor cat, una vez que has terminado
de escribir, presiona CTRL + D para salir del editorUna de las principales dificultades que surge en el ámbito del desarrollo de software es la incompatibilidad de las herramientas que
se van a utilizar con respecto a los diferentes sistemas operativos. Así, en muchas ocasiones, el software no es multiplataforma,
es decir, se ha creado para ejecutarse en un sistema operativo concreto. Esto genera problemas si el sistema operativo sobre el
que se va a trabajar no coincide con el que precisa el software que se quiere utilizar, por lo que en estos casos sería necesario
instalar el sistema operativo requerido por el softwareds-2024@ds2024-VirtualBox:~/Documentos/practica$ 
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ 
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ 
```

# tac nuevo.txt :



The screenshot shows a terminal window with a dark background and light-colored text. In the top right corner, the terminal title is "ds-2024@ds2024-VirtualBox: ~/Do". The main text area contains the command "ds-2024@ds2024-VirtualBox:~/Documentos/practica\$ tac nuevo.txt" followed by the content of the file: "de escribir, presiona CTRL + D para salir del editorpara probar el editor cat, una vez que has terminado Esto es un nuevo archivo txt". Below this, the prompt "ds-2024@ds2024-VirtualBox:~/Documentos/practica\$" is visible.

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ tac nuevo.txt
de escribir, presiona CTRL + D para salir del editorpara probar el editor cat, una vez que has terminado
Esto es un nuevo archivo txt
ds-2024@ds2024-VirtualBox:~/Documentos/practica$
```

Comando: file

Sintaxis: **file nombrearchivo.txt**

El comando file te permite comprobar un tipo de archivo, ya sea texto, imagen o binario.



```
ds-2024@ds2024-VirtualBox:~$ cd Documentos
ds-2024@ds2024-VirtualBox:~/Documentos$ cd practica
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ ls
contenido1.txt  contenido2.txt  contenido.txt
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ file contenido.txt
contenido.txt: Unicode text, UTF-8 text, with CRLF line terminators
ds-2024@ds2024-VirtualBox:~/Documentos/practica$
```

# Comandos zip y unzip

El comando zip te permite comprimir elementos en un archivo ZIP con la relación de compresión óptima. Sintaxis:  
zip [opciones] archivozip archivo1 archivo2

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ ls
contenido1.txt  contenido2.txt  contenido.txt  nuevo2.txt  nuevo.txt
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ zip comprimido nuevo.txt nuevo2.txt
adding: nuevo.txt (deflated 21%)
adding: nuevo2.txt (deflated 44%)
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ ls
comprimido.zip  contenido1.txt  contenido2.txt  contenido.txt  nuevo2.txt  nuevo.txt
ds-2024@ds2024-VirtualBox:~/Documentos/practica$
```

Utiliza el comando unzip para extraer el archivo comprimido.

Sintaxis: **unzip [opción] nombre\_archivo.zip**

**unzip comprimido.zip**

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ ls
comprimido.zip contenido1.txt contenido2.txt contenido.txt nuevo2.txt nuevo.txt
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ unzip comprimido.zip
Archive: comprimido.zip
replace nuevo.txt? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: nuevo.txt
replace nuevo2.txt? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: nuevo2.txt
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ 
ds-2024@ds2024-VirtualBox:~/Documentos/practica$
```

# Comando tar

El comando tar archiva varios elementos en un archivo TAR, un formato similar al ZIP con compresión opcional.

Sintaxis: **tar [opciones] [fichero\_archivo] [archivo de destino o directorio]**

# tar -cvzf comprimido3.tar /home/ds-2024/Documentos/practica



ds-2024@ds2024-VirtualBox: ~/Documentos/practica\$

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ ls
comprimido.zip contenido1.txt contenido2.txt contenido.txt nuevo2.txt nuevo.txt
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ # vamos a crear una carpeta llamada prueba2
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ mkdir prueba2
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ ls
comprimido.zip contenido1.txt contenido2.txt contenido.txt nuevo2.txt nuevo.txt prueba2
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ pwd
/home/ds-2024/Documentos/practica
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ # me voy al directorio donde quiero dejar mi archivo tar
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ cd prueba2
ds-2024@ds2024-VirtualBox:~/Documentos/practica/prueba2$ tar -cvzf comprimido3.tar /home/ds-2024/Documentos/practica
tar: Eliminando la '/' inicial de los nombres
/home/ds-2024/Documentos/practica/
/home/ds-2024/Documentos/practica/prueba2/
/home/ds-2024/Documentos/practica/prueba2/comprimido3.tar
/home/ds-2024/Documentos/practica/nuevo.txt
/home/ds-2024/Documentos/practica/comprimido.zip
/home/ds-2024/Documentos/practica/contenido.txt
/home/ds-2024/Documentos/practica/contenido2.txt
/home/ds-2024/Documentos/practica/nuevo2.txt
/home/ds-2024/Documentos/practica/contenido1.txt
ds-2024@ds2024-VirtualBox:~/Documentos/practica/prueba2$ ls
comprimido3.tar
ds-2024@ds2024-VirtualBox:~/Documentos/practica/prueba2$
```

```
tar -cvzf comprimido3.tar /home/ds-2024/Documentos/practica
```

- c: Crea un nuevo archivo tar.
- v: Muestra información detallada sobre el proceso de creación del archivo tar.
- z: Comprime el archivo tar usando el algoritmo gzip.
- f: Especifica el nombre del archivo tar.

# Otro ejemplo

**tar -cvf nuevo\_fichero.tar fichero1 directorio2 fichero3 :**

Crea “nuevo\_fichero.tar”, que contendrá “fichero1”, todo el “directorio2” (y sus subdirectorios) y “fichero3”.

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ ls
comprimido.zip contenido1.txt contenido2.txt contenido.txt nuevo2.txt nuevo.txt prueba2
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ ls prueba2
comprimido3.tar
ds-2024@ds2024-VirtualBox: ~/Documentos/practica$ tar cvf nuevo_comprimido.tar nuevo.txt /home/ds-2024/Documentos/practica/prueba2 contenido.txt
tar cvf: orden no encontrada
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ tar -cvf nuevo_comprimido.tar nuevo.txt /home/ds-2024/Documentos/practica/prueba2 contenido.txt
nuevo.txt
tar: Eliminando la '/' inicial de los nombres
/home/ds-2024/Documentos/practica/prueba2/
/home/ds-2024/Documentos/practica/prueba2/comprimido3.tar
tar: Eliminando la '/' inicial de los objetivos de los enlaces
contenido.txt
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ ls
comprimido.zip contenido1.txt contenido2.txt contenido.txt nuevo2.txt nuevo_comprimido.tar nuevo.txt prueba2
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ █
```

# tar -xvf fichero.tar

Extraerá el contenido de “fichero.tar” y lo dejará en el directorio en el que esté.

```
contenido.txt
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ ls
comprimido.zip contenido1.txt contenido2.txt contenido.txt nuevo2.txt nuevo_comprimido.tar nuevo.txt prueba2
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ tar -xvf nuevo_comprimido.tar
nuevo.txt
home/ds-2024/Documentos/practica/prueba2/
home/ds-2024/Documentos/practica/prueba2/comprimido3.tar
contenido.txt
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ ls
comprimido.zip contenido1.txt contenido2.txt contenido.txt home nuevo2.txt nuevo_comprimido.tar nuevo.txt prueba2
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ ls home
```

# Comando gzip : gzip fichero1

Comprime los ficheros que recibe como parámetros, eliminando los originales.

**gzip fichero1** creará en el mismo directorio la versión comprimida de “fichero1” (“fichero1.gz”) y eliminará “fichero1”



ds-2024@ds2024-VirtualB

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ ls
comprimido.zip contenido1.txt contenido2.txt contenido.txt nuevo2.txt nuevo.txt prueba2
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ gzip nuevo2.txt nuevo.txt
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ ls
comprimido.zip contenido1.txt contenido2.txt contenido.txt nuevo2.txt.gz nuevo.txt.gz prueba2
ds-2024@ds2024-VirtualBox:~/Documentos/practica$
```

# Comando gunzip

Realiza la operación inversa sobre los archivos comprimidos que recibe como parámetro, restaurando los originales.

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ ls  
comprimido.zip contenido1.txt contenido2.txt contenido.txt nuevo2.txt.gz nuevo.txt.gz prueba2  
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ gunzip nuevo2.txt  
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ ls  
comprimido.zip contenido1.txt contenido2.txt contenido.txt nuevo2.txt nuevo.txt.gz prueba2  
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ gunzip nuevo.txt  
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ ls  
comprimido.zip contenido1.txt contenido2.txt contenido.txt nuevo2.txt nuevo.txt prueba2  
ds-2024@ds2024-VirtualBox:~/Documentos/practica$
```

# Comando head

El comando `head` se utiliza para mostrar las primeras líneas de un archivo de texto. A continuación, se enumeran las opciones más comunes del comando `head`:

`-n`: Especifica el número de líneas que se deben mostrar.

# Ejemplo: head -n 3 contenido.txt

Muestra las 3 primeras líneas del archivo contenido.txt

```
ds-2024@ds2024-VirtualBox: ~/Documentos/practica
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ ls
comprimido.zip  contenido1.txt  contenido2.txt  contenido.txt  nuevo2.txt  nuevo_comprimido.tar  nuevo.txt  pruebaz
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ head -n 3 contenido.txt
Una de las principales dificultades que surge en el ámbito del desarrollo de software es la incompatibilidad de las herramientas que se van a utilizar con respecto a los diferentes sistemas operativos. Así, en muchas ocasiones, el software no es multiplataforma, es decir, se ha creado para ejecutarse en un sistema operativo concreto. Esto genera problemas si el sistema operativo sobre el
ds-2024@ds2024-VirtualBox:~/Documentos/practica$
```

**-c:** Especifica el número de bytes que se deben mostrar.

**head -c 10 contenido.txt**

Este comando mostrará los primeros 10 bytes del archivo "contenido.txt".

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ head -c 10 contenido.txt  
Una de lasds-2024@ds2024-VirtualBox:~/Documentos/practica$  
ds-2024@ds2024-VirtualBox:~/Documentos/practica$
```

# Comando tail

El comando tail muestra las diez últimas líneas de un archivo, lo que resulta útil para comprobar nuevos datos y errores.

Sintaxis: **tail [opción] archivo.txt**

# tail -2 contenido.txt

```
ds-2024@ds2024-VirtualBox: ~/Documentos/practica
```

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ tail -n contenido.txt
tail: el número de lineas no es válido: «contenido.txt»
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ tail -2 contenido.txt
que se va a trabajar no coincide con el que precisa el software que se quiere utilizar, por lo que en estos casos sería necesario
instalar el sistema operativo requerido por el softwareds-2024@ds2024-VirtualBox:~/Documentos/practica$
```

ds-2024@ds2024-VirtualBox:~/Documentos/practica\$

ds-2024@ds2024-VirtualBox:~/Documentos/practica\$ █

# Comando diff

Compara el contenido de dos archivos y muestra las diferencias. Se utiliza para alterar un programa sin modificar el código.

**diff [opción] archivo1 archivo2**

A continuación, se indican algunas opciones aceptables:

- c: muestra la diferencia entre dos archivos en un formulario contextual.
- u: muestra la salida sin información redundante.
- i: hace que el comando diff no distinga entre mayúsculas y minúsculas.

# diff contenido.txt nuevo2.txt

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ ls
comprimido.zip  contenido1.txt  contenido2.txt  contenido.txt  nuevo2.txt  nuevo_comprimido.tar  nuevo.txt  prueba2
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ diff contenido.txt nuevo2.txt
1c1,3
< Una de las principales dificultades que surge en el ámbito del desarrollo de software es la incompatibilidad de las herramientas que
---
> Esto es un nuevo archivo txt
> para probar el editor cat, una vez que has terminado
> de escribir, presiona CTRL + D para salir del editorUna de las principales dificultades que surge en el ámbito del desarrollo de software es la incompatibilidad de las herramientas que
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ █
```

# Otro ejemplo con -c:

diff -c contenido.txt nuevo2.txt

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ diff -c contenido.txt nuevo2.txt
*** contenido.txt      2024-04-03 20:17:46.000000000 +0200
--- nuevo2.txt 2024-04-04 09:22:02.000000000 +0200
*****
*** 1,4 ****
! Una de las principales dificultades que surge en el ámbito del desarrollo de software es la incompatibilidad de las herramientas que
se van a utilizar con respecto a los diferentes sistemas operativos. Así, en muchas ocasiones, el software no es multiplataforma,
es decir, se ha creado para ejecutarse en un sistema operativo concreto. Esto genera problemas si el sistema operativo sobre el
que se va a trabajar no coincide con el que precisa el software que se quiere utilizar, por lo que en estos casos sería necesario
--- 1,6 ----
! Esto es un nuevo archivo txt
! para probar el editor cat, una vez que has terminado
! de escribir, presiona CTRL + D para salir del editorUna de las principales dificultades que surge en el ámbito del desarrollo de software es la incompatibilidad de las herramientas que
se van a utilizar con respecto a los diferentes sistemas operativos. Así, en muchas ocasiones, el software no es multiplataforma,
es decir, se ha creado para ejecutarse en un sistema operativo concreto. Esto genera problemas si el sistema operativo sobre el
que se va a trabajar no coincide con el que precisa el software que se quiere utilizar, por lo que en estos casos sería necesario
ds-2024@ds2024-VirtualBox:~/Documentos/practica$
```

Con -u:

diff -u contenido.txt nuevo2.txt

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ diff -u contenido.txt nuevo2.txt
--- contenido.txt      2024-04-03 20:17:46.000000000 +0200
+++ nuevo2.txt    2024-04-04 09:22:02.000000000 +0200
@@ -1,4 +1,6 @@
-Una de las principales dificultades que surge en el ámbito del desarrollo de software es la incompatibilidad de las herramientas que
+Esto es un nuevo archivo txt
+para probar el editor cat, una vez que has terminado
+de escribir, presiona CTRL + D para salir del editorUna de las principales dificultades que surge en el ámbito del desarrollo de software es la incompatibilidad de las herramientas que
se van a utilizar con respecto a los diferentes sistemas operativos. Así, en muchas ocasiones, el software no es multiplataforma,
es decir, se ha creado para ejecutarse en un sistema operativo concreto. Esto genera problemas si el sistema operativo sobre el
que se va a trabajar no coincide con el que precisa el software que se quiere utilizar, por lo que en estos casos sería necesario
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ █
```

# Comando find

Se usa para buscar archivos dentro de un directorio concreto.

**find [opción] [ruta] [expresión]**

Ejemplo:

**find /home -name comprimido3.tar**

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ ls
comprimido.zip contenido1.txt contenido2.txt contenido.txt nuevo2.txt nuevo_comprimido.tar nuevo.txt prueba2
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ ls prueba2
comprimido3.tar
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ find /home -name comprimido3.tar
/home/ds-2024/Documentos/practica/prueba2/comprimido3.tar
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ find /home -name nuevo.txt
/home/ds-2024/Documentos/practica/nuevo.txt
ds-2024@ds2024-VirtualBox:~/Documentos/practica$
```

# Actividad 6. Investigar y desarrollar algunos ejemplos con los comandos (en Linux):

- kill
- apt-get
- sudo
- su
- df
- du
- ping
- top
- htop
- ps
- uname
- hostname
- time
- shutdown
- wget
- netstat
- history
- man
- echo
- cal
- ln

Realizar la entrega en un documento .docx y también incluye el fichero .txt con el histórico de los comandos utilizados. Se debe hacer al menos un ejemplo de cada comando.

# Edición de archivos: nano, vi

## Editor vi

Vi es un editor de texto de pantalla completa que está presente en la mayoría de los sistemas operativos tipo Unix y Linux. Su nombre proviene de "Visual Editor". Vi es conocido por su eficiencia y su capacidad para manejar archivos de texto grandes con facilidad, así como por su potente conjunto de características.

# Editor vi

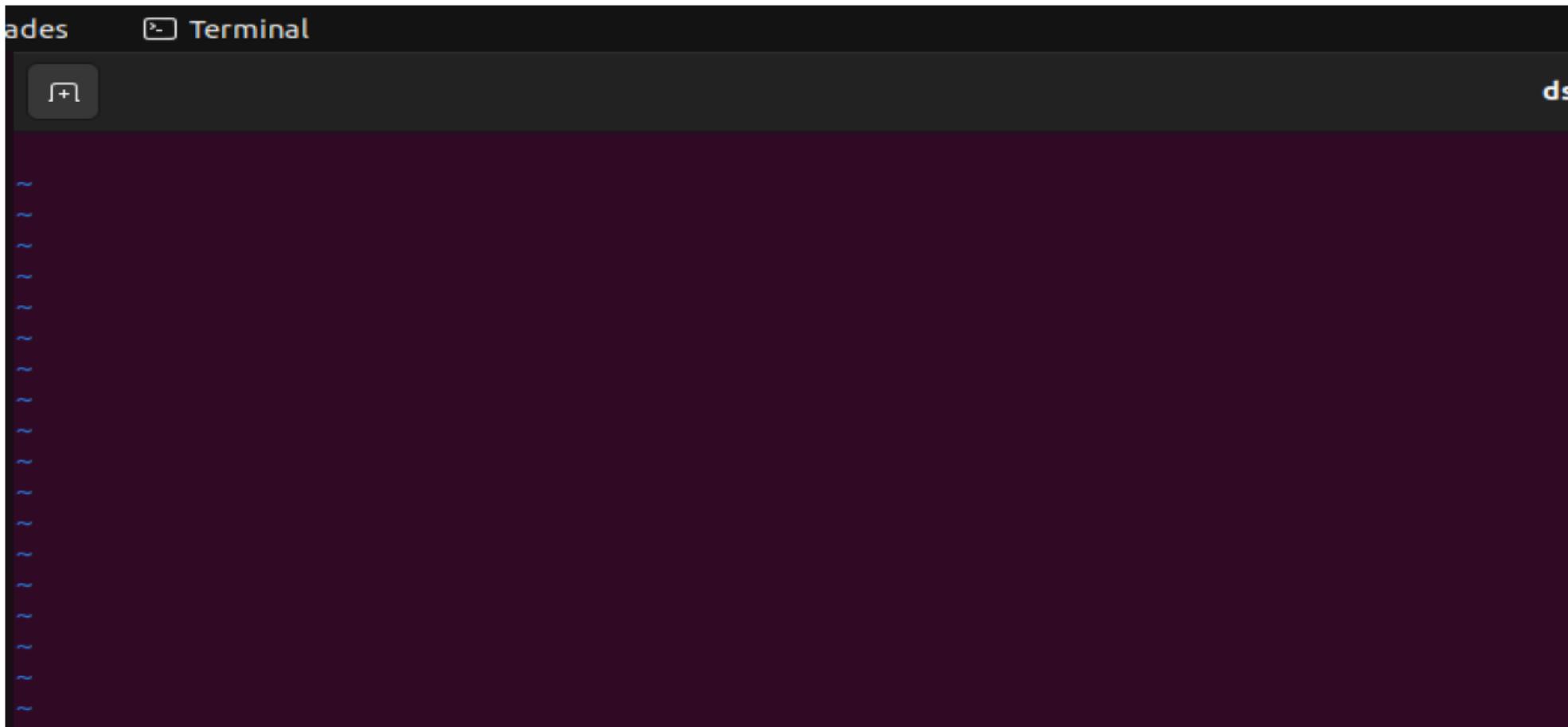
vi opera en dos modos principales: el modo de comando y el modo de inserción. En el modo de comando, los usuarios pueden navegar por el texto, realizar búsquedas, realizar ediciones y ejecutar comandos. En el modo de inserción, los usuarios pueden escribir y editar el texto de la manera habitual. Esta separación de modos permite una edición eficiente y rápida una vez que el usuario se familiariza con los comandos básicos de vi.

En el modo de edición, vi está esperando que se escriba el texto del fichero (por tanto, interpreta lo escrito como texto).

```
$/practica$ # Vamos a crear un archivo con nombre p.txt  
$/practica$ touch p.txt  
$/practica$ ls  
nido2.txt  contenido.txt  nuevo2.txt  nuevo_comprimido.tar  nuevo.txt  prueba2  p.txt  
$/practica$
```

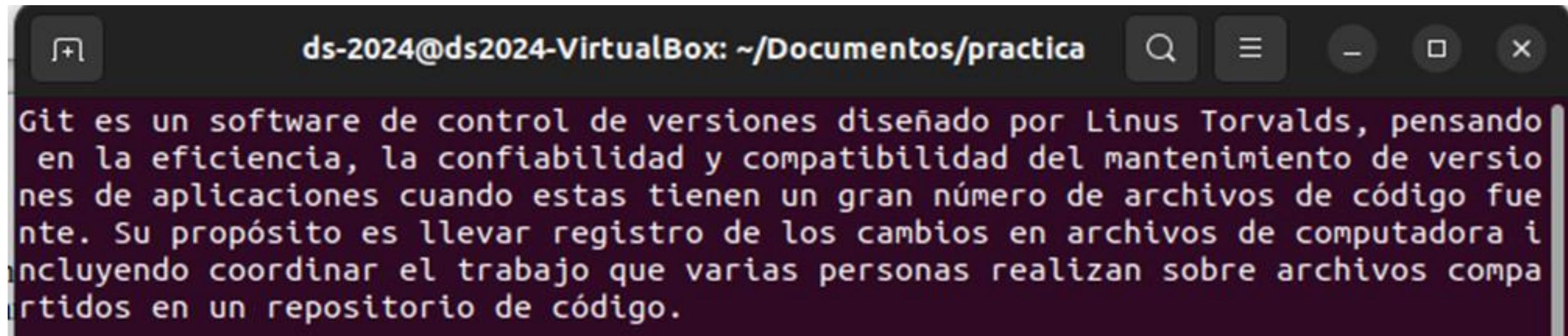
Para abrir un fichero con el editor vi basta con escribir:  
vi nombre\_archivo.algo  
Si el fichero no existe lo crea.

En este ejemplo abrimos el fichero p.txt escribiendo: **vi p.txt**



Hemos entrado al modo edición de vi donde, todo lo que escribas lo interpreta como texto que va ser añadido al fichero.

Vamos a escribir cualquier texto en el:

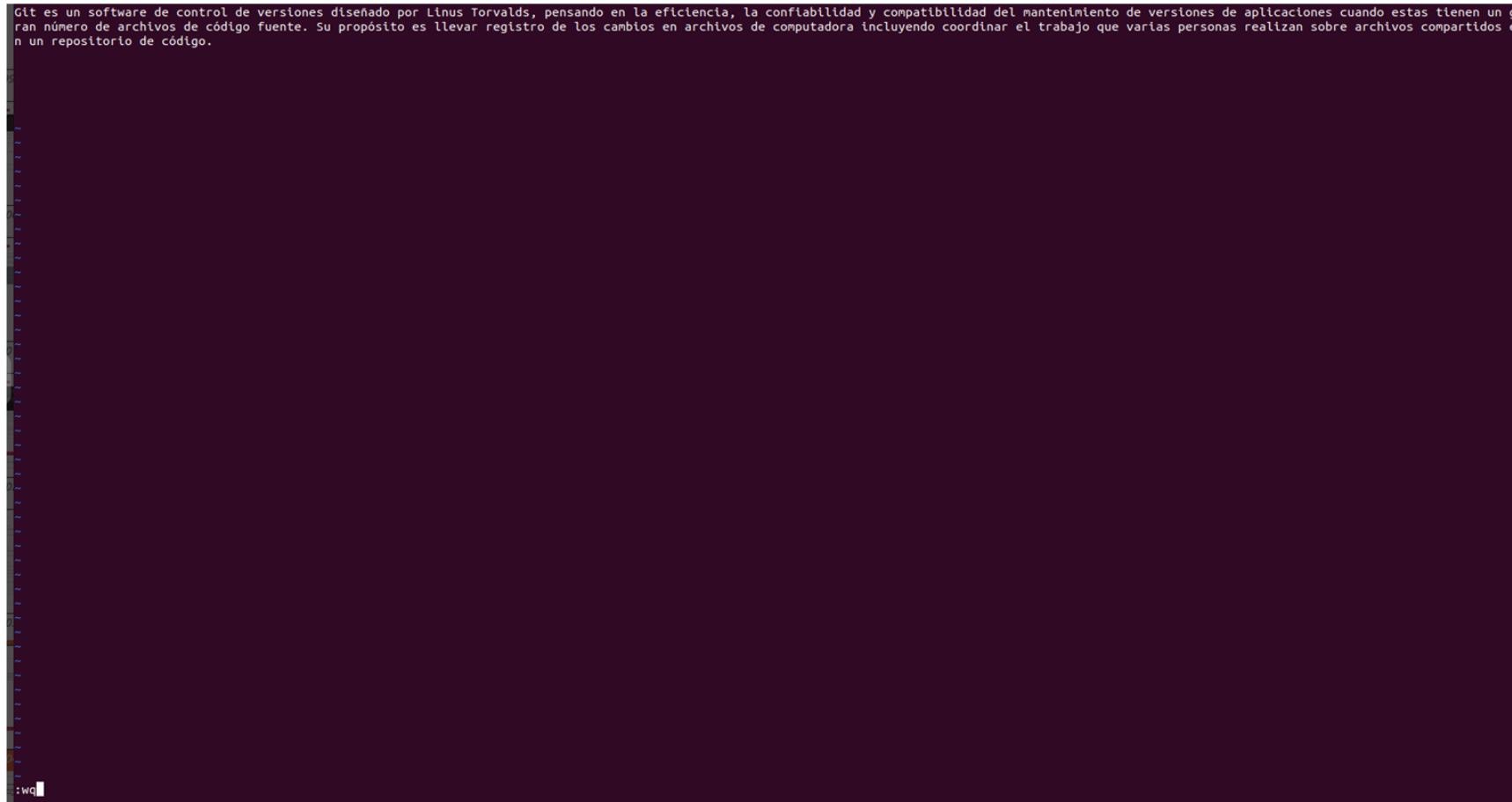


```
ds-2024@ds2024-VirtualBox: ~/Documentos/practica
```

Git es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia, la confiabilidad y compatibilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente. Su propósito es llevar registro de los cambios en archivos de computadora incluyendo coordinar el trabajo que varias personas realizan sobre archivos compartidos en un repositorio de código.

Para guardar y salir debemos entrar al modo de comando, para ello escribimos los dos puntos ‘:’

Luego de escribir los dos puntos ‘:’ **vi** te lleva al modo de comandos, que se encuentra en la última fila del editor.

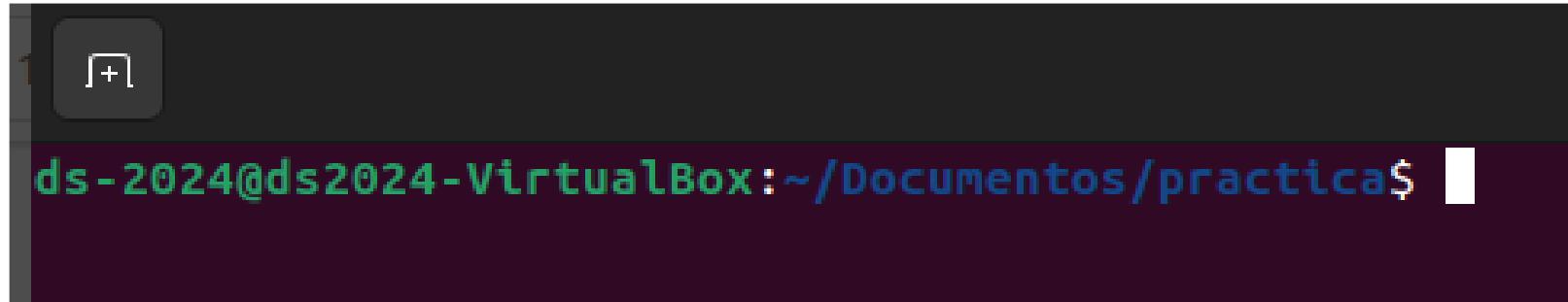


A screenshot of a terminal window showing the vi editor in command mode. The window has a dark background with white text. At the top, there is a status bar with some text in Spanish. The main area of the window is mostly blank, representing the content of the file being edited. In the bottom right corner, there is a small icon representing the command mode. The overall appearance is that of a standard Linux terminal interface.

Verás que vi está a la espera de instrucciones, para guardar y salir escribimos: wq y luego Enter.



Una vez que guarda y sale del editor, nos lleva de nuevo a la terminal

A screenshot of a terminal window. The window has a dark background with a light gray header bar. In the top-left corner of the header bar is a small square button containing a white plus sign (+). The main area of the terminal shows a command-line interface with a green prompt: "ds-2024@ds2024-VirtualBox:~/Documentos/practica\$". To the right of the prompt is a white vertical scroll bar.

Podemos visualizar el archivo con cat para verificar los cambios: **cat p.txt**

# Comandos de vi:

- i insertar antes del cursor
- a añadir detrás del cursor
- o añadir una línea en blanco
- x borrar un carácter
- j borrar el final de línea (une dos líneas)
- dd borrar la línea completa
- u deshacer la última edición
- :q salir
- :q! salir sin guardar
- :w guardar
- :wq guardar y salir
- :set nu muestra números de línea
- :set nonu oculta números de línea

# Actividad 7

Abre el editor Vi desde la terminal.

Crea un nuevo archivo llamado "p-1.txt" utilizando Vi.

Cambia al modo de inserción y escribe el siguiente texto:

Este es un ejercicio de práctica de Vi.

En este ejercicio, aplicaremos algunos de los principales comandos de Vi.

Espero que aprendas Vi.

Guarda los cambios y vuelve al modo de comando.

Navega por el texto utilizando los siguientes comandos:

Utiliza las teclas de dirección para moverte arriba, abajo, izquierda y derecha.

Usa las teclas "w" y "b" para moverte hacia adelante y

hacia atrás palabra por palabra.

Utiliza las teclas "o" y "\$" para ir al principio y al final de la línea, respectivamente.

Realiza las siguientes acciones de edición:

Elimina la segunda línea del texto.

Copia la primera línea y pégala al final del documento.

Cambia la palabra "Esperamos" por "Espero" en la segunda línea.

- Guarda los cambios y vuelve al modo de comando.
- Busca la palabra "disfrutes" en el texto.
- Reemplaza todas las ocurrencias de la palabra "Vi" por "vi" en el texto.

# Visualización de archivos con formato

El comando `pr file` imprime por consola el contenido de los archivos de una manera formateada, por columnas, controlando el tamaño de página y poniendo cabeceras al comienzo.

`pr file` produce una salida estándar de 66 líneas por página, con un encabezamiento de 5 líneas: 2 en blanco, 1 de identificación y otras 2 líneas en blanco.

# Opciones de pr

- pr -ln file produce una salida de n líneas por página.
- pr -F file hace una pausa para presentar la página, hasta que se pulsa Return para continuar.
- pr -t file suprime las 5 líneas del encabezamiento y las del final de página.
- pr -wn file ajusta la anchura de la línea a n posiciones.
- pr -d file lista el archivo con espaciado doble.
- pr -h `caracteres` file, el argumento o cadena de caracteres `caracteres` se convertirán en la cabecera del listado.
- pr +n file imprime el archivo a partir de la página n.

# Editor nano.

- Nano es un editor de texto ligero y fácil de usar.
- Es ideal para editar archivos de configuración y scripts.
- Está disponible en la mayoría de las distribuciones de Linux.

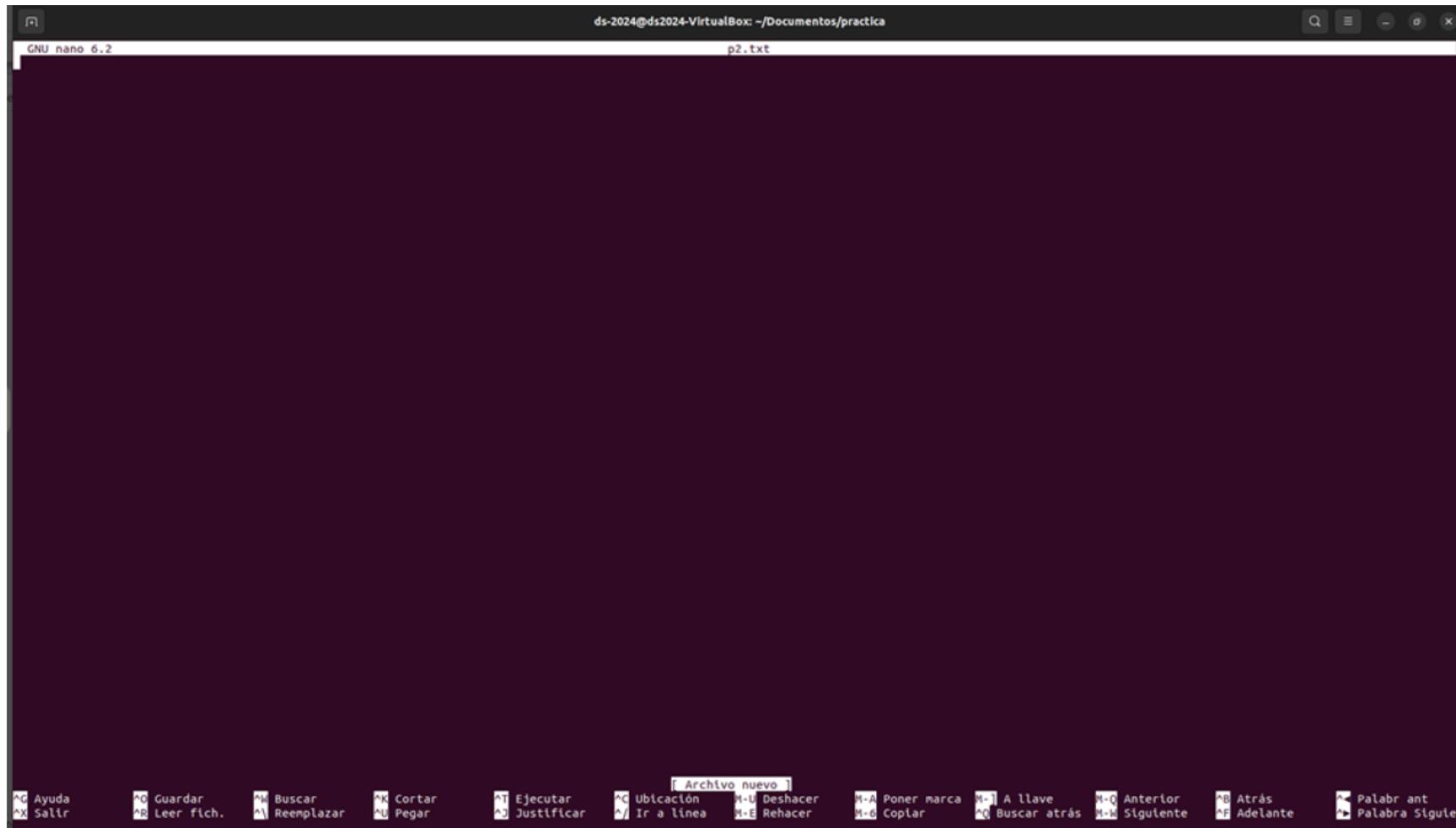
# Comandos básicos de nano:

- Ctrl+O: Guardar el archivo.
- Ctrl+X: Salir de nano.
- Ctrl+C: Interrumpir la operación actual.
- Ctrl+U: Deshacer el último cambio.
- Ctrl+R: Rehacer el último cambio.
- Ctrl+W: Buscar una cadena de texto.
- Ctrl+J: Justificar el texto.

Para obtener una lista completa de los comandos escribes en la consola: **man nano**

Ejemplo. Comenzamos por crear un archivo con nano, escribimos: nano p2.txt.

Se abrirá el editor de textos de nano:



**Nota:** También se abre el editor con solo escribir nano. Luego de hacer las ediciones se puede guardar con un nombre cualquiera.

# Escribe el siguiente texto en el editor:

Este es un archivo de ejemplo creado con Nano.

En esta clase aprenderemos a utilizar Nano para editar texto en Unix/Linux.

```
GNU nano 6.2
Este es un archivo de ejemplo creado con Nano.
En esta clase aprenderemos a utilizar Nano para editar texto en Unix/Linux.
```

Puedes usar las teclas de dirección para moverte por el texto y realizar ediciones necesarias.

Para guardar el archivo, presiona Ctrl + O.

# Ctrl + O

A screenshot of a terminal window titled "ds-2024@ds2024-VirtualBox: ~/Documentos/practica". The window shows the "GNU nano 6.2" text editor with the following content:

```
Este es un archivo de ejemplo creado con Nano.  
En esta clase aprenderemos a utilizar Nano para editar texto en Unix/Linux.
```

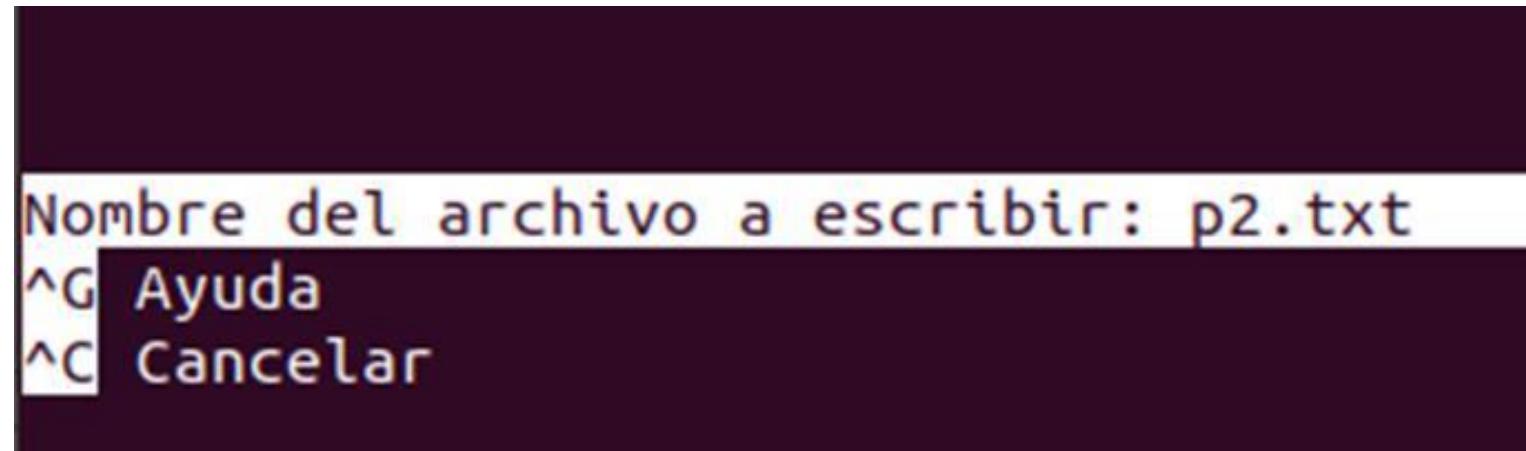
The terminal prompt at the bottom is "Nombre del archivo a escribir: p2.txt". Below the prompt, there is a menu bar with the following options:

- M-D Formato DOS
- M-M Formato Mac
- M-A Añadir
- M-P Anteponer
- M-B Respaldar fichero
- ^T Navegar

Below the menu bar, there are additional key bindings:

- ^A Ayuda
- ^C Cancelar

Se te pedirá que confirmes el nombre del archivo. Si es un nuevo archivo, puedes simplemente presionar Enter para confirmar el nombre predeterminado, o puedes escribir un nombre diferente y luego presionar Enter.



Una vez que el archivo está guardado, puedes salir de Nano presionando Ctrl + X.

# Abrir un Archivo Existente en Nano

Para abrir un archivo existente en Nano, simplemente escribe `sudo nano nombre_del_archivo` en la terminal y presiona Enter.

Se abrirá el archivo especificado en Nano y podrás editarlo de la misma manera que lo hiciste con el nuevo archivo.

# Búsqueda y Reemplazo de Texto

Para buscar una palabra específica en el texto, presiona **Ctrl + W**.

- Se abrirá un campo de búsqueda en la parte inferior de la ventana. Escribe la palabra que deseas buscar y presiona Enter.
- Nano resaltará la primera ocurrencia de la palabra buscada.
- Para reemplazar una palabra, presiona Ctrl + \.
- Se abrirá un campo de reemplazo en la parte inferior de la ventana. Escribe la palabra con la que deseas reemplazar y presiona Enter.
- Nano reemplazará la palabra resaltada (o la próxima ocurrencia si hay más de una) con la palabra de reemplazo.

# Copiar y Pegar Bloques de Texto

Para copiar un bloque de texto, mueve el cursor al principio del bloque y presiona **Ctrl + ^**

- Luego, mueve el cursor al final del bloque y presiona Alt + A.
- El bloque de texto se resaltará.
- Presiona Ctrl + K para cortar el bloque de texto resaltado.
- Mueve el cursor a la posición donde deseas pegar el bloque de texto.
- Presiona Ctrl + U para pegar el bloque de texto.

# Guardado Automático y Recuperación de Archivos

Nano tiene la capacidad de guardar automáticamente el contenido del archivo. Para activar esta función, presiona Alt + O.

- Aparecerá un mensaje indicando que se activó el guardado automático.
- Si por alguna razón el editor Nano se cierra inesperadamente o se produce un fallo, la próxima vez que abras el mismo archivo con Nano, el programa te preguntará si deseas recuperar el contenido no guardado del archivo.

# Actividad 8. Práctica de Nano: Edición y Funcionalidades Avanzadas

Abre un nuevo archivo en nano con el nombre "ejercicio\_practica.txt" utilizando el comando:  
**sudo nano ejercicio\_practica.txt** en la terminal.

- Escribe el siguiente texto en el archivo:

*Este es un ejercicio de práctica de Nano.*

*En este ejercicio, aplicaremos las funcionalidades avanzadas del editor de texto Nano.*

*Exploraremos la búsqueda y reemplazo de texto, así como el copiado y pegado de bloques de texto.*

*Espero que aprendas con este ejercicio.*

## Actividad 8. Práctica de Nano: Edición y Funcionalidades Avanzadas

- Utiliza el comando de búsqueda para encontrar la palabra "Nano" en el texto.
- Reemplaza todas las ocurrencias de la palabra "Nano" por "nano" en el texto.
- Copia el segundo párrafo ("En este ejercicio...") y pégalo al final del documento.
- Guarda el archivo con los cambios realizados utilizando Ctrl + O.
- Cierra Nano utilizando Ctrl + X.

# Búsqueda en archivos: grep

El comando grep es una herramienta poderosa en sistemas Linux que se utiliza para buscar patrones dentro de archivos de texto. Su nombre proviene de "global regular expression print", lo que refleja su función principal de búsqueda de expresiones regulares y mostrar las líneas que coinciden.

El comando **grep 'conjunto de caracteres' file1 file2 file3** busca una palabra, clave o frase en un conjunto de ficheros y muestra las líneas que contienen el texto buscado. Si el conjunto de caracteres está compuesto por dos o más palabras separadas por un espacio, se debe escribir entre apóstrofes, ', para evitar que la consola interprete esas palabras como otros parámetros del comando grep.

# Uso básico de grep

Sintaxis:  
grep ‘palabra o frase’  
nombre\_archivo.algo

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ grep "Linux" samurai.txt
A Shell Samurai has an intuitive understanding of the CLI and Linux
fundamentals. They can navigate the Command Line and configure the Linux
Let's learn Linux and kick ass at it together!
use to learn Linux fundamentals more deeply. To assist with that goal, I strongly
give. Without a real working Linux system, you're just kinda reading words and
CLI, but it is much quicker and most production Linux systems you touch will
not offer a GUI. If you decide to install Linux on an old computer or laptop that
options for getting a Linux shell at your disposal. We won't go deep on every one
of these options, but we'll guide you towards victory. A Linux Samurai has to do
Install Linux on an old Laptop or Desktop computer
Installing Linux on an old system is a great option to get started using Linux.
Installing Linux and using it as your daily driver is a great way to dive into the
deep end. Just be cautious as many newbies may install Linux on their “main
install Linux on a machine that isn't your primary computer.
Use Windows Subsystem for Linux or WSL
Linux (WSL) is a feature of Windows that allows you to run a Linux environment
Windows Subsystem for Linux is probably the easiest way to get a Linux shell
up quickly, but I don't totally recommend it because it isn't a “true” Linux
Here's how to get a Linux Shell using WSL:
• Enable the Windows Subsystem for Linux feature:
- Check the box next to “Windows Subsystem for Linux”
for Linux.
Linux system on your laptop or desktop. This option is flexible but requires a
you need to and run any ISO image and flavor of Linux. The Virtual Machine is
Dual Booting is a way of installing Linux alongside another operating system
on the same machine. This is a similar route to just installing Linux on a
Point is, any basic Linux server shouldn't cost more than $5/month. For some
server, be sure to use the latest version of Ubuntu, which is the Linux
The path you take to get access to a Linux Shell doesn't matter too much. All
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ █
```

# Uso de grep con varios archivos

- Ejemplo: grep “install Linux” samurai.txt samurai2.txt

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ grep "install Linux" samurai.txt samurai2.txt
samurai.txt:not offer a GUI. If you decide to install Linux on an old computer or laptop that
samurai.txt:deep end. Just be cautious as many newbies may install Linux on their "main
samurai.txt:install Linux on a machine that isn't your primary computer.
samurai2.txt:not offer a GUI. If you decide to install Linux on an old computer or laptop that
samurai2.txt:deep end. Just be cautious as many newbies may install Linux on their "main
samurai2.txt:install Linux on a machine that isn't your primary computer.
ds-2024@ds2024-VirtualBox:~/Documentos/practica$
```

# Opciones comunes

- La opción `-i` permite realizar la búsqueda sin distinguir entre mayúsculas y minúsculas. Por ejemplo:

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ grep -i "Shell" samurai.txt
What is a Shell Samurai?
A Shell Samurai has an intuitive understanding of the CLI and Linux
OS to do their bidding. A Shell Samurai tells computers what to do, not the other
A Shell Samurai does not know everything and never claims to. They are
A Shell Samurai never stops learning. They're always improving their skills
Thank you for joining me and taking the path to becoming a Shell Samurai!
hallucinating what a system should behave like. You need to use a real shell and
options for getting a Linux shell at your disposal. We won't go deep on every one
Windows Subsystem for Linux is probably the easiest way to get a Linux shell
Here's how to get a Linux Shell using WSL:
you can access your shell is by opening Command Prompt or Powershell and
The path you take to get access to a Linux Shell doesn't matter too much. All
that matters is that you start and get a basic bash shell up and running.
ds-2024@ds2024-VirtualBox:~/Documentos/practica$
```

# Opciones comunes

- La opción **-r** se utiliza para realizar una búsqueda recursiva en todos los archivos de un directorio. Por ejemplo:

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ grep -r "terminal" /home/ds-2024/Documentos/practica
/home/ds-2024/Documentos/practica/samurai.txt:has a Window manager, you'll just need to use the terminal program to follow
/home/ds-2024/Documentos/practica/samurai.txt:password will not show up on the terminal! This is called blind typing and
/home/ds-2024/Documentos/practica/samurai.txt:• Congrats you should now have your very own Ubuntu terminal up and
/home/ds-2024/Documentos/practica/samurai2.txt:has a Window manager, you'll just need to use the terminal program to follow
/home/ds-2024/Documentos/practica/samurai2.txt:password will not show up on the terminal! This is called blind typing and
/home/ds-2024/Documentos/practica/samurai2.txt:• Congrats you should now have your very own Ubuntu terminal up and
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ █
```

# Opciones comunes

La opción -v se utiliza para mostrar las líneas que no coinciden con el patrón especificado. Por ejemplo:

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ grep -v "Linux" samurai.txt
What is a Shell Samurai?
OS to do their bidding. A Shell Samurai tells computers what to do, not the other
way around.
A Shell Samurai does not know everything and never claims to. They are
resourceful, creative and modest. When an issue pops up that they can't solve,
they're able to finding documentation or resources to push through to victory.
A Shell Samurai never stops learning. They're always improving their skills
and learning more.
Thank you for joining me and taking the path to becoming a Shell Samurai!
- Stetson Blake
```

# Búsqueda con salida de conteo

La opción -l se utiliza para mostrar solo los nombres de los archivos que contienen el patrón especificado. Por ejemplo:

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ grep -l "Linux" *.txt
p2.txt
samurai2.txt
samurai.txt
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ █
```

# Búsqueda con salida de impresión de línea

La opción -n se utiliza para mostrar las líneas que contienen el patrón especificado.

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ grep -n "Linux" samurai.txt
2:A Shell Samurai has an intuitive understanding of the CLI and Linux
3:fundamentals. They can navigate the Command Line and configure the Linux
12:Let's learn Linux and kick ass at it together!
18:use to learn Linux fundamentals more deeply. To assist with that goal, I strongly
20:give. Without a real working Linux system, you're just kinda reading words and
25:CLI, but it is much quicker and most production Linux systems you touch will
26:not offer a GUI. If you decide to install Linux on an old computer or laptop that
30:options for getting a Linux shell at your disposal. We won't go deep on every one
31:of these options, but we'll guide you towards victory. A Linux Samurai has to do
33:Install Linux on an old Laptop or Desktop computer
34:Installing Linux on an old system is a great option to get started using Linux.
35:Installing Linux and using it as your daily driver is a great way to dive into the
36:deep end. Just be cautious as many newbies may install Linux on their "main"
39:install Linux on a machine that isn't your primary computer.
47:Use Windows Subsystem for Linux or WSL
49:Linux (WSL) is a feature of Windows that allows you to run a Linux environment
51:Windows Subsystem for Linux is probably the easiest way to get a Linux shell
52:up quickly, but I don't totally recommend it because it isn't a "true" Linux
58:Here's how to get a Linux Shell using WSL:
59:+ Enable the Windows Subsystem for Linux feature:
62:- Check the box next to "Windows Subsystem for Linux"
84:for Linux.
87:Linux system on your laptop or desktop. This option is flexible but requires a
89:you need to and run any ISO image and flavor of Linux. The Virtual Machine is
95:Dual Booting is a way of installing Linux alongside another operating system
96:on the same machine. This is a similar route to just installing Linux on a
107:Point is, any basic Linux server shouldn't cost more than $5/month. For some
113:server, be sure to use the latest version of Ubuntu, which is the Linux
115:The path you take to get access to a Linux shell doesn't matter too much. All
ds-2024@ds2024-VirtualBox:~/Documentos/practica$
```

# Operador pipe “|”

El operador pipe (|) ,llamado tubería, es una característica poderosa de la terminal de Linux que permite la comunicación entre dos comandos al enviar la salida de uno como entrada al otro. Esto facilita el procesamiento y la manipulación de datos de forma eficiente y dinámica:

Función del operador pipe: Permite conectar la salida estándar (stdout) de un comando con la entrada estándar (stdin) de otro comando. Esto significa que el resultado producido por el primer comando se utiliza como entrada para el segundo comando.

# Sintaxis básica de “|”

La sintaxis para usar el operador de tubería es **comando1 | comando2**, donde comando1 es el comando cuya salida queremos utilizar como entrada para comando2.

Ejemplo: **ls | grep "archivo"**

Busca archivos en un directorio cuyos nombres contienen la palabra "archivo". Este comando listaría todos los archivos en el directorio actual y mostraría solo aquellos cuyos nombres coinciden con el patrón "archivo".

# ls | grep "samurai"

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ ls | grep "samurai"
samurai2.txt
samurai.txt
ds-2024@ds2024-VirtualBox:~/Documentos/practica$
```

# Aplicaciones comunes del operador pipe |

El operador pipe (|) en la terminal de Linux se utiliza no solo para pasar la salida de un comando como entrada a otro comando, sino también para una variedad de otras aplicaciones comunes que facilitan el manejo de datos y la automatización de tareas.

Algunas de las aplicaciones más comunes del operador de tubería:

**1. Filtrado de resultados:** El operador pipe se usa comúnmente junto con el comando grep para filtrar la salida de un comando en función de un patrón de búsqueda. Ejemplo:

```
ls | grep "samurai"
```

# Aplicaciones comunes del operador pipe |

**2. Redirección de salida:** El operador pipe se puede utilizar junto con la redirección de salida (>) para enviar la salida combinada de varios comandos a un archivo. Por ejemplo:

Supongamos que queremos listar los archivos en el directorio actual y guardar esa lista en un archivo llamado "**lista\_archivos.txt**". Podemos hacerlo utilizando el comando **ls** para listar los archivos y luego redirigir la salida a un archivo utilizando el operador pipe (|) junto con la redirección de salida (>):

# ls | tee lista\_archivos.txt

Este comando ejecutará ls para listar los archivos en el directorio actual y luego utilizará tee para mostrar la salida en la pantalla y, al mismo tiempo, redirigirá la salida al archivo "lista\_archivos.txt"

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ ls | tee "lista_archivos.txt"
comprimido.zip
contenido1.txt
contenido2.txt
contenido.txt
data.txt
ejemplo.csv
nuevo2.txt
nuevo_comprimido.tar
nuevo.txt
p-1.txt
p2.txt
prueba2
p.txt
samurai2.txt
samurai.txt
```

# ls | tee lista\_archivos.txt

Luego puedes verificar que el archivo "lista\_archivos.txt" haya sido creado y contenga la lista de archivos del directorio actual ejecutando cat:

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ cat lista_archivos.txt
comprimido.zip
contenido1.txt
contenido2.txt
contenido.txt
data.txt
ejemplo.csv
nuevo2.txt
nuevo_comprimido.tar
nuevo.txt
p-1.txt
p2.txt
prueba2
p.txt
samurai2.txt
samurai.txt
ds-2024@ds2024-VirtualBox:~/Documentos/practica$
```

# ls | tee lista\_archivos.txt

Si escribes **ls** puedes ver que el archivo: “lista\_archivos.txt” está creado:

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ ls
comprimido.zip  contenido2.txt  data.txt    lista_archivos.txt  nuevo_comprimido.tar  p-1.txt  prueba2  samurai2.txt
contenido1.txt  contenido.txt   ejemplo.csv  nuevo2.txt       nuevo.txt           p2.txt  p.txt    samurai.txt
ds-2024@ds2024-VirtualBox:~/Documentos/practica$
```

# Aplicaciones comunes del operador de tubería

**3. Conteo de resultados:** Se puede utilizar el operador pipe junto con el comando **wc -l** para contar el número de líneas en la salida de un comando, la sintaxis:

**comando1 | wc -l**

Por ejemplo: Supongamos que queremos contar el número de archivos en un directorio específico. Podemos hacerlo utilizando el comando **ls** para listar los archivos y luego contar el número de archivos en la salida utilizando el comando **wc -l** junto con el operador pipe (**|**):

# ls | wc -l

Este comando ejecutará ls para listar los archivos en el directorio actual y luego utilizará wc -l para contar el número de líneas en la salida, que corresponde al número de archivos en el directorio.

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ ls | wc -l
16
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ ls
comprimido.zip contenido2.txt data.txt lista_archivos.txt nuevo_comprimido.tar p-1.txt prueba2 samurai2.txt
contenido1.txt contenido.txt ejemplo.csv nuevo2.txt nuevo.txt p2.txt p.txt samurai.txt
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ █
```

# Encadenamiento de múltiples comandos

El operador de tubería (|) en la terminal de Linux también permite encadenar múltiples comandos, lo que significa que la salida de un comando se utiliza como entrada para otro comando, y así sucesivamente. Esto permite construir flujos de trabajo complejos y realizar múltiples tareas de forma eficiente en una sola línea de comando.

- 1. Encadenamiento de dos comandos:** La sintaxis básica para encadenar dos comandos es **comando1 | comando2**, donde la salida del comando1 se pasa como entrada al comando2.

## 2. Encadenamiento de más de dos comandos:

Se pueden encadenar más de dos comandos utilizando múltiples operadores de tubería consecutivos. La sintaxis sería:

**comando1 | comando2 | comando3 |...**

Supongamos que queremos listar todos los archivos en un directorio, filtrar los archivos cuyo nombre contienen la palabra "documento", y luego ordenar esa lista alfabéticamente. Podemos hacerlo encadenando tres comandos: **ls, grep y sort**.

# ls | grep "p" | sort

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ ls | grep "p" | sort
comprimido.zip
ejemplo.csv
nuevo_comprimido.tar
p-1.txt
p2.txt
prueba2
p.txt
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ ls | grep "arch" | sort
lista_archivos.txt
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ ls | grep "samurai" | sort
samurai2.txt
samurai.txt
ds-2024@ds2024-VirtualBox:~/Documentos/practica$
```

```
ls | grep "p" | sort
```

Este comando primero ejecutará **ls** para listar todos los archivos en el directorio actual. Luego, la salida de ls se pasará al comando **grep**, que filtrará solo los archivos cuyo nombre contienen la palabra "documento". Finalmente, la salida de grep se pasará al comando **sort**, que ordenará la lista de archivos resultante alfabéticamente.

Actividad 9. Crear una carpeta con nombre act9 y dentro de ella guardar todos los archivos necesarios y utilizados en esta actividad. Las respuestas deben ser entregadas en .docx junto a la captura de salida de la terminal. Además, incluye el histórico de los comandos utilizados en un fichero .txt o .log. Todo deber entregado en un archivo comprimido en .zip utilizando la terminal.

1. Contar el número de líneas en un archivo de texto.
2. Ordenar un listado de nombres de archivos por orden alfabético
3. Buscar archivos con una extensión específica en un directorio y contarlos
4. Mostrar solo las líneas únicas en un archivo de texto
5. Mostrar solo los nombres de los archivos con extensión ".txt" en un directorio y ordenarlos alfabéticamente
6. Buscar archivos modificados hoy en un directorio y contarlos
7. Mostrar las líneas únicas en un archivo de texto y contarlas
8. Filtrar las líneas que contienen una palabra específica en un archivo, ordenarlas alfabéticamente y contarlas.
9. Contar la cantidad de palabras únicas en un archivo de texto, luego ordenarlas alfabéticamente.
10. Filtrar líneas que contienen una dirección IP válida en un archivo de registro y contarlas. Detalles del ejercicio 10 en la siguiente página.

# Ejercicio 10.

Debes crear (desde la terminal) un archivo con nombre: archivo.log y escribir el siguiente contenido:

[2022-04-01 12:30:45] Conexión desde la dirección IP 192.168.1.1

[2022-04-01 12:31:20] Acceso a la página desde la dirección IP 10.0.0.1

[2022-04-01 12:32:05] Error de autenticación desde la dirección IP 192.168.1.2

[2022-04-01 12:33:10] Conexión exitosa desde la dirección IP 172.16.0.1

[2022-04-01 12:34:25] Conexión desde la dirección IP 300.400.500.600 (IP inválida)

[2022-04-01 12:35:00] Intento de acceso desde la dirección IP 192.168.300.1 (IP inválida)

[2022-04-01 12:36:15] Error de conexión desde la dirección IP 999.999.999.999 (IP inválida)

Comprobar si la solución correcta (o alternativa) es:

```
cat archivo.log | grep -E "\b(25[0-5]|2[0-4][0-9]|01)?[0-9][0-9]?\b\.(25[0-5]|2[0-4][0-9]|01)?[0-9][0-9]?\b\.(25[0-5]|2[0-4][0-9]|01)?[0-9][0-9]?\b\.(25[0-5]|2[0-4][0-9]|01)?[0-9][0-9]?\b" | sort | uniq | wc -l
```





































