

Modulo 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.

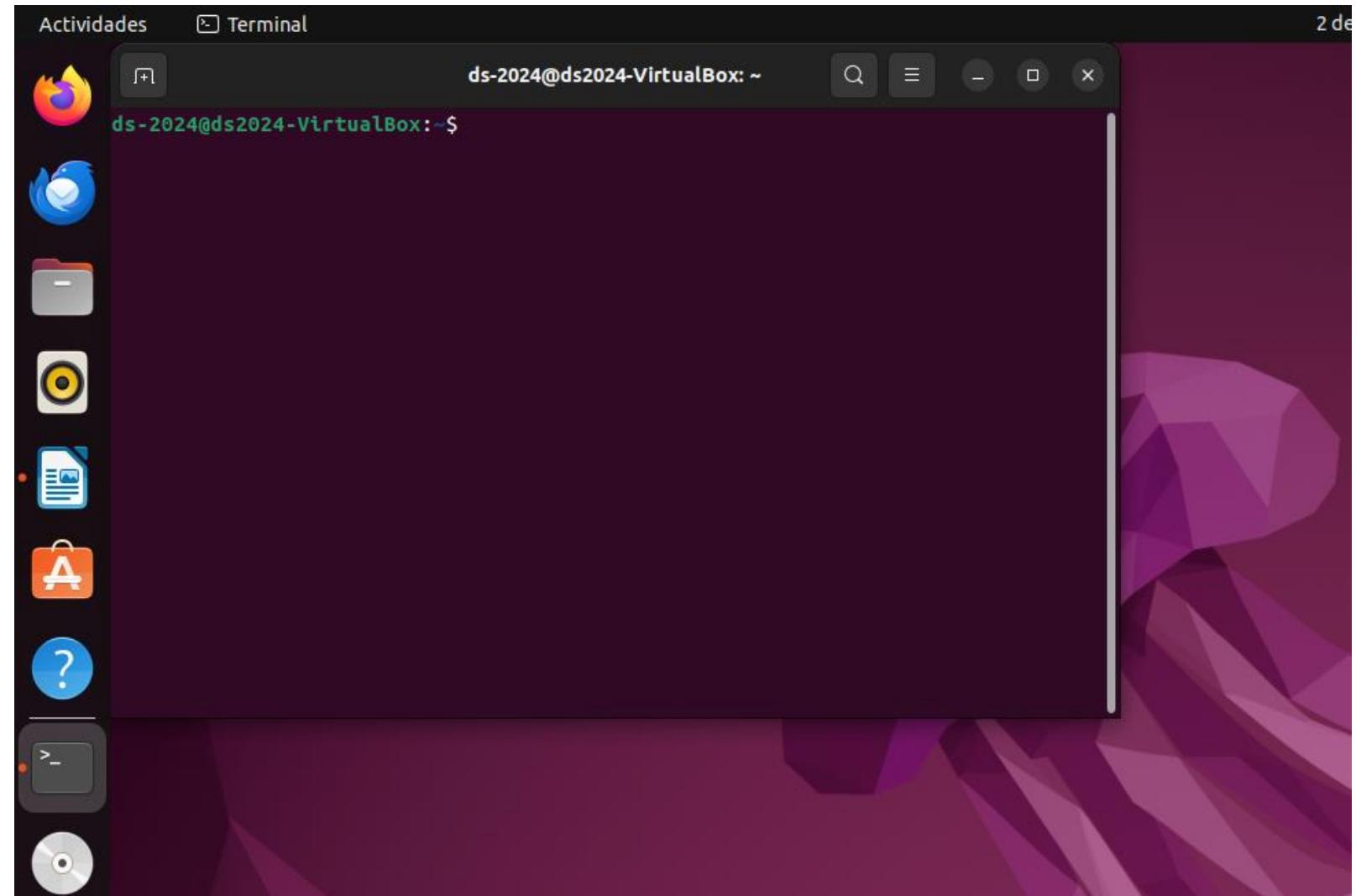
Conceptos básicos. Shell de comandos

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.

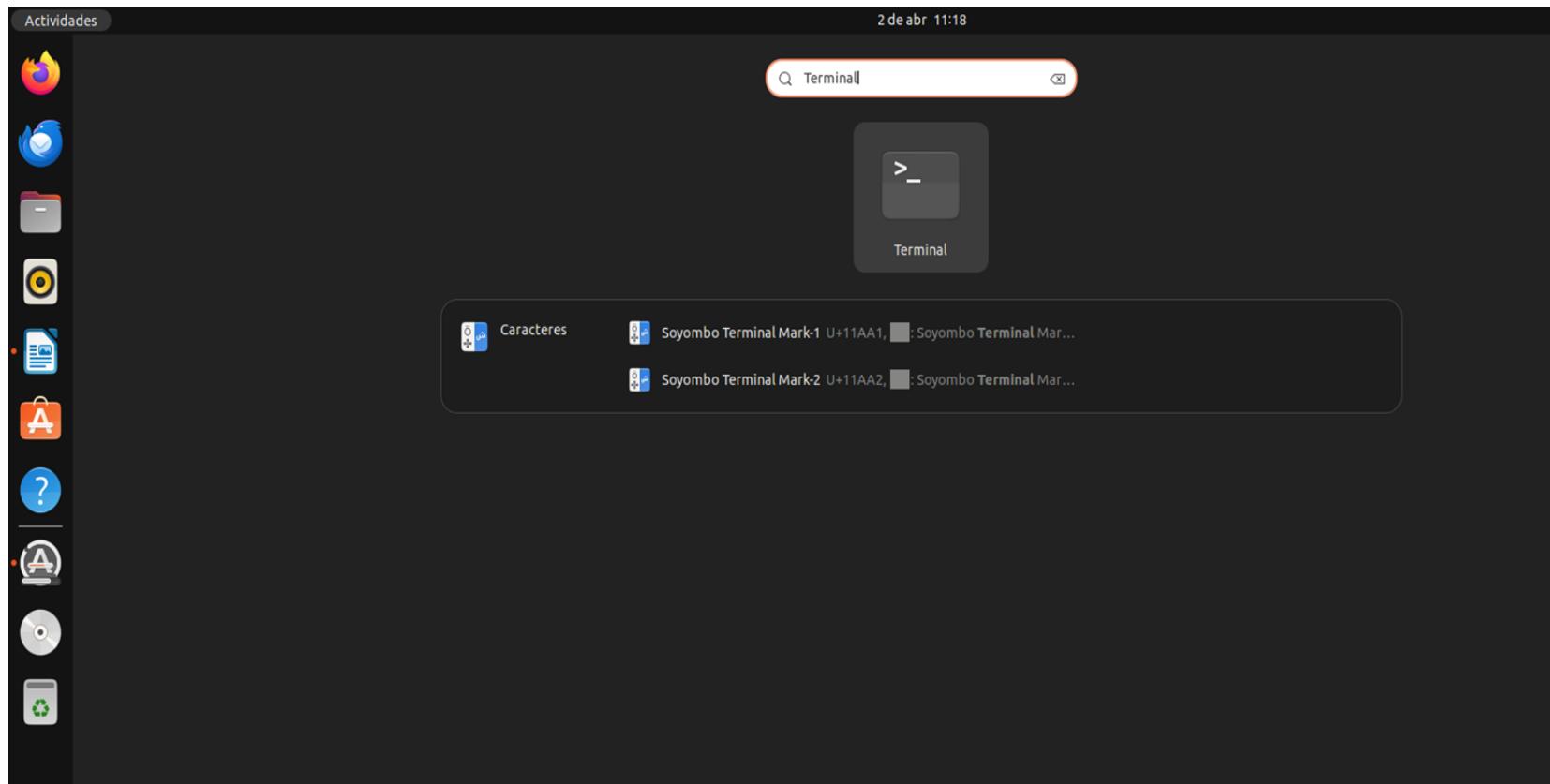
Conceptos básicos. Shell de comandos

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

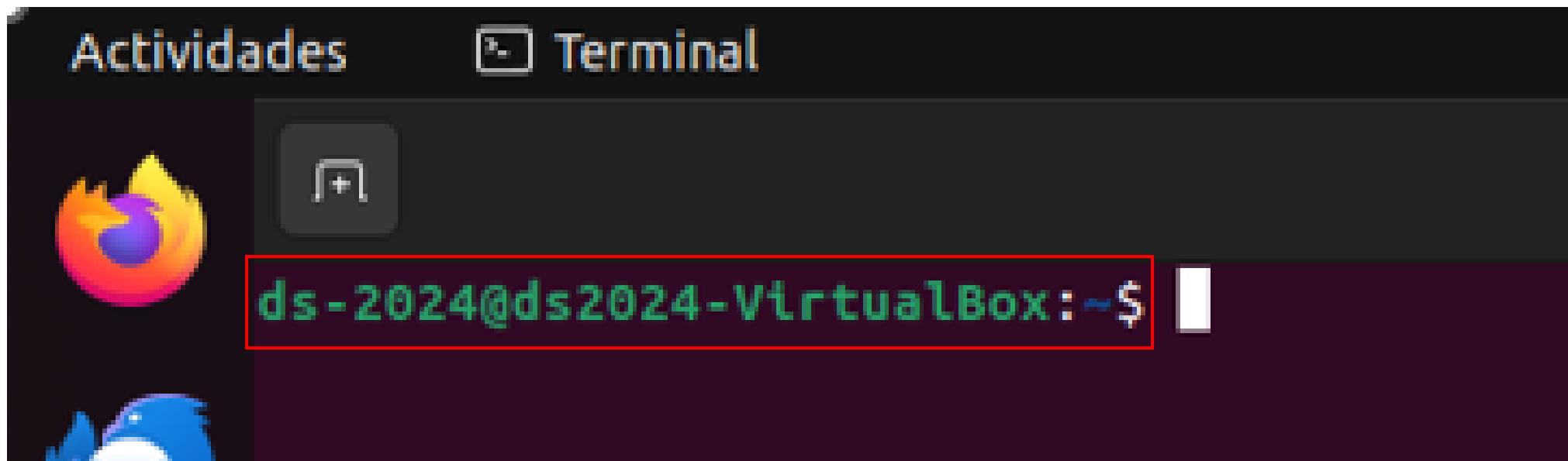


o bien se pulsa en un icono
que aparece en la parte
inferior izquierda,
de la barra de
herramientas inferior del
escritorio de Ubuntu



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.



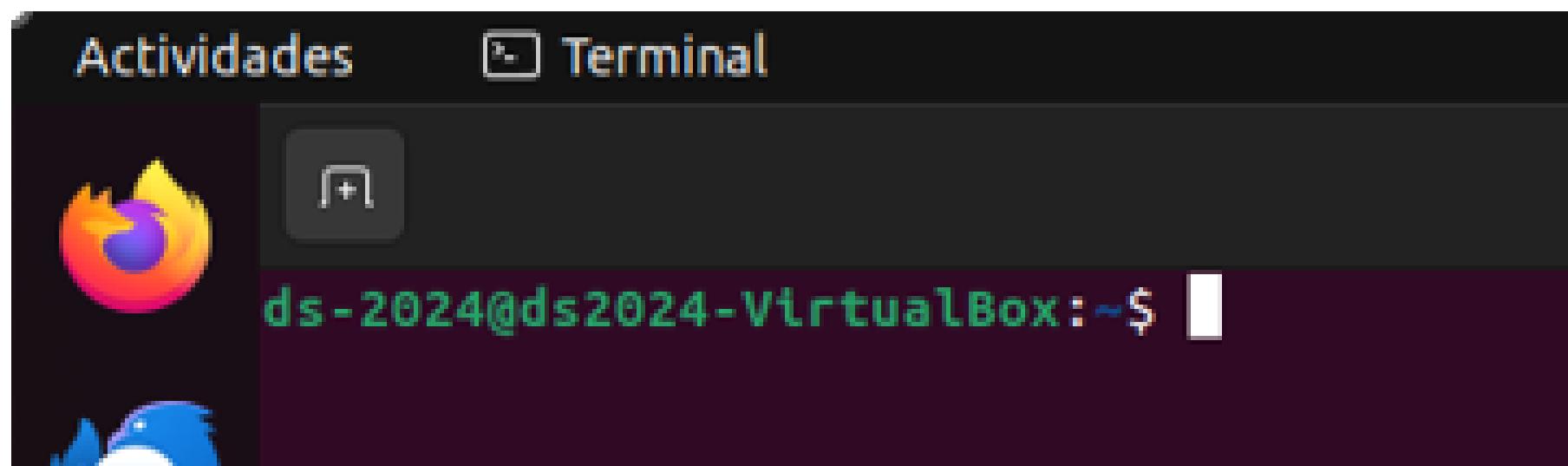
Al comienzo de la línea de órdenes aparece el usuario en la forma **usuario@máquina:directorio\$**, donde:

“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

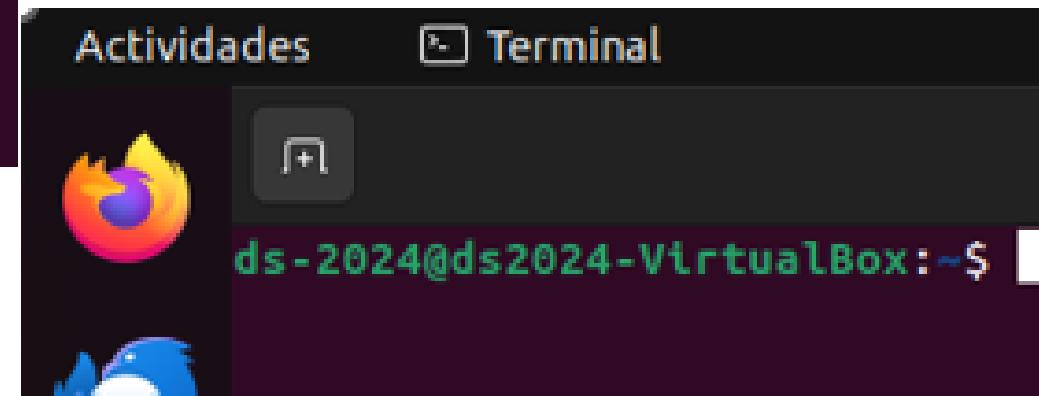
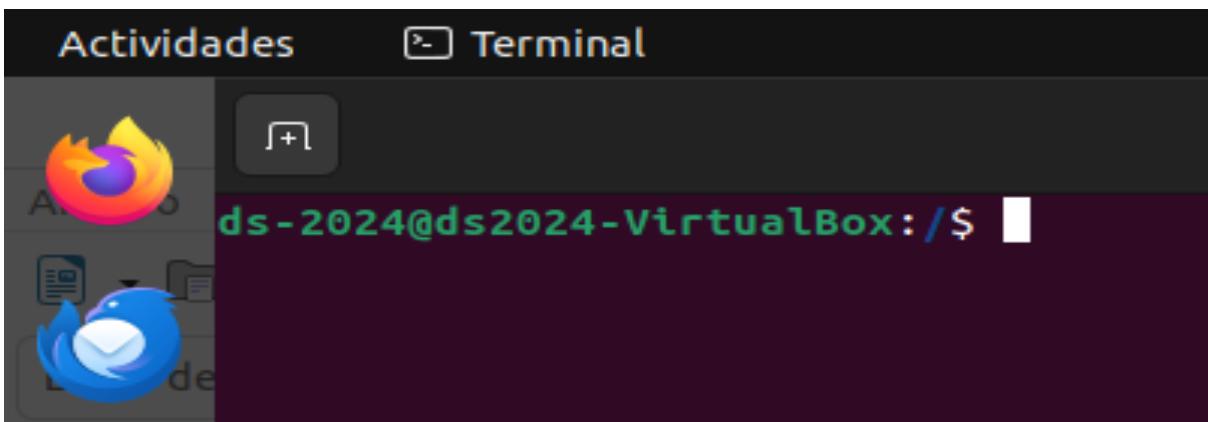


Algunas características generales son:

- 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.

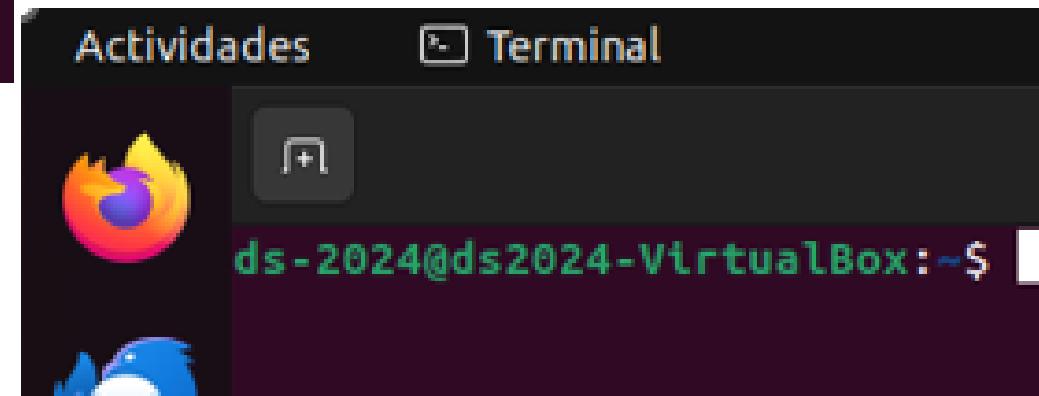
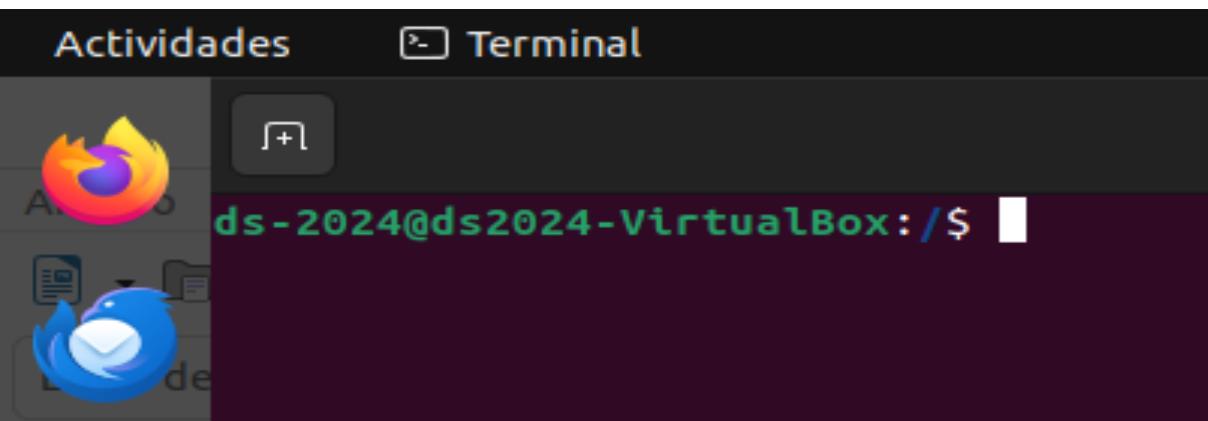
Algunas características generales son:

- 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

- * <^o 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...)
- * <^o Compartidos: Contiene archivos que se pueden encontrar en un ordenador y utilizarse en otro, o incluso compartirse entre usuarios.

Estructura

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

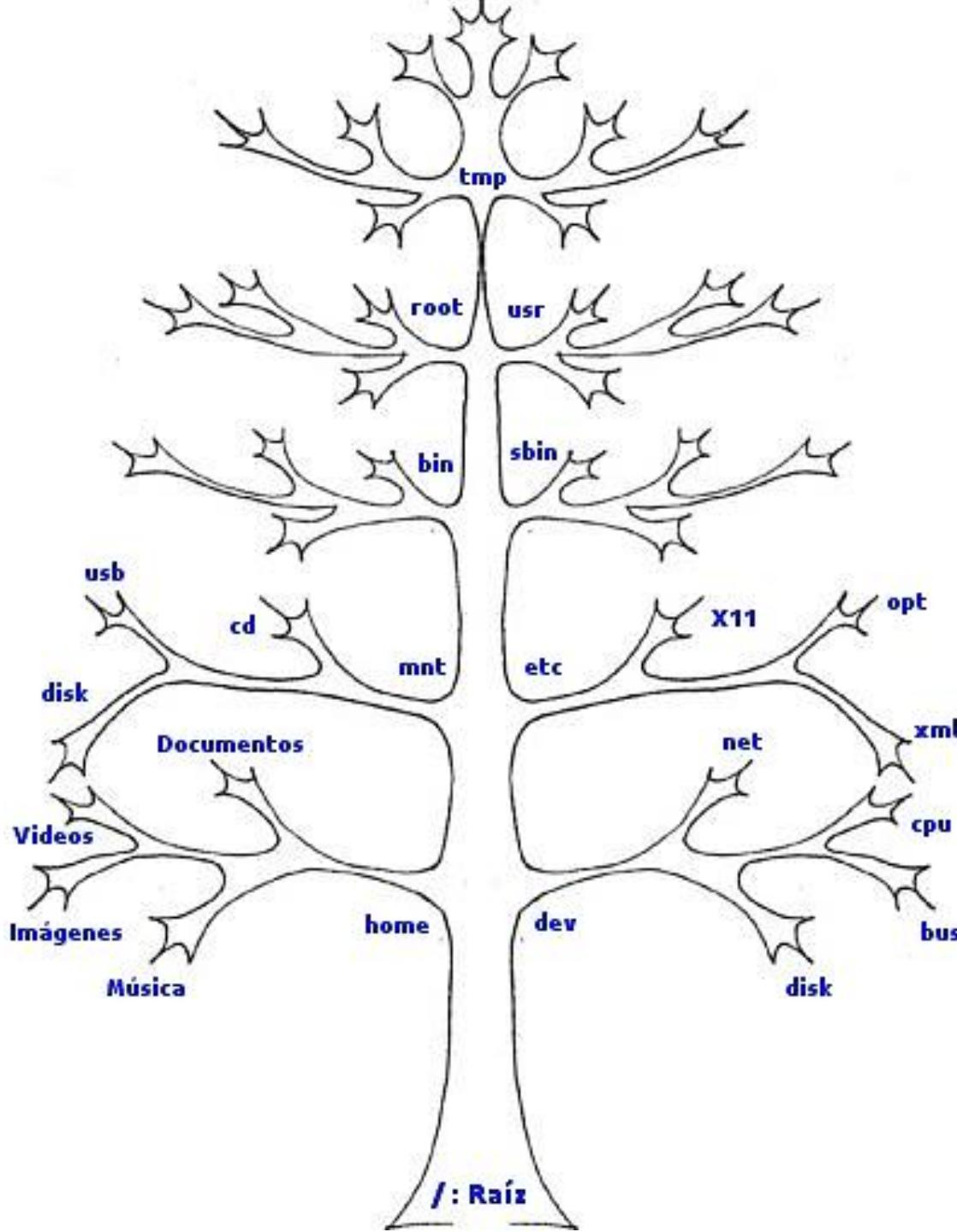
Estructura

- * 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

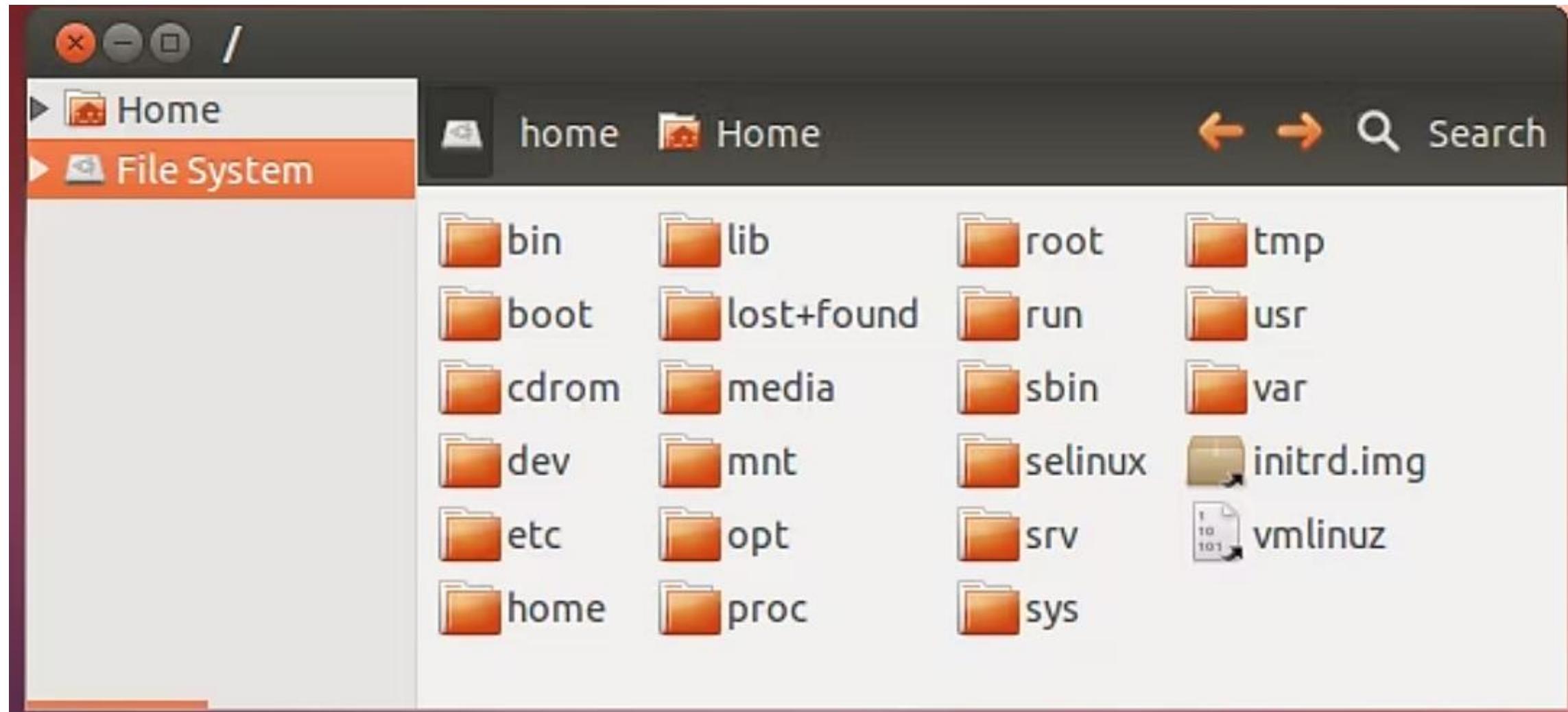
Directorio raíz de la jerarquía del sistema de ficheros
/

bin/	BINARIOS DE ORDENES DE USUARIO ESENCIALES
boot/	ARCHIVOS ESTÁTICOS DEL CARGADOR DE AUTOARRANQUE
dev/	ARCHIVOS DE DISPOSITIVOS
etc/	CONFIGURACION DEL SISTEMA DEL HOST
home/	DIRECTORIOS CASA DE LOS USUARIOS
lib/	BIBLIOTECAS COMPARTIDAS Y MODULOS DEL NUCLEO (KERNEL)
media/	PUNTO DE MONTAJE PARA MEDIOS DESMONTABLES
mnt/	PUNTO DE MONTAJE PARA SISTEMAS DE ARCHIVOS TEMPORALMENTE MONTADOS
opt/	PAQUETES DE APLICACIONES DE SOFTWARE COMPLEMENTARIAS
sbin/	BINARIOS DEL SISTEMA
SVr/	DATOS PARA SERVICIOS PROPORCIONADOS POR EL SISTEMA
tmp/	FICHEROS TEMPORALES
usr/	UTILIDADES Y APLICACIONES DE USUARIO(S)
var/	ARCHIVOS VARIABLES
root/	DIRECTORIOS CASA DEL USUARIO ADMINISTRADOR
proc/	ESTADO DE LOS PROCESOS E INFORMACION DEL NUCLEO

home/user1
home/user2

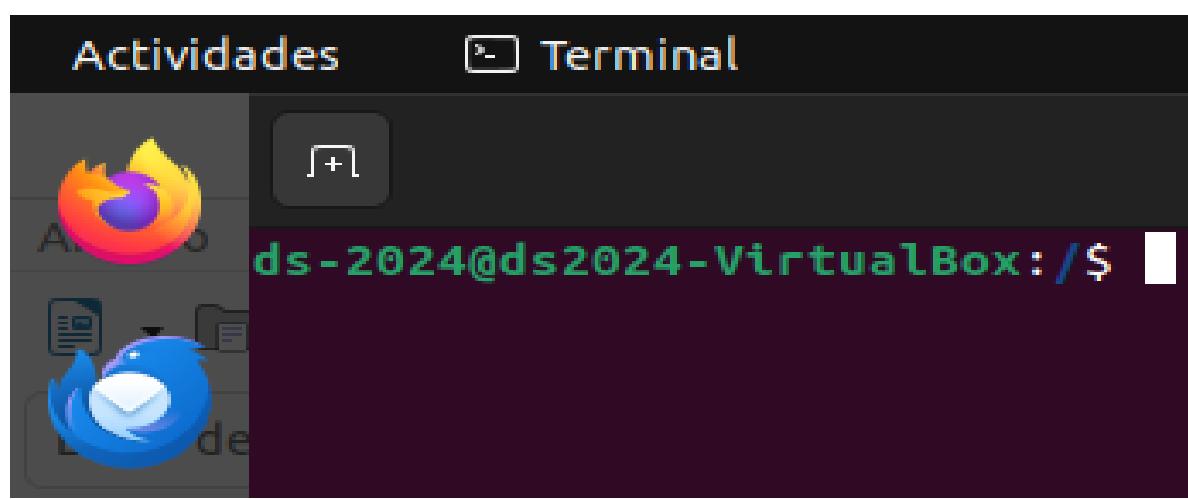
usr/bin/
usr/include/
usr/lib/
usr/local/bin/
usr/local/
usr/sbin/
usr/local/games/
usr/share/

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

Seguimos con las características generales: 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.

Seguimos con las 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

Seguimos con las características generales: 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.

Seguimos con las características generales: 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.

Seguimos con las características generales: 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 `ls --color -S -r --exclude='*'`

En este caso, se utilizan los siguientes modificadores:

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

-S: Ordena los archivos por tamaño.

-r: Ordena los archivos en orden inverso.

--exclude='*' : Excluye los archivos que comienzan con un punto ("."), que son archivos ocultos.

Actividad 4. Importancia de Linux y Shell para la Ciencia de Datos.

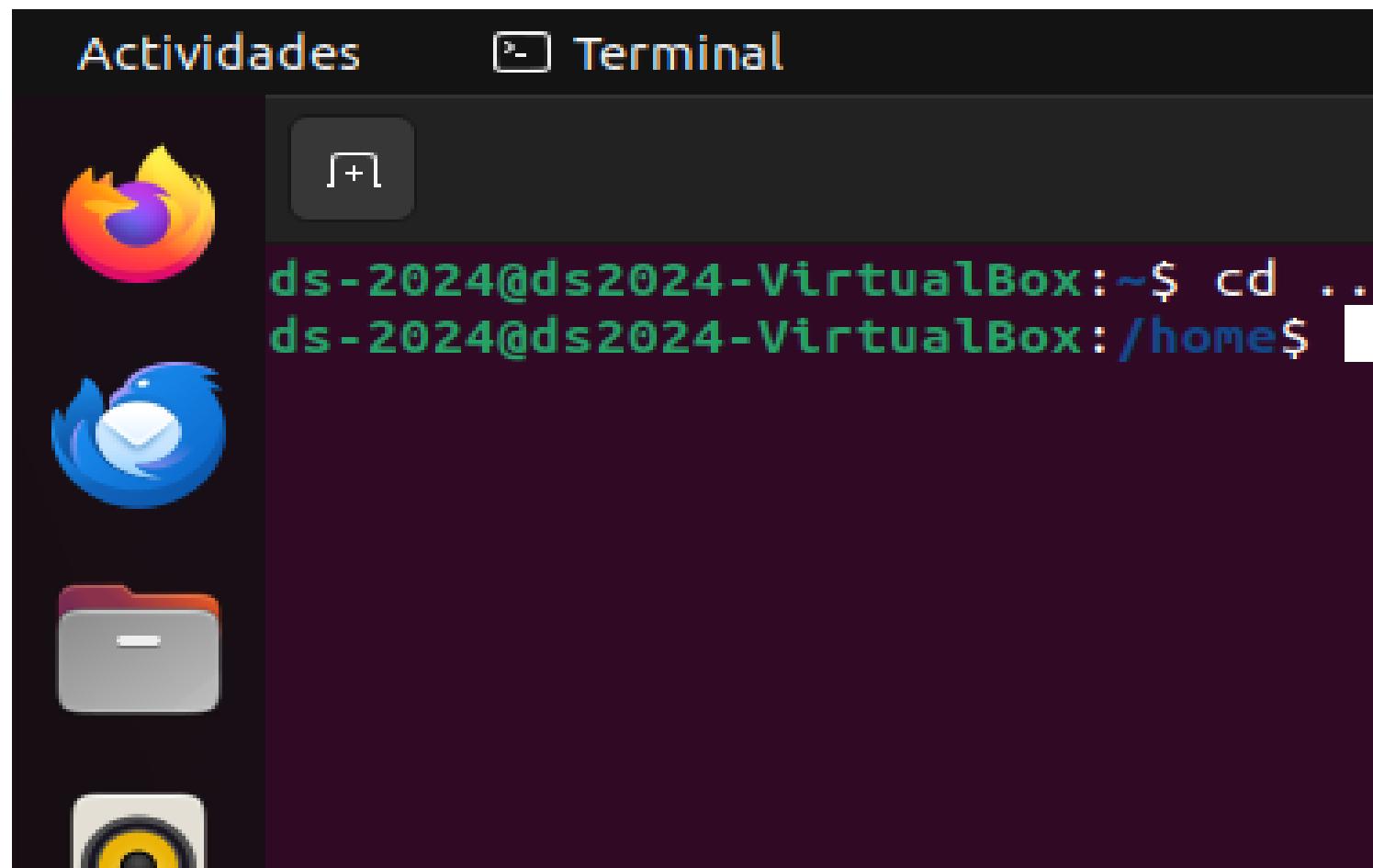
Preparar una presentación donde se explique la importancia y las ventajas que tiene el uso de Linux y la terminal (shell) de Linux para la ciencia de datos. Mencione ejemplos y un caso de uso.

Cantidad mínima de diapositivas: 20

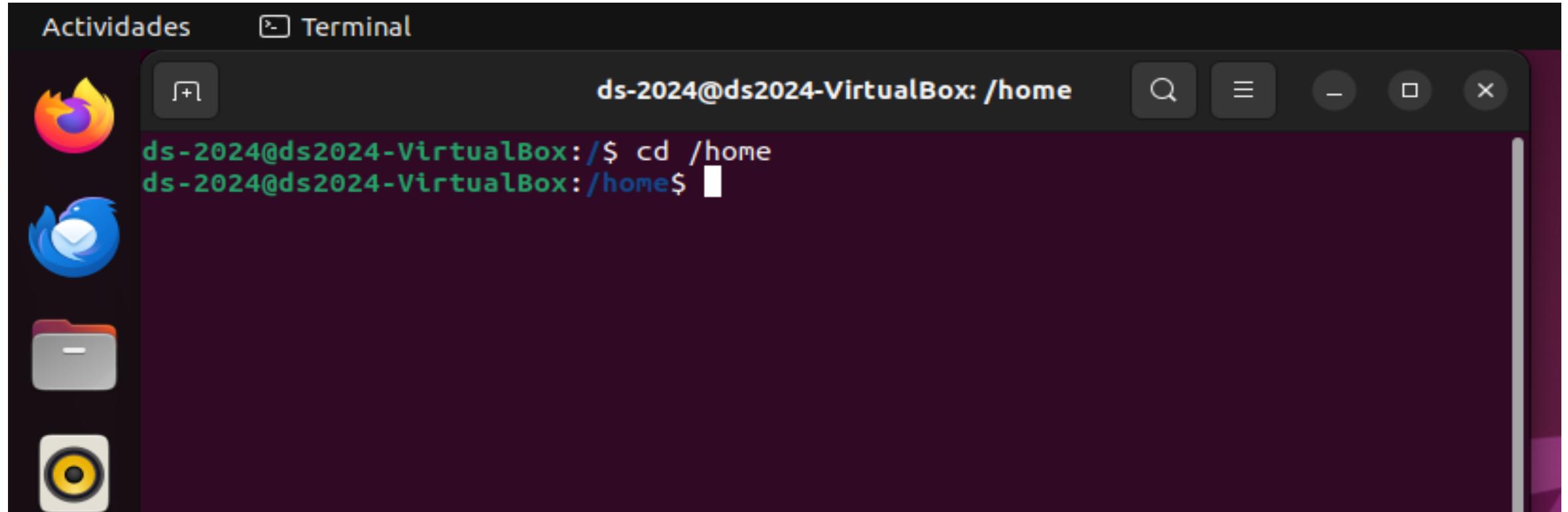
Piense que la presentación es para un público general sin nociones de ciencia de datos.

Comandos básicos de navegación por el sistema de archivos

Cambio de directorio: **cd ..**

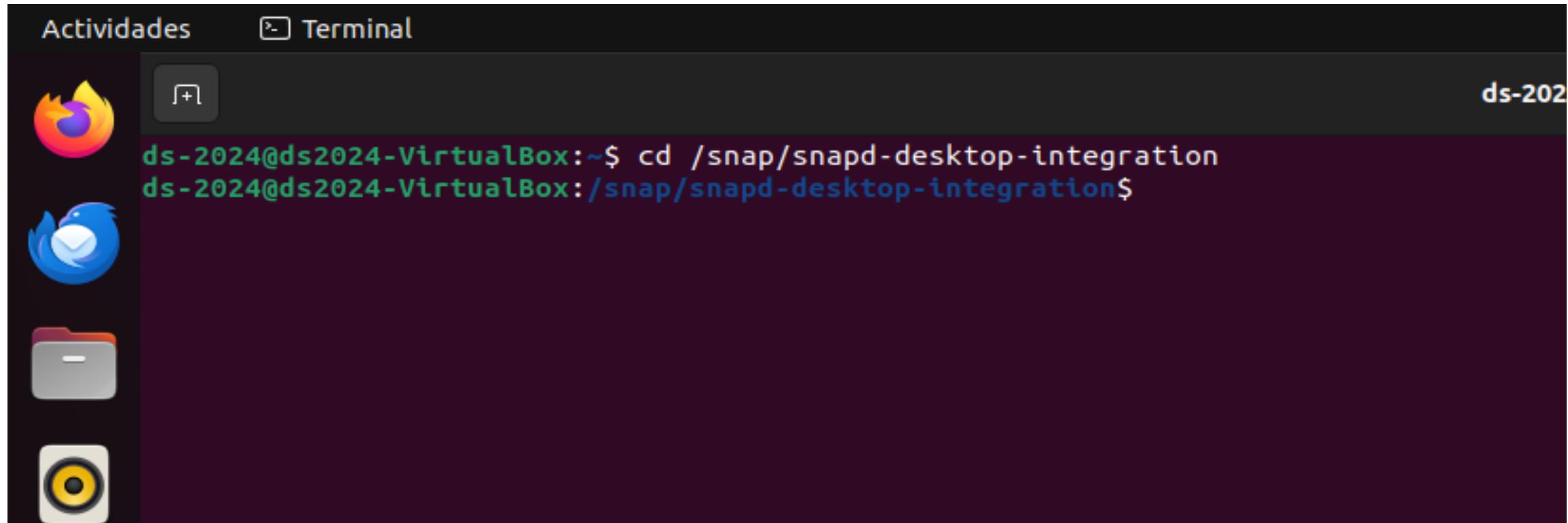


También se puede escribir cd con la ruta a llegar



También se puede escribir cd con la ruta a llegar.

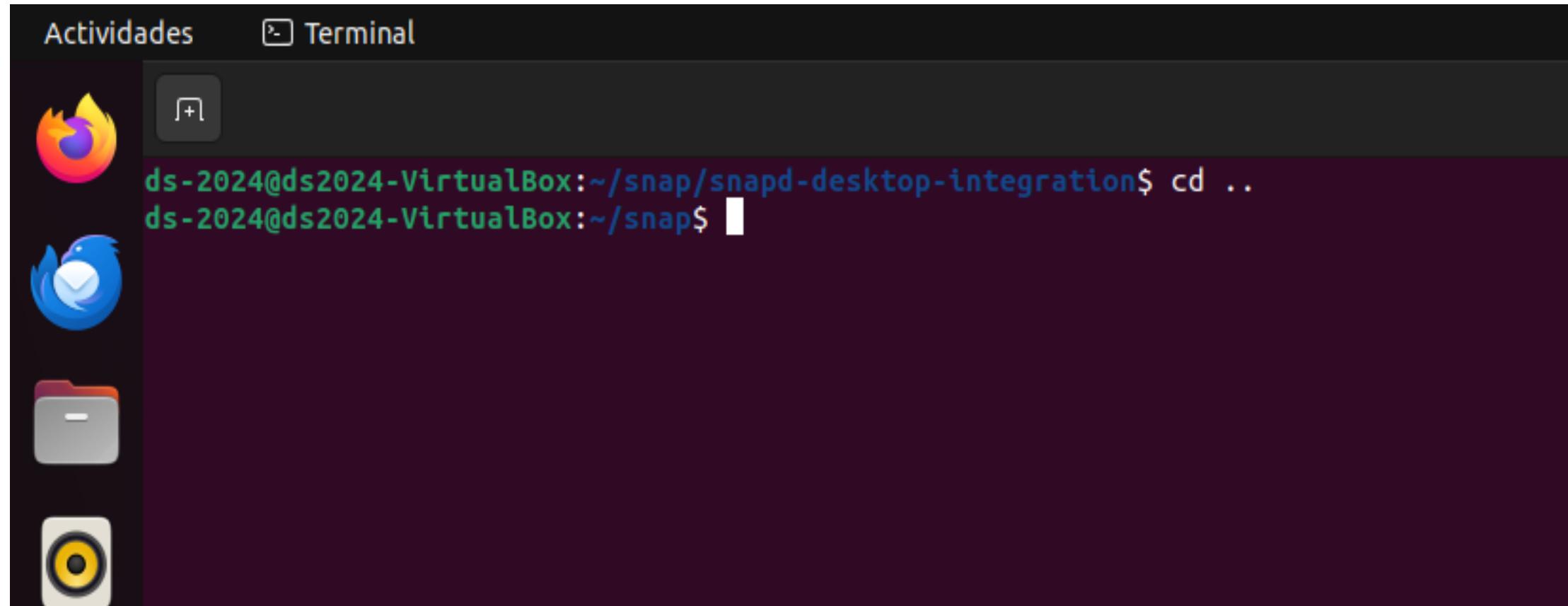
Otro ejemplo:



A screenshot of a Linux desktop environment. At the top, there is a dark header bar with the text "Actividades" and "Terminal". On the right side of the header, there is a user icon labeled "ds-202". Below the header, there is a dock area with several icons: a red Firefox icon, a blue Mail icon, a folder icon with a minus sign, and a speaker icon. The main window is a terminal window titled "Terminal". The terminal window has a dark background and contains the following text:
ds-2024@ds2024-VirtualBox:~\$ cd /snap/snapd-desktop-integration
ds-2024@ds2024-VirtualBox:/snap/snapd-desktop-integration\$

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 mail client (Thunderbird), a file manager (Nautilus), and a volume control (Pulseaudio). Above the dock, a header bar shows "Actividades" and "Terminal". The terminal window is open and displays a command-line session:

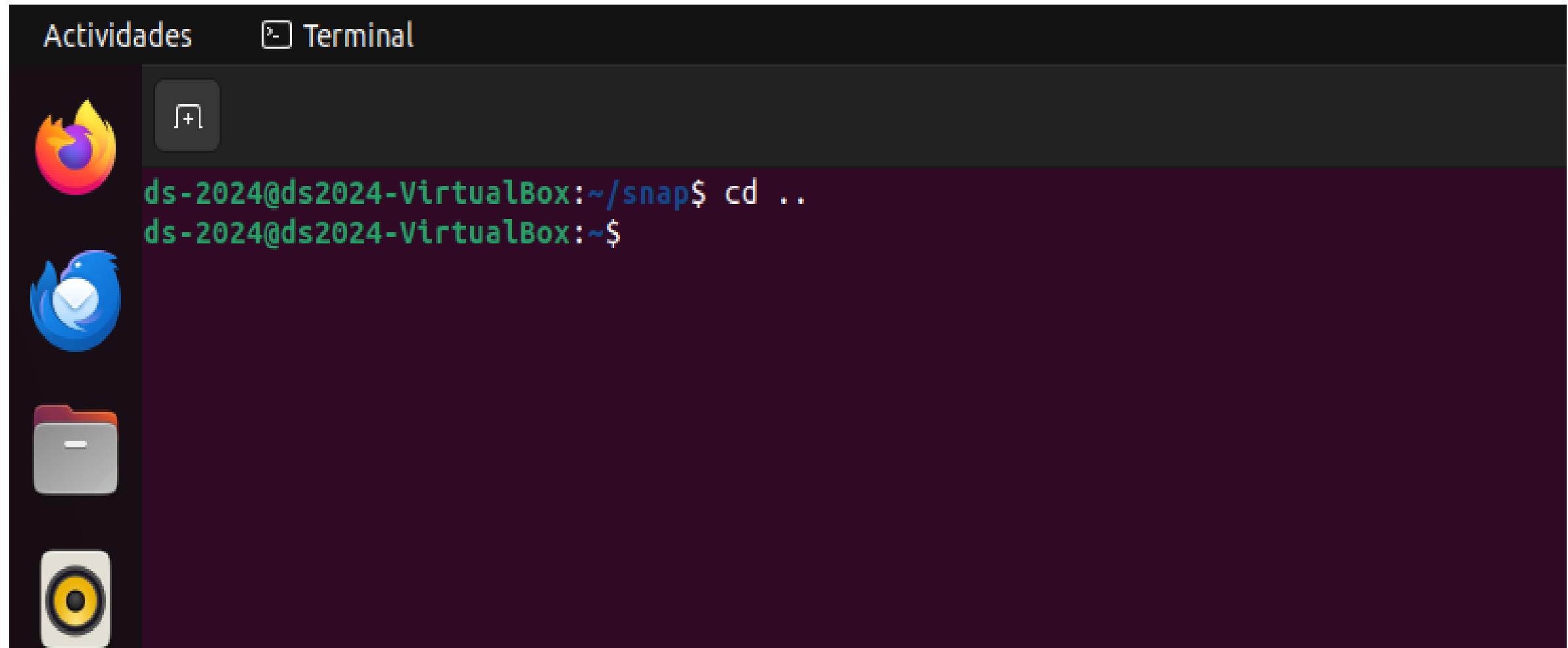
```
ds-2024@ds2024-VirtualBox:~/snap/snapd-desktop-integration$ cd ..  
ds-2024@ds2024-VirtualBox:~/snap$
```

The terminal has a dark theme with light-colored text. The cursor is visible at the end of the second line of text.

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

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

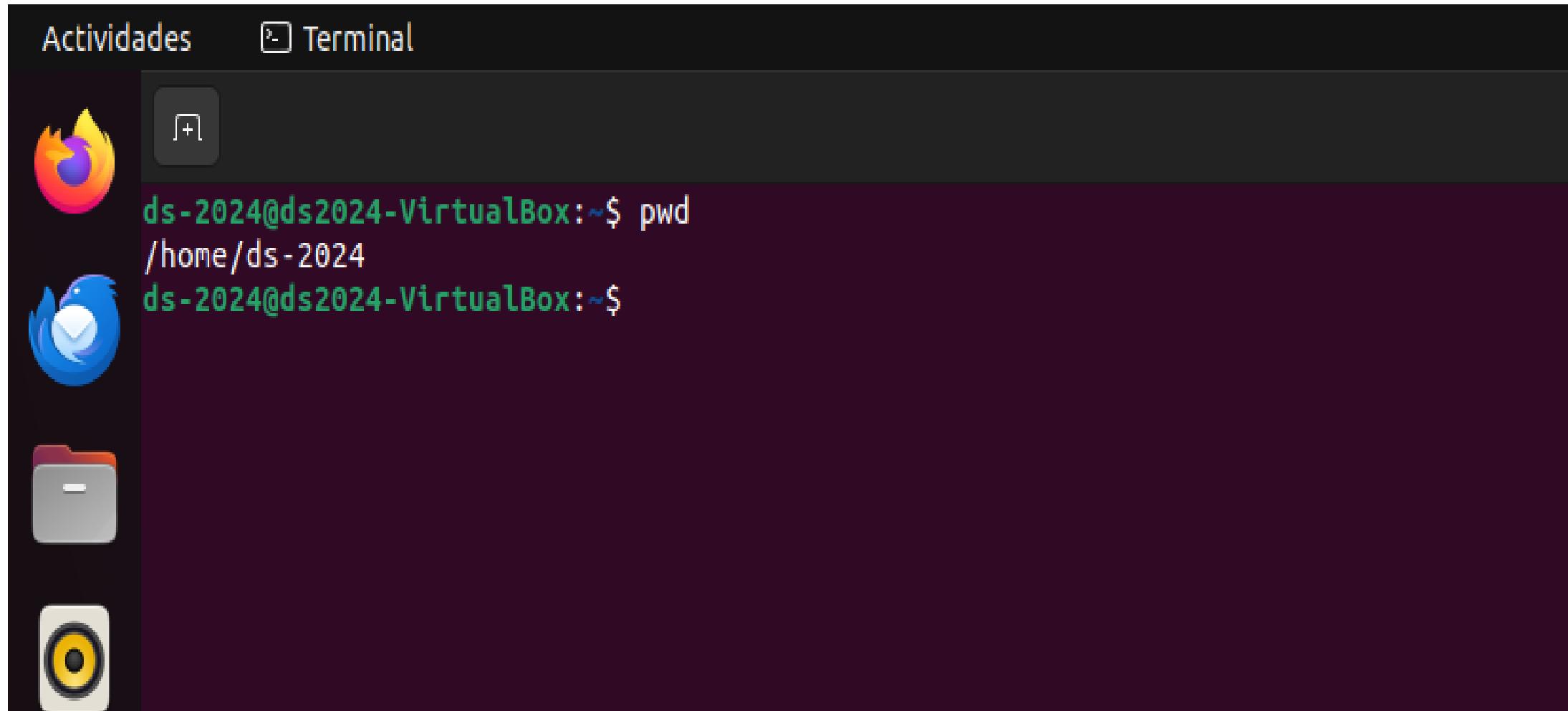
Actividades Terminal 3 de abr 08:29

```
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

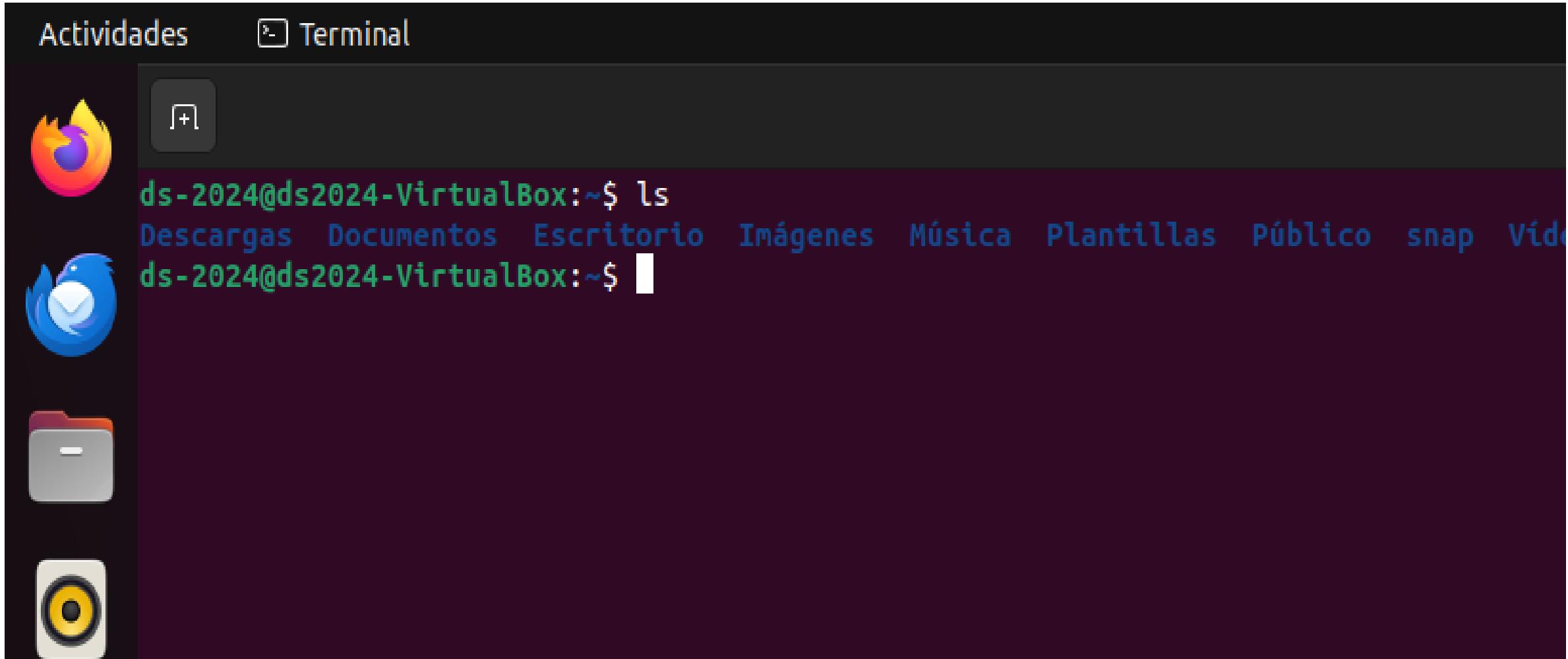


The screenshot shows a Linux desktop interface with a dark theme. At the top, there is a dock with several icons: a red and orange flame (Firefox), a blue bird (Thunderbird), a folder (File Manager), and a speaker (Speaker). To the right of the dock is a terminal window titled "Terminal". The terminal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

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

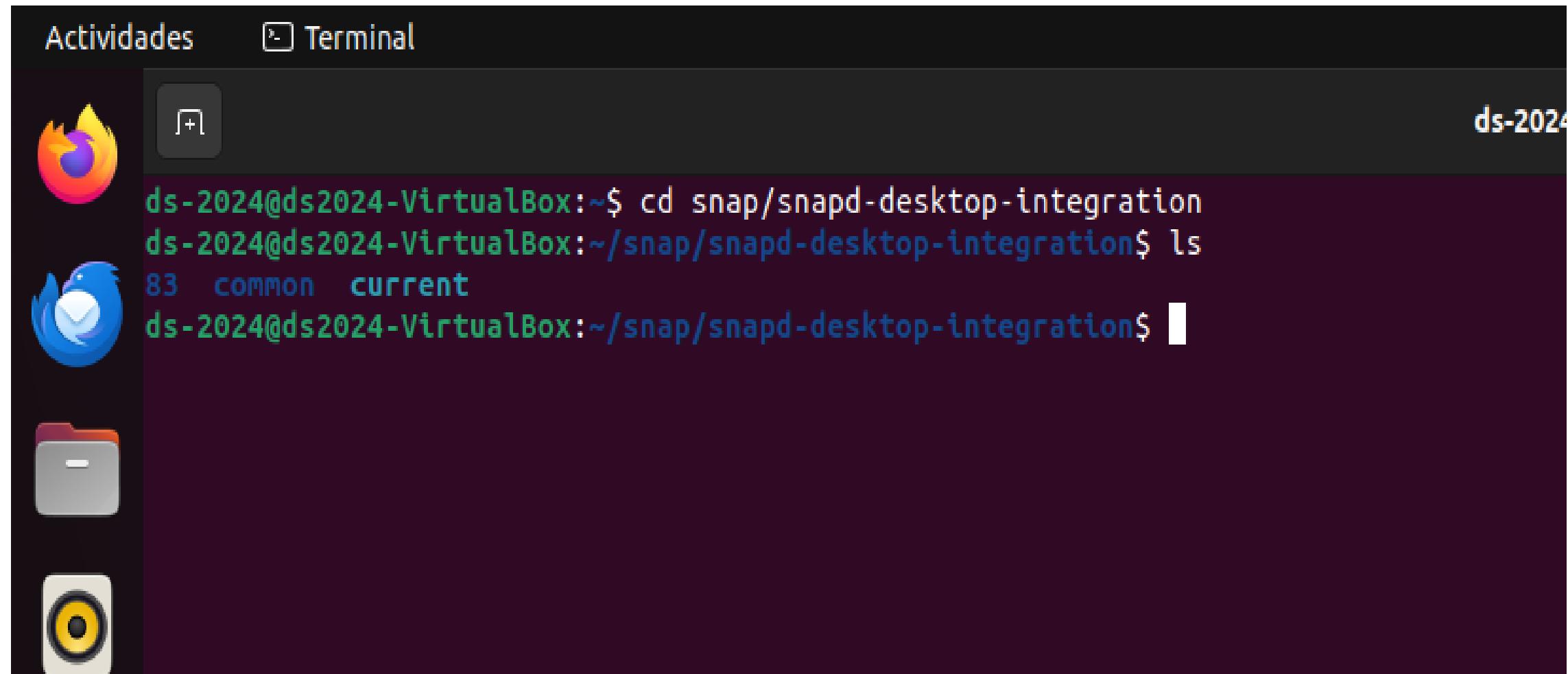
Listado del contenido de un directorio : ls

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. On the left, there's a dock with icons for a browser (Firefox), a file manager (Nautilus), and a音量 (volume) control. The main area shows a dark-themed desktop with a terminal window open. The terminal window has a title bar with "Actividades" and "Terminal". The terminal itself displays the command "ls" run from the home directory of a user named "ds-2024" on a VirtualBox machine. The output of the command is a list of directories: "Descargas", "Documentos", "Escritorio", "Imágenes", "Música", "Plantillas", "Público", "snap", and "Vídeos".

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

Otro ejemplo:

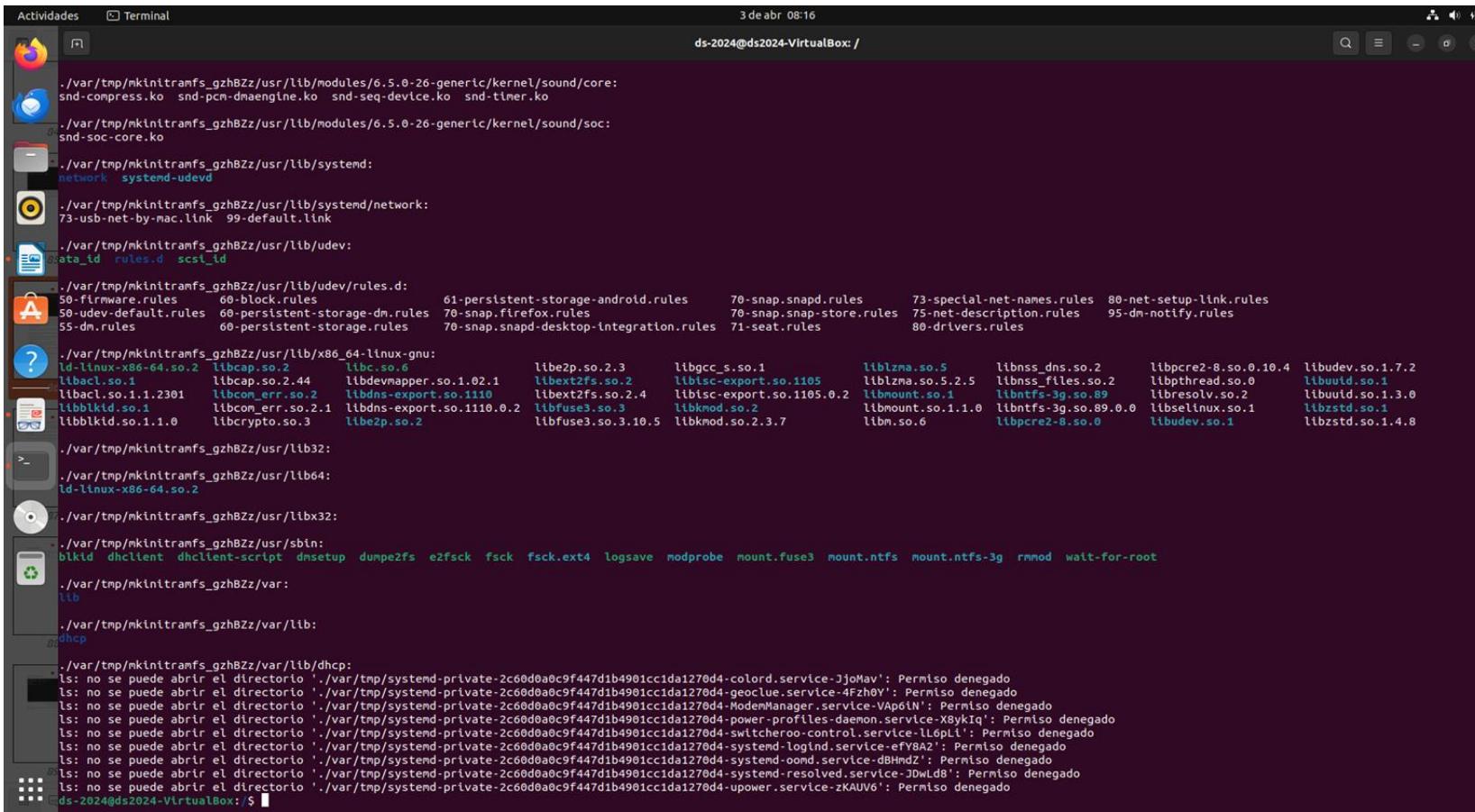


The screenshot shows a dark-themed desktop environment with a dock on the left containing icons for a browser (Firefox), file manager (Nautilus), and system settings (Speaker). At the top, there is a header bar with 'Actividades' and 'Terminal' tabs, and a search bar with a magnifying glass icon. On the right side of the header, the user's name 'ds-2024' is displayed. The main area is a terminal window with the following text:

```
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$
```

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

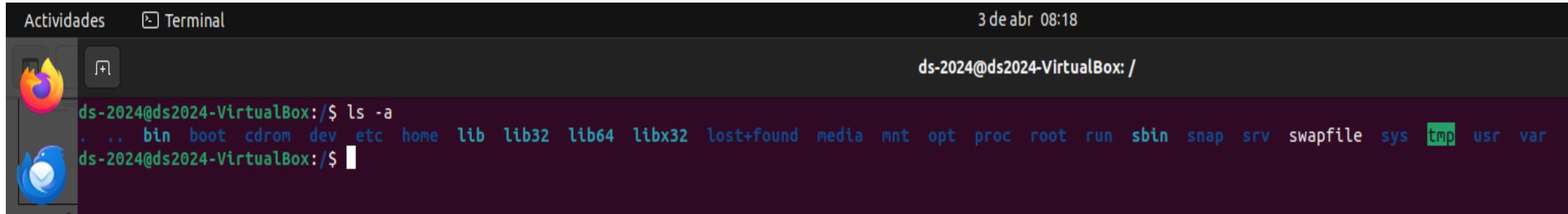
ls -R lista recursivamente los subdirectorios encontrados. Es decir, muestra los ficheros y directorios del actual, así como subdirectorios y su contenido:



The screenshot shows a terminal window titled "Terminal" with the command "ls -R" running. The output lists files and directories from the current directory down to subdirectories. The terminal window has a dark background with light-colored text. The top bar shows the date and time as "3 de abr 08:16" and the host name as "ds-2024@ds2024-VirtualBox: /". The bottom of the terminal shows the command prompt "ds-2024@ds2024-VirtualBox: \$".

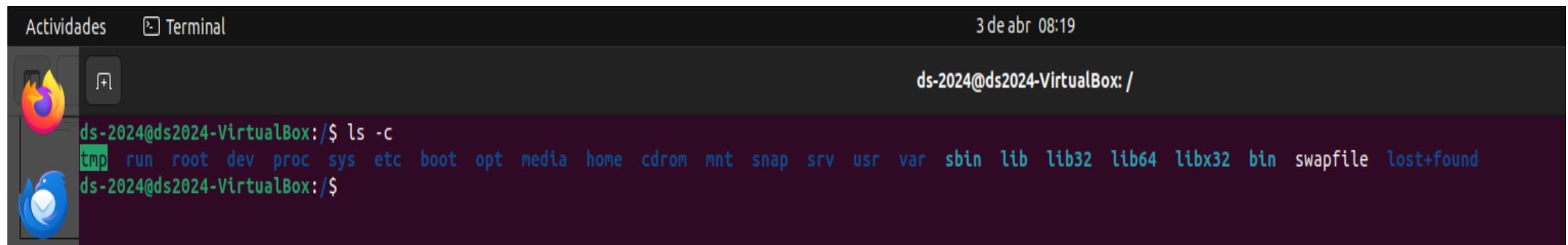
```
Actividades Terminal 3 de abr 08:16
ds-2024@ds2024-VirtualBox: /
./var/tmp/mkinitramfs_gzhBZz/usr/lib/modules/6.5.0-26-generic/kernel/sound/core:
snd-compress.ko snd-pcm-dnaengine.ko snd-seq-device.ko snd-timer.ko
./var/tmp/mkinitramfs_gzhBZz/usr/lib/modules/6.5.0-26-generic/kernel/sound/soc:
snd-soc-core.ko
./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:
ata_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 libbcap.so.2 libcc.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 liblzlma.so.5.2.5 libnss_files.so.2 libpthread.so.0 libubtld.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 libntfs-3g.so.89 libresolv.so.2 libubtld.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 libntfs-3g.so.89.0.0 libselinux.so.1 libzstd.so.1
libblkid.so.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 dnsetup 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-2c60d0a0c9f447db4901cc1da1270d4-.color.service-JjoMav': Permiso denegado
ls: no se puede abrir el directorio './var/tmp/systemd-private-2c60d0a0c9f447db4901cc1da1270d4-.geoclue.service-4Fzh0Y': Permiso denegado
ls: no se puede abrir el directorio './var/tmp/systemd-private-2c60d0a0c9f447db4901cc1da1270d4-.ModemManager.service-Vap0In': Permiso denegado
ls: no se puede abrir el directorio './var/tmp/systemd-private-2c60d0a0c9f447db4901cc1da1270d4-.power-profiles-daemon.service-XBykIq': Permiso denegado
ls: no se puede abrir el directorio './var/tmp/systemd-private-2c60d0a0c9f447db4901cc1da1270d4-.switcheroo-control.service-Ll6pl1': Permiso denegado
ls: no se puede abrir el directorio './var/tmp/systemd-private-2c60d0a0c9f447db4901cc1da1270d4-.systemd-logind.service-efYBA2': Permiso denegado
ls: no se puede abrir el directorio './var/tmp/systemd-private-2c60d0a0c9f447db4901cc1da1270d4-.systemd-logind.service-efYBA2': Permiso denegado
ls: no se puede abrir el directorio './var/tmp/systemd-private-2c60d0a0c9f447db4901cc1da1270d4-.systemd-resolved.service-zDwId8': Permiso denegado
ls: no se puede abrir el directorio './var/tmp/systemd-private-2c60d0a0c9f447db4901cc1da1270d4-.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.



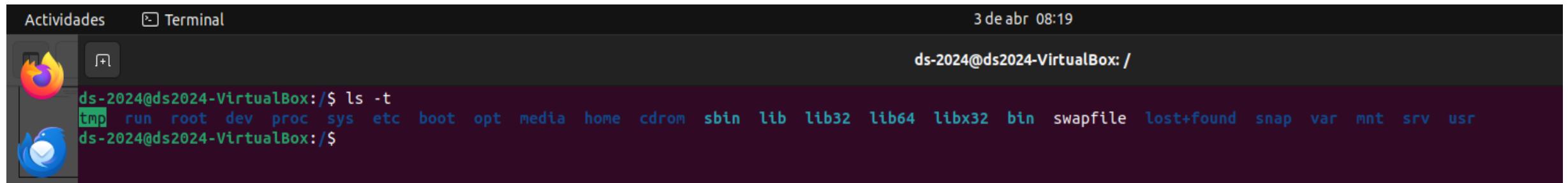
A screenshot of a Linux desktop environment. At the top, there's a dark header bar with the word "Actividades" and a "Terminal" icon. On the right side of the header, it shows the date "3 de abr 08:18". Below the header is a terminal window with a dark background and light-colored text. The terminal window has a title bar that says "ds-2024@ds2024-VirtualBox: /". Inside the terminal, the command "ls -a" is run, displaying a long list of directory entries including ". . ." and many files 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", and ".var".

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 date "3 de abr 08:19" at the top right. The terminal title is "ds-2024@ds2024-VirtualBox: /". The command "ls -c" is run, resulting in a list of directory entries that is significantly shorter than the one from "ls -a". The entries include ".tmp", ".run", ".root", ".dev", ".proc", ".sys", ".etc", ".boot", ".opt", ".media", ".home", ".cdrom", ".mnt", ".snap", ".srv", ".var", ".sbin", ".lib", ".lib32", ".lib64", ".libx32", ".bin", ".swapfile", and ".lost+found".

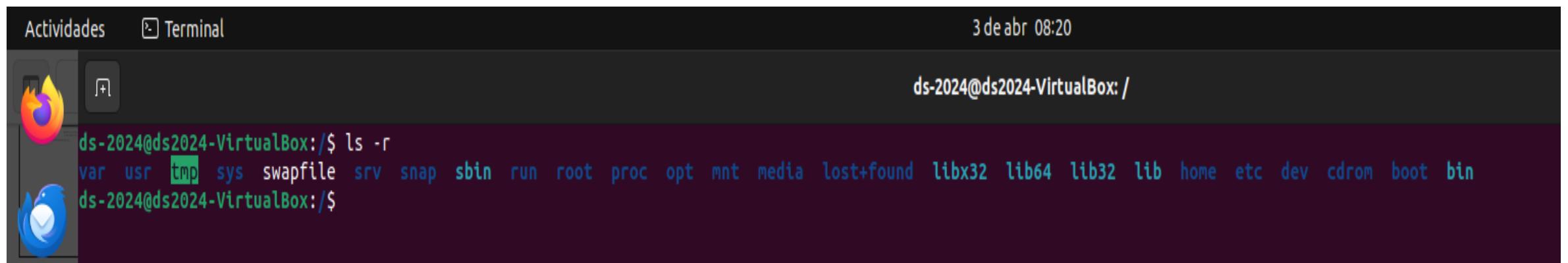
ls -t muestra el contenido del directorio ordenado por día y hora de modificación.



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 "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 including tmp, run, root, dev, proc, sys, etc, sorted by modification time. The terminal ends with "ds-2024@ds2024-VirtualBox:/\$".

```
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:/$
```

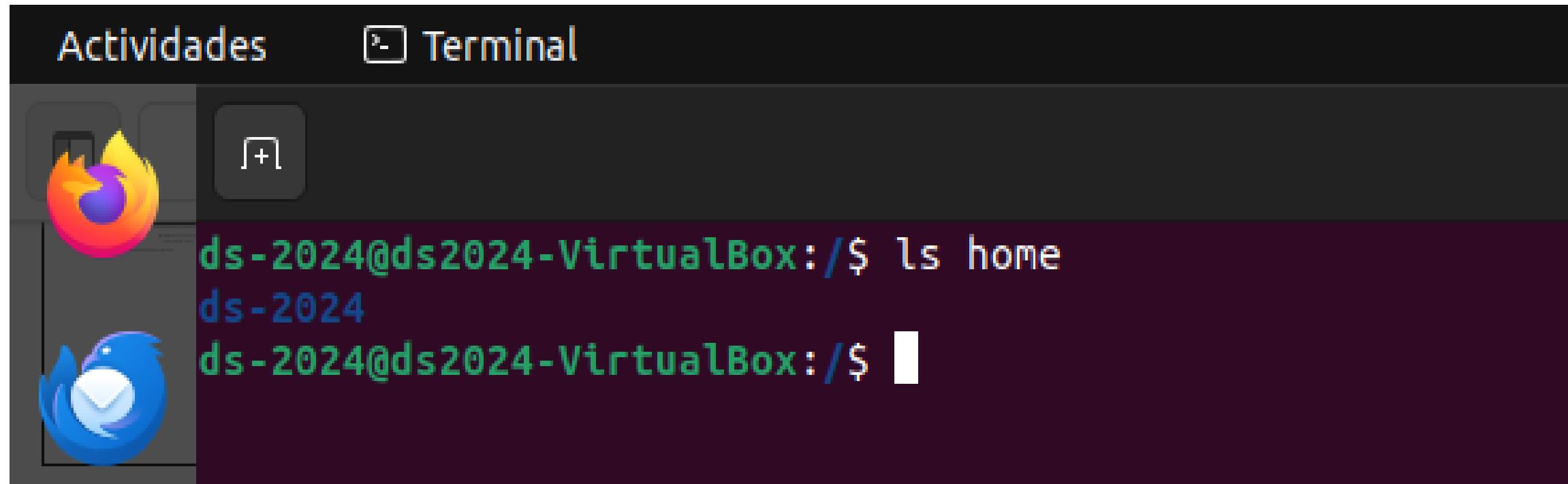
ls -r muestra el contenido del directorio y lo ordena alfabéticamente en orden inverso.



A screenshot of a Linux desktop environment. 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 "3 de abr 08:20". 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 -r" is run, displaying the same list of directory contents as the previous terminal, but in reverse alphabetical order. The terminal ends with "ds-2024@ds2024-VirtualBox:/\$".

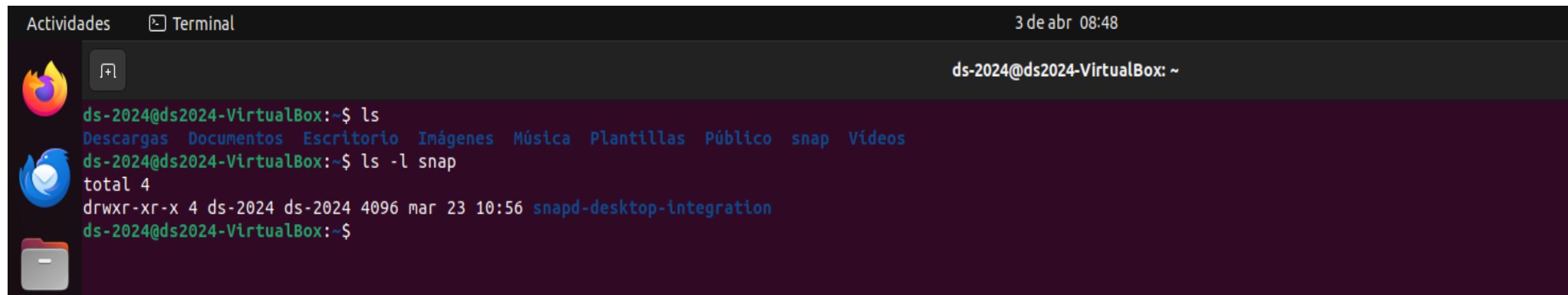
```
ds-2024@ds2024-VirtualBox:/$ ls -r
var usr tmp sys swapfile srv snap sbin run root proc opt mnt media lost+found libx32 lib64 lib32 lib home etc dev cdrom boot bin
ds-2024@ds2024-VirtualBox:/$
```

ls subdir muestra el contenido del subdirectorio “subdir”.

A screenshot of a dark-themed Ubuntu desktop environment. At the top, there's a dock with icons for the Dash, Home, and Terminal. The Terminal window is open and shows the command "ls home" being run in a terminal session. The output of the command, "ds-2024", is visible. The desktop background is a dark blue gradient.

```
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.

A screenshot of a Linux desktop environment, likely Ubuntu, showing a terminal window. The terminal window has a dark background and contains the following command-line session:

```
Actividades Terminal 3 de abr 08:48
ds-2024@ds2024-VirtualBox: ~
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:~$
```

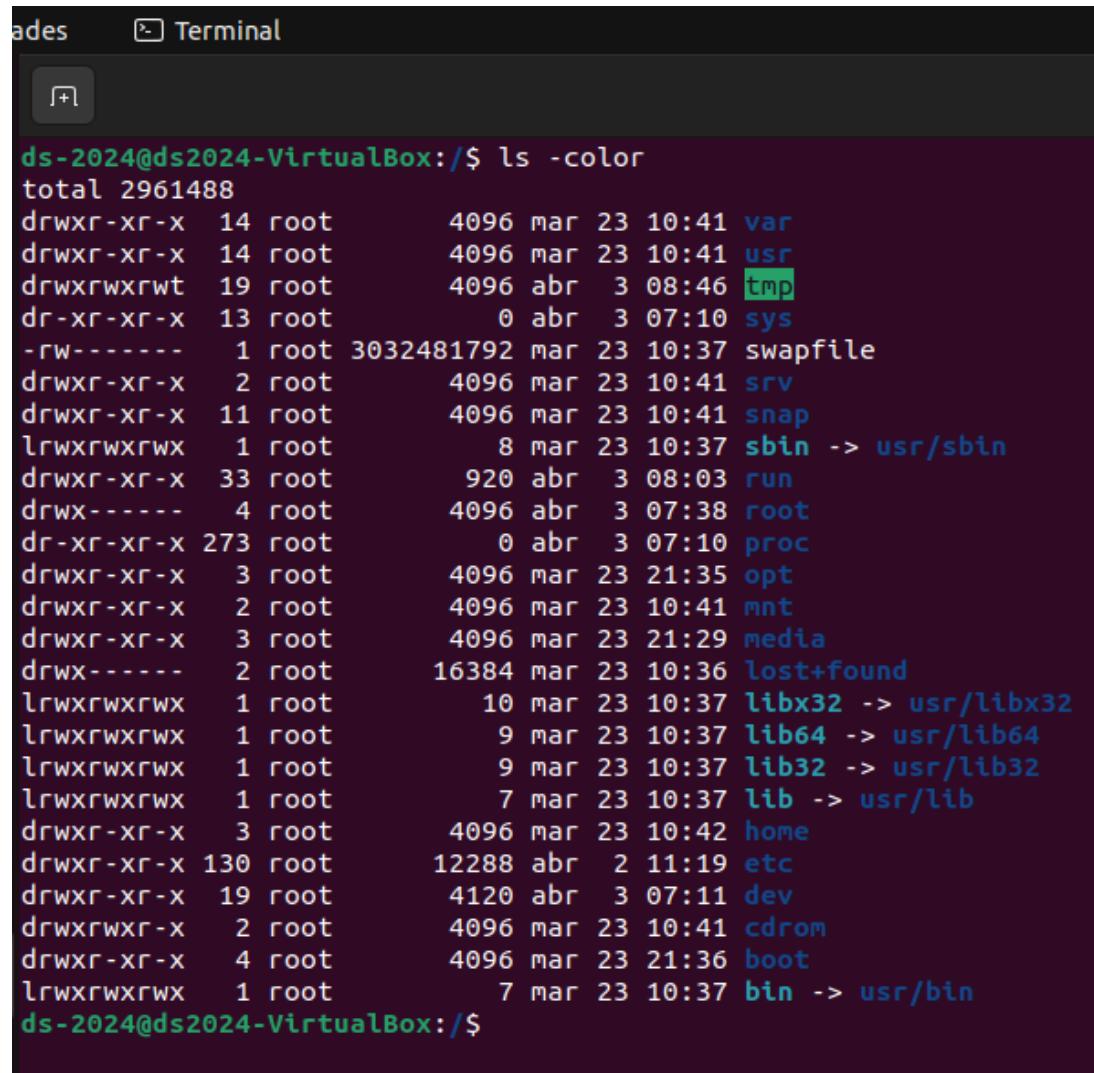
The desktop interface includes a dock with icons for the Dash, Home, and other applications like the Dash, Terminal, and a file manager.

Otro ejemplo con: ls –l linux

Actividades Terminal 3 de abr 08:54

```
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 lib
ds-2024@ds2024-VirtualBox:/lib$ ls
apg           debug          gold-ld
apparmor      dpkg            groff
apt           emacsclient-common grub
aspell         environment.d  grub-legacy
bfd-plugins   evolution-data-server gvfs
binfmt.d      file            hdparm
bluetooth     firewalld       init
brltty        firmware        initramfs-tools
cnf-update-db gcc             ispell
command-not-found girepository-1.0 kernel
compat-ld     gnome-session    klibc
console-setup gnome-settings-daemon-3.0 klibc-K8e6D0mVI9JpyGMLR7qNe5iZeBk.so NetworkManager
cpp           gnome-settings-daemon-42 libreoffice
crda          gnome-shell      linux
cups          gnupg           linux-boot-probes
dbus-1.0      gnupg2          linux-sound-base
ds-2024@ds2024-VirtualBox:/lib$ ls -l linux
total 4
drwxr-xr-x 2 root root 4096 mar 23 10:51 triggers
ds-2024@ds2024-VirtualBox:/lib$
```

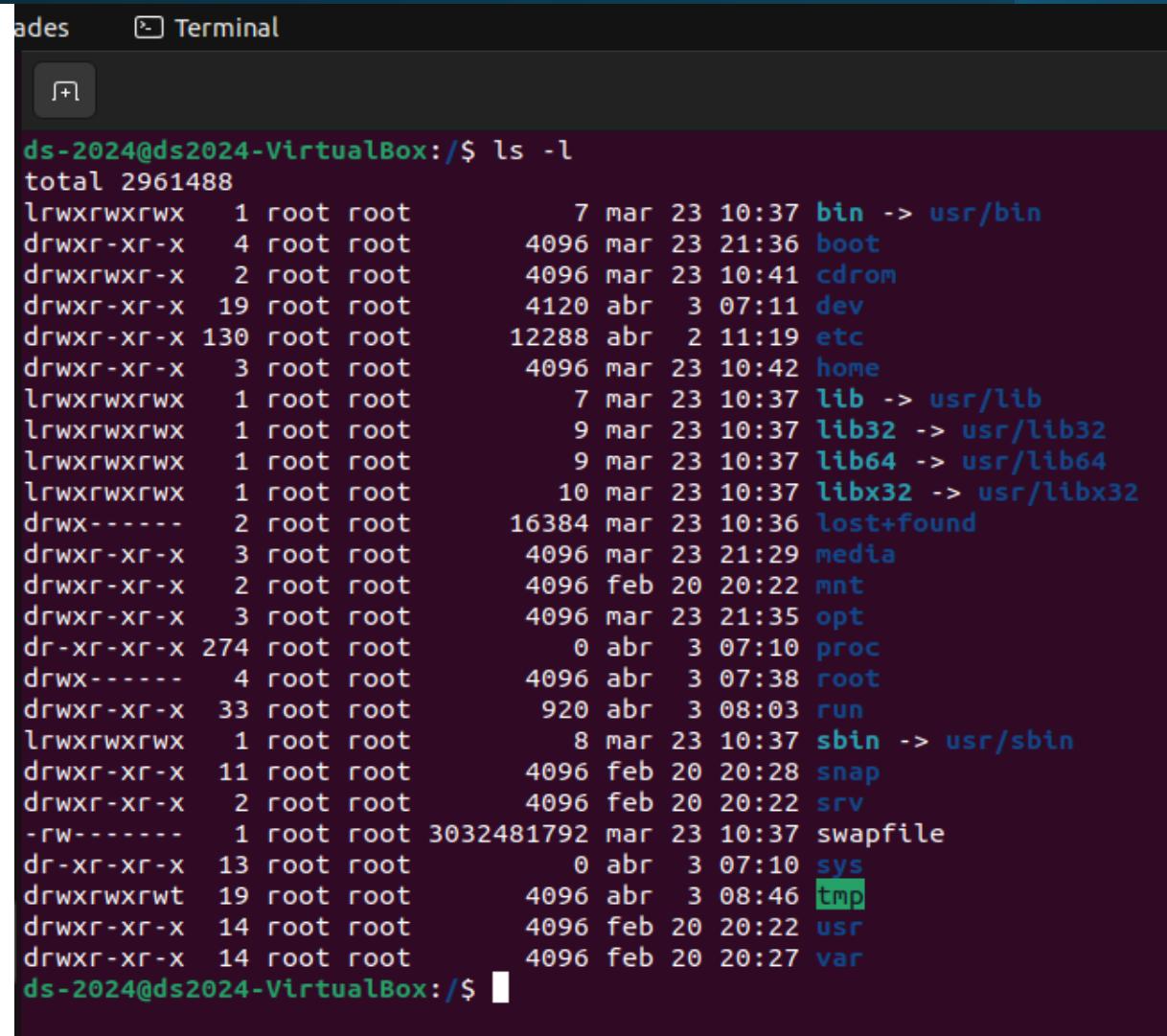
ls -color muestra el contenido del directorio coloreado: verde para los ejecutables, azul para las carpetas, fucsia para las imágenes, rojo para los comprimidos, etc.



The screenshot shows a terminal window titled "Terminal" with the command "ls -color" run in it. The output lists numerous system directories and files, each colored according to its type: green for executables, blue for directories, magenta for symbolic links, and black for regular files. The terminal is running on a Linux system, as indicated by the prompt "ds-2024@ds2024-VirtualBox:~\$".

```
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
```

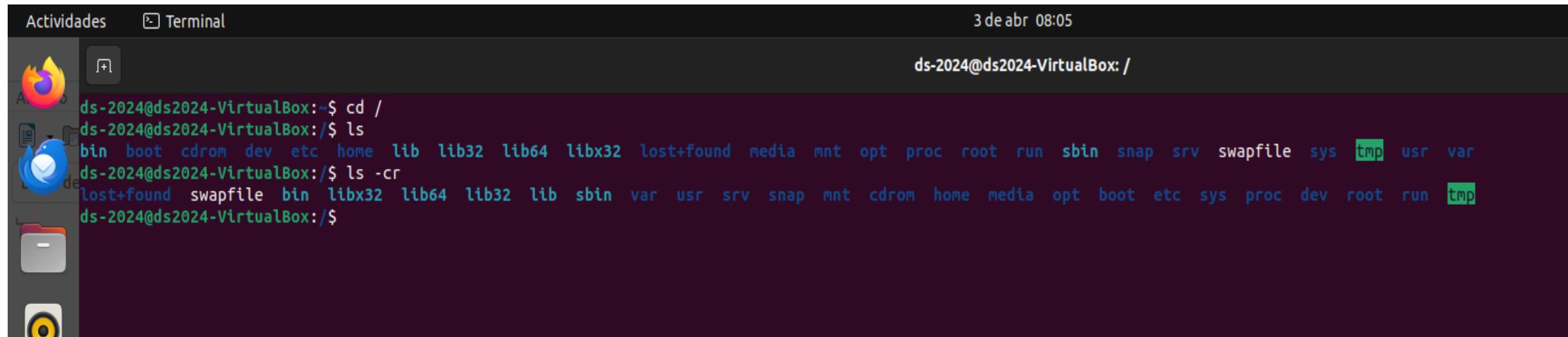
ls -l muestra toda la información de cada archivo, incluyendo permisos, tipo de archivo, tamaño y fecha de creación o del último cambio introducido, etc.



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. Many entries are symbolic links pointing to other locations like "/usr/bin", "/usr/lib", and "/usr/lib32".

```
ades Terminal
+
ds-2024@ds2024-VirtualBox:/$ ls -l
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
ds-2024@ds2024-VirtualBox:/$
```

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 content is as follows:

```
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. It includes standard Linux icons for file operations (trash, folder, etc.) on the left side.

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.

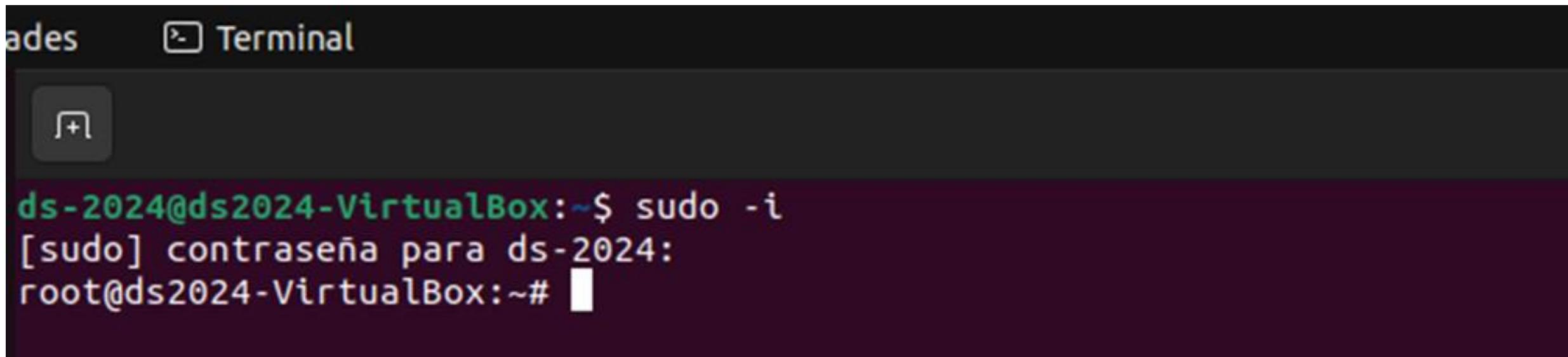
El carácter “?” se utiliza para representar cualquier carácter una única vez.

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.

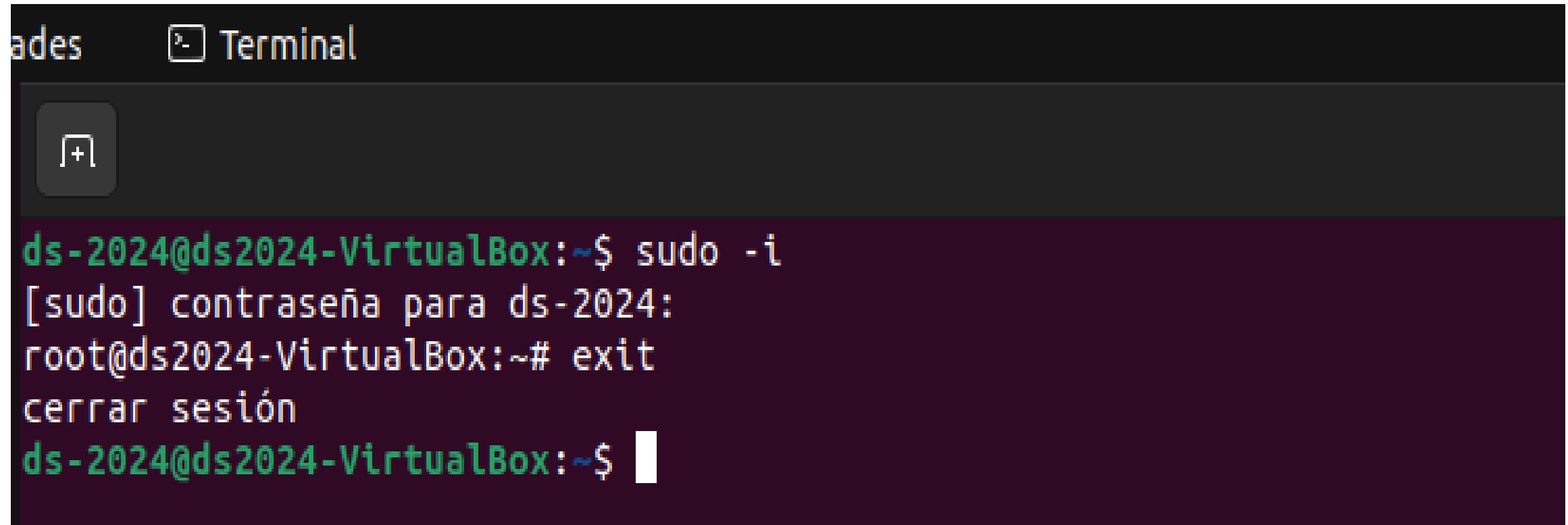
Para cambiar al modo superusuario: **sudo -i**.
Luego introducir la contraseña de root



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:~# █
```

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:

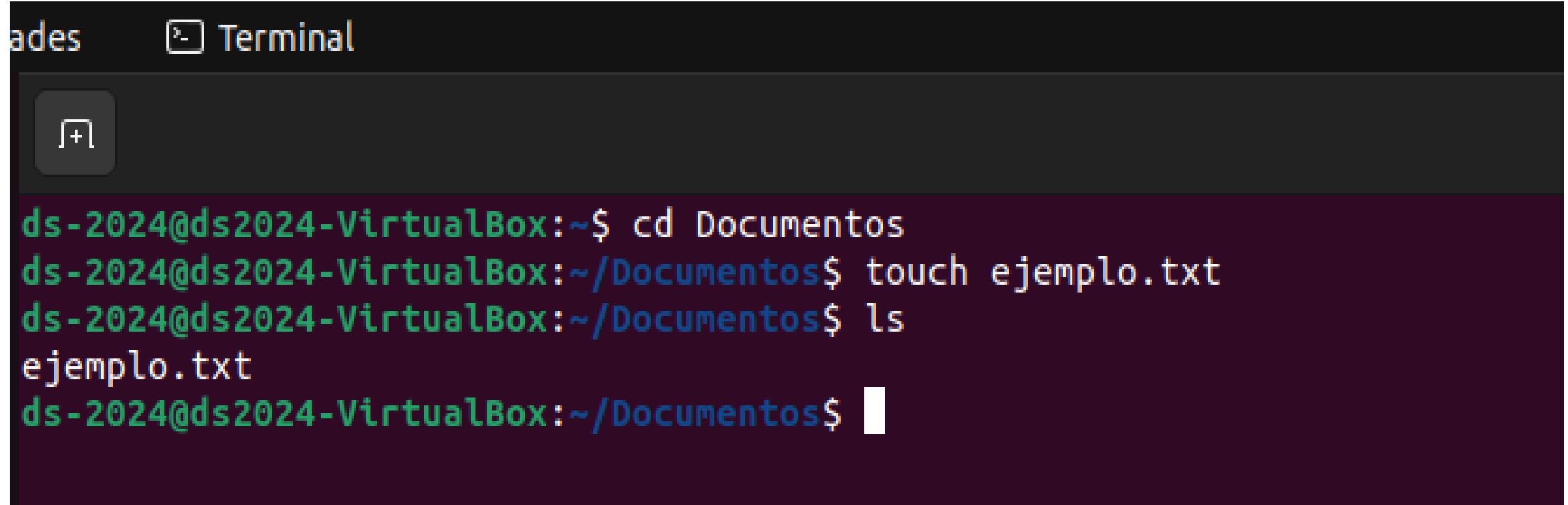
```
ades Terminal
[+]
ds-2024@ds2024-VirtualBox:~$ sudo -i
[sudo] contraseña para ds-2024:
root@ds2024-VirtualBox:~# exit
cerrar sesión
ds-2024@ds2024-VirtualBox:~$
```

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. At the top, there is a tab labeled "Terminal". The terminal 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$
```

The terminal window also features a small icon with a plus sign (+) in the top-left corner, likely for opening a new terminal tab.

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

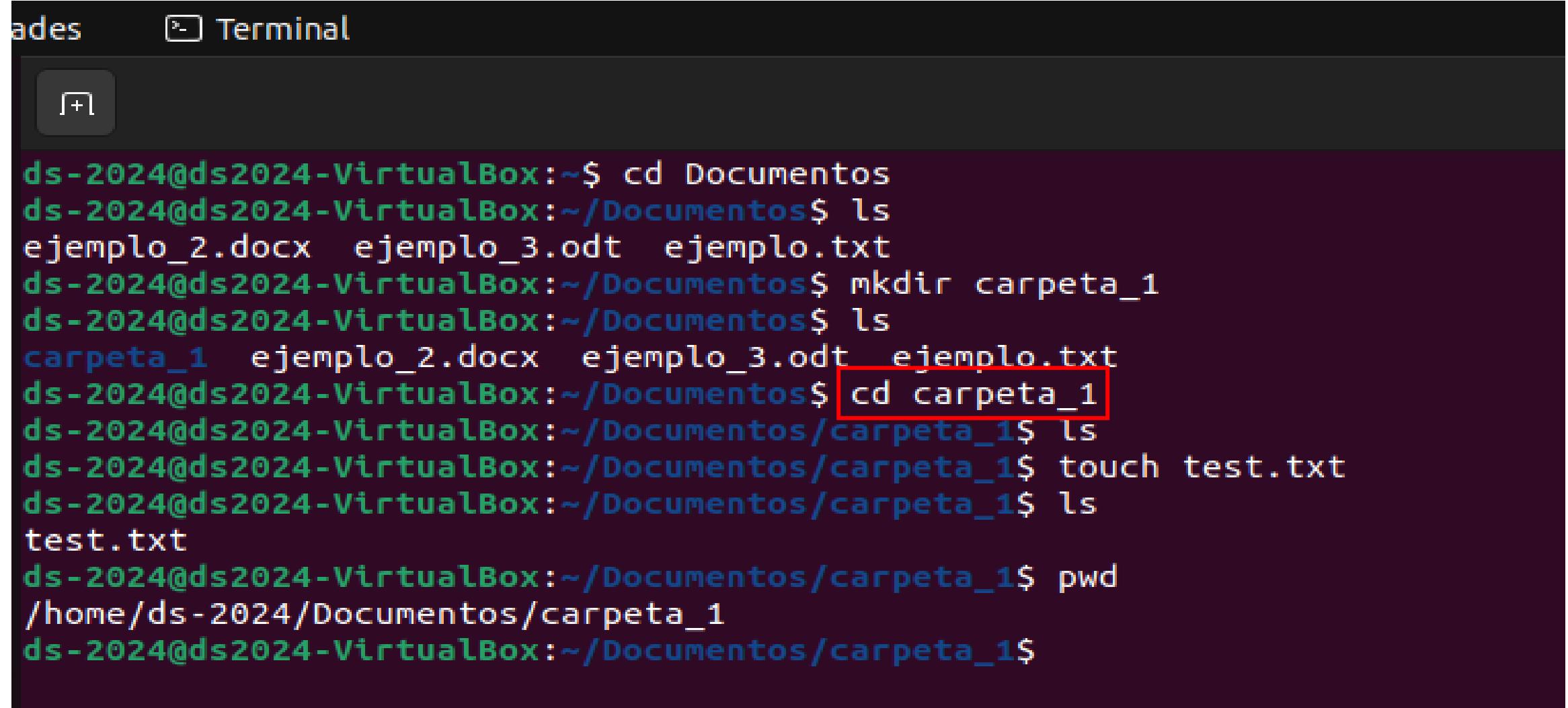
Ejemplo:

The screenshot shows a dark-themed desktop environment with a terminal window open. The terminal window has tabs for 'Ades' and 'Terminal'. A new tab icon is visible on the left. The terminal content is as follows:

```
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 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.

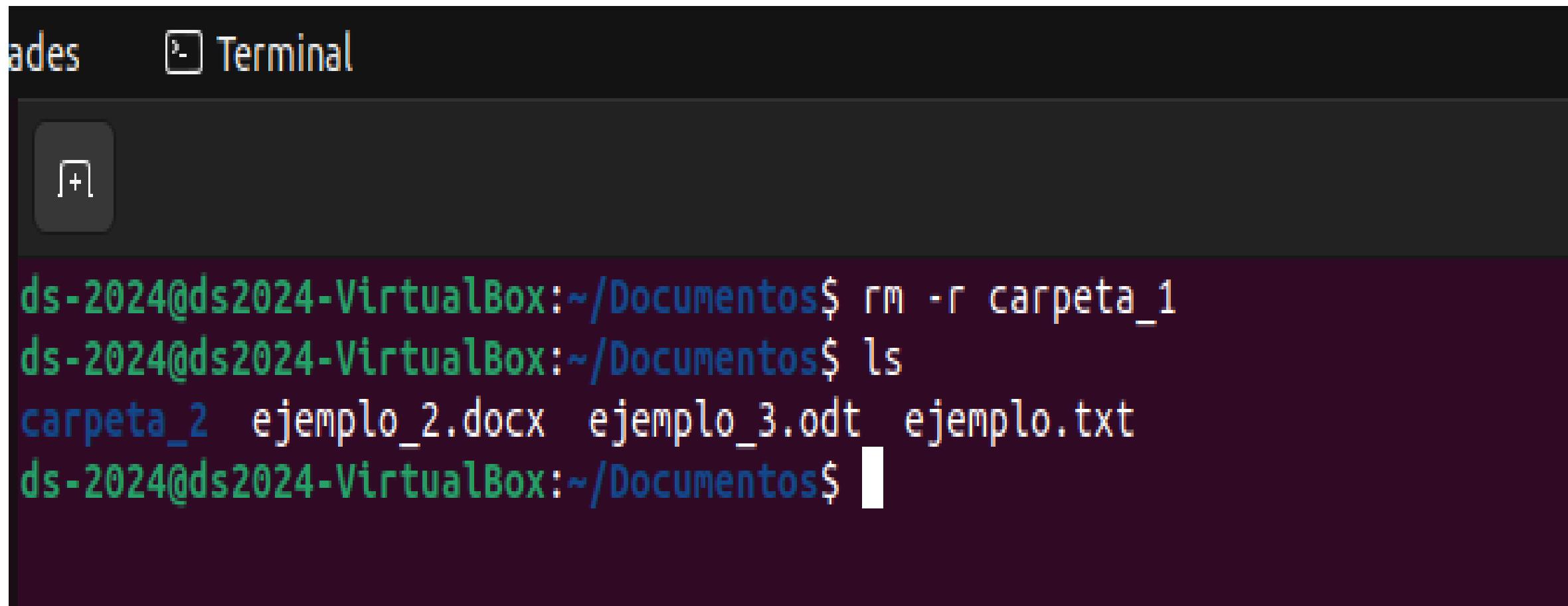
Ejemplo:

```
andes Terminal
[+]

ds-2024@ds2024-VirtualBox:~/Documentos/carpeta_1$ cd ..
ds-2024@ds2024-VirtualBox:~/Documentos$ mkdir carpeta_2
ds-2024@ds2024-VirtualBox:~/Documentos$ ls
carpeta_1  carpeta_2  ejemplo_2.docx  ejemplo_3.odt  ejemplo.txt
ds-2024@ds2024-VirtualBox:~/Documentos$ mkdir carpeta_3
ds-2024@ds2024-VirtualBox:~/Documentos$ ls
carpeta_1  carpeta_2  carpeta_3  ejemplo_2.docx  ejemplo_3.odt  ejemplo.txt
ds-2024@ds2024-VirtualBox:~/Documentos$ ls -l
total 12
drwxrwxr-x 2 ds-2024 ds-2024 4096 abr  3 09:15 carpeta_1
drwxrwxr-x 2 ds-2024 ds-2024 4096 abr  3 09:23 carpeta_2
drwxrwxr-x 2 ds-2024 ds-2024 4096 abr  3 09:23 carpeta_3
-rw-rw-r-- 1 ds-2024 ds-2024     0 abr  3 09:06 ejemplo_2.docx
-rw-rw-r-- 1 ds-2024 ds-2024     0 abr  3 09:07 ejemplo_3.odt
-rw-rw-r-- 1 ds-2024 ds-2024     0 abr  3 09:05 ejemplo.txt
ds-2024@ds2024-VirtualBox:~/Documentos$ rmdir carpeta_3
ds-2024@ds2024-VirtualBox:~/Documentos$ ls
carpeta_1  carpeta_2  ejemplo_2.docx  ejemplo_3.odt  ejemplo.txt
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

A screenshot of a macOS desktop environment. At the top, there is a dark menu bar with the Apple logo and some icons. Below the menu bar, a Dock contains several application icons, including Finder, Mail, Safari, and others. A window titled "Terminal" is open in the foreground. The terminal window has a dark background and displays a command-line session. The session starts with the prompt "ds-2024@ds2024-VirtualBox:~/Documentos\$". The user then types "rm -r carpeta_1" and presses enter. After a brief delay, they type "ls" and press enter again. The terminal shows the contents of the directory: "carpeta_2" and three files: "ejemplo_2.docx", "ejemplo_3.odt", and "ejemplo.txt". Finally, the user types "ds-2024@ds2024-VirtualBox:~/Documentos\$" and presses enter, ending the session.

```
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$
```

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

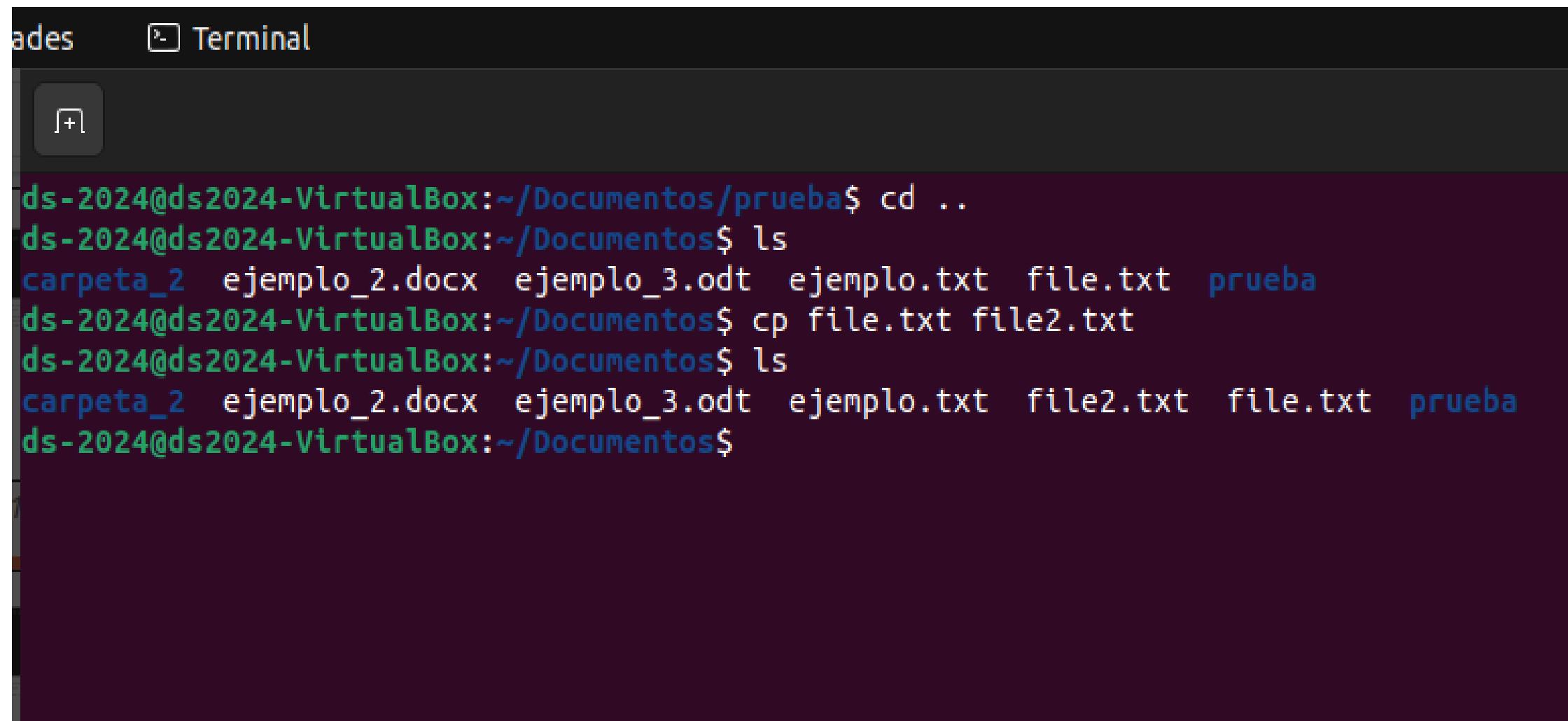
ds-2024@ds2024-VirtualBox:~/Documentos$ mkdir prueba
ds-2024@ds2024-VirtualBox:~/Documentos$ ls
carpeta_2 ejemplo_2.docx ejemplo_3.odt ejemplo.txt prueba
ds-2024@ds2024-VirtualBox:~/Documentos$ touch file.txt
ds-2024@ds2024-VirtualBox:~/Documentos$ ls
carpeta_2 ejemplo_2.docx ejemplo_3.odt ejemplo.txt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

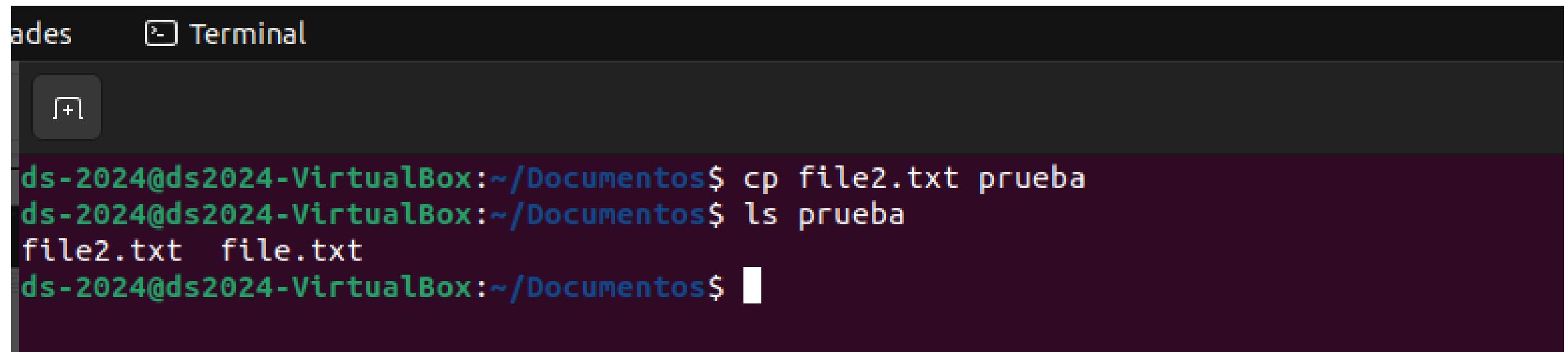


The screenshot shows a terminal window titled "Terminal" with a dark theme. The user's session is identified by "ades" at the top left. The terminal displays the following command-line session:

```
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$
```

The terminal window also features a "+" button in the top-left corner of its main area.

Otro ejemplo especificando ruta:



The screenshot shows a terminal window with a dark background and light-colored text. At the top, there's a tab bar with a 'Terminal' tab. Below the tabs is a toolbar with a '+' icon. The main area of the terminal contains the following command-line session:

```
ades      ✘ Terminal
+
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 linea
                       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 exista
                         (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                      establece el contexto de seguridad SELinux del fichero de
                         destino al tipo predeterminado
                         context[ CTX]
                         cosa 7 o si se especifica CTX anterior 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$ 
```

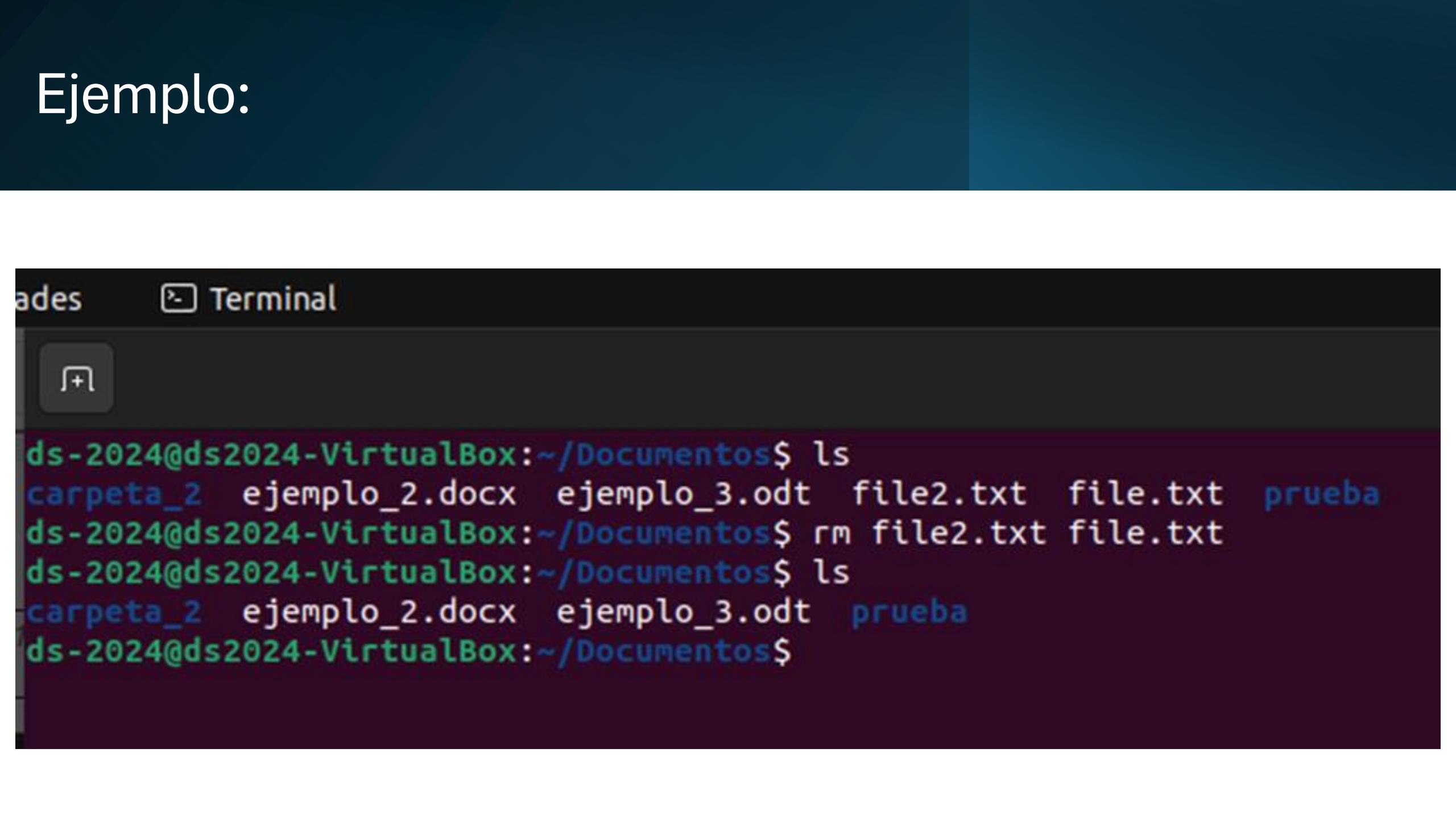
Ejemplo 2:

```
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:

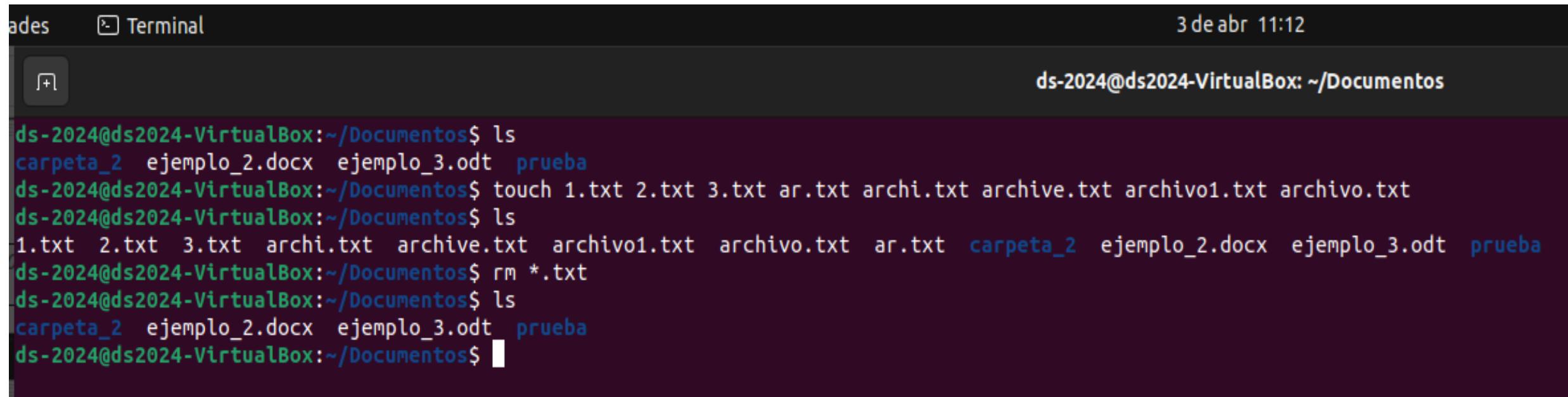


The screenshot shows a dark-themed macOS desktop environment. At the top, the Dock bar is visible with the Alfred icon and the Terminal icon. The main window is a terminal application window titled "Terminal". It contains the following command-line session:

```
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$
```

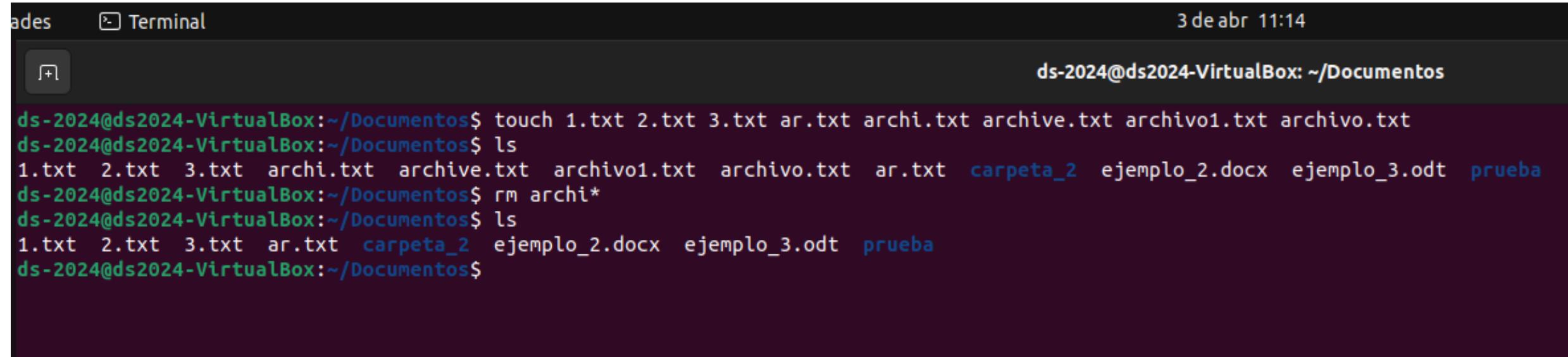
Para eliminar todos los archivos con una extensión específica: **rm *.txt**



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***

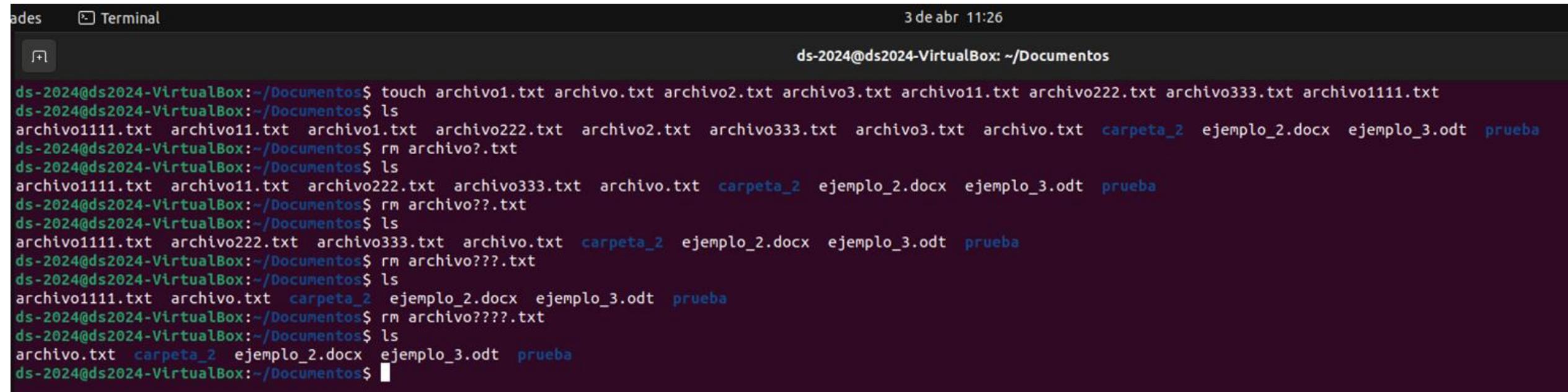


The screenshot shows a terminal window with the following details:

- Top bar: "ades" (partially visible), "Terminal", and the date/time "3 de abr 11:14".
- Icon bar: A small icon with a plus sign.
- Text area:
 - Current directory: "ds-2024@ds2024-VirtualBox: ~/Documentos"
 - Command history:

```
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



The screenshot shows a terminal window titled "Terminal" with the command line interface. The user is in the "/Documents" directory of a VirtualBox machine. The terminal output demonstrates the use of the "rm" command with a wildcard pattern to remove files containing a specific letter.

```
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 archivo2.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".
- Date and 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"
 - Multiple confirmation prompts from "rm" asking if files should be deleted, each followed by a "s" (for yes) and a new line.
 - "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:

```
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$
```

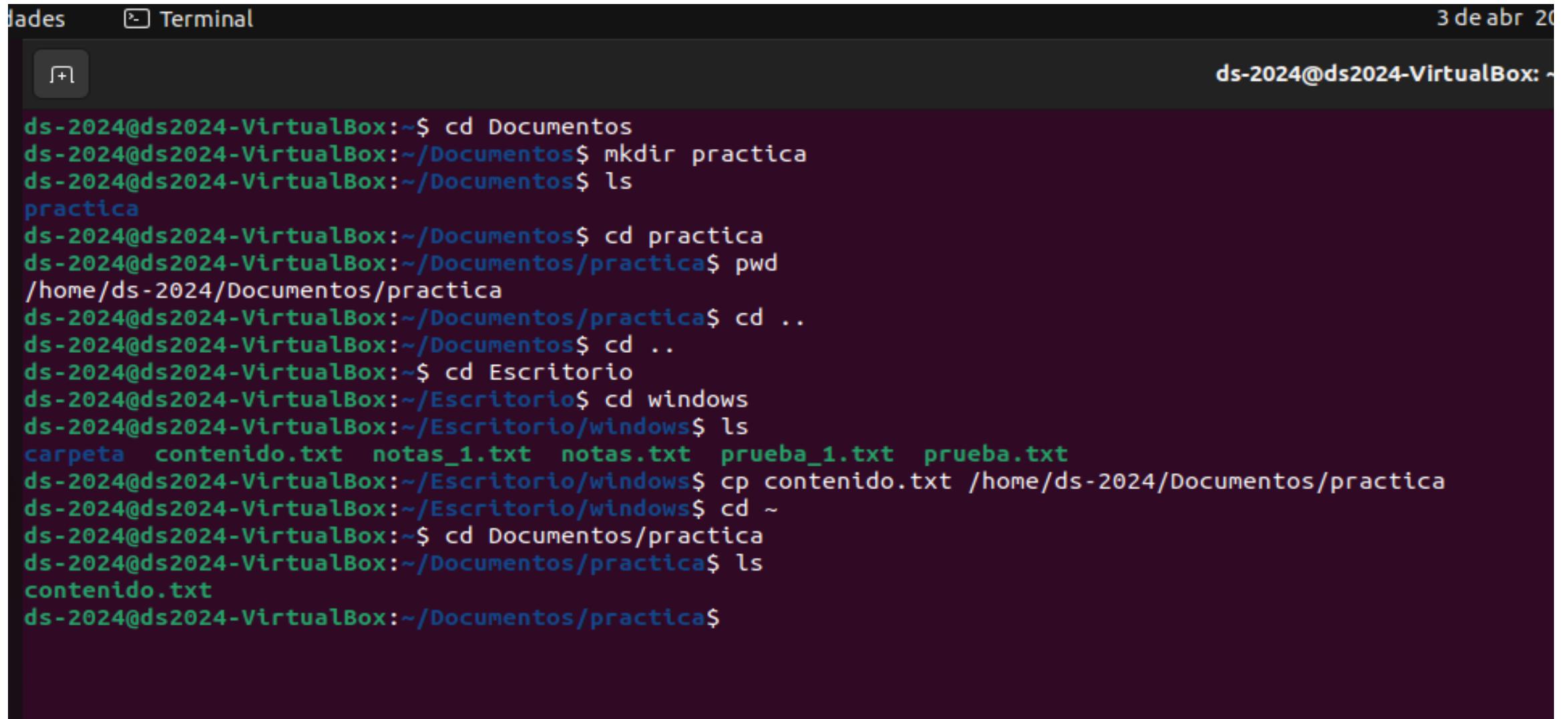
Actividad 5. Sistemas de ficheros de Windows y MacOs

- Para los usuarios de Windows , investigar la estructura o sistema de directorios de Windows (la forma como están estructuradas las carpetas y su jerarquía). Explicar sus principales ramas. Replicar los comandos dados en clase (aquellos que te permita la terminal de Windows). Realizar la entrega en un documento de PowerPoint; todos los comandos usados deben mostrarse en el archivo entregable mediante capturas de pantalla. La terminal de Windows que vas a utilizar se llama: **Powershell** y ya viene instalada por defecto. Para ejecutarlo escribe en el buscador de Windows: Powershell y presionas Enter.
- Para los usuarios de Mac, investigar la estructura o sistema de directorios de MacOs (la forma como están estructuradas las carpetas y su jerarquía). Explicar sus principales ramas. Replicar los comandos dados en clase (aquellos que te permita la terminal de Mac). Realizar la entrega en un documento de PowerPoint; todos los comandos usados deben mostrarse en el archivo entregable mediante capturas de pantalla.

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

Antes de usar el comando **cat** vamos a guardar los archivos **contenido.txt**, **contenido1.txt** y **contenido2.txt** en nuestra carpeta compartida. Luego, vamos a copiar nuestro archivo **contenido.txt** que se encuentra en la carpeta compartida dentro de una carpeta que llamaremos **práctica** que esté, a su vez, dentro de Documentos:

Los pasos a continuación:



A screenshot of a Linux desktop environment showing a terminal window. The terminal title is "Terminal". The date in the top right corner is "3 de abr 2024". The terminal window contains the following command-line session:

```
dares Terminal 3 de abr 2024
ds-2024@ds2024-VirtualBox:~$ cd Documentos
ds-2024@ds2024-VirtualBox:~/Documentos$ mkdir practica
ds-2024@ds2024-VirtualBox:~/Documentos$ ls
practica
ds-2024@ds2024-VirtualBox:~/Documentos$ cd practica
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ pwd
/home/ds-2024/Documentos/practica
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ cd ..
ds-2024@ds2024-VirtualBox:~/Documentos$ cd ..
ds-2024@ds2024-VirtualBox:~$ cd Escritorio
ds-2024@ds2024-VirtualBox:~/Escritorio$ cd windows
ds-2024@ds2024-VirtualBox:~/Escritorio/windows$ ls
carpeta contenido.txt notas_1.txt prueba_1.txt prueba.txt
ds-2024@ds2024-VirtualBox:~/Escritorio/windows$ cp contenido.txt /home/ds-2024/Documentos/practica
ds-2024@ds2024-VirtualBox:~/Escritorio/windows$ cd ~
ds-2024@ds2024-VirtualBox:~$ cd Documentos/practica
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ ls
contenido.txt
ds-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.

Vamos a copiar los archivos contenido1.txt y contenido2.txt dentro de la carpeta practica:

Jades Terminal 3 de abr 21:13

```
ds-2024@ds2024-VirtualBox:~$ cd Documentos/practica
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ ls
contenido.txt
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ pwd
/home/ds-2024/Documentos/practica
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ cd ~
ds-2024@ds2024-VirtualBox:~$ cd Escritorio/windows
ds-2024@ds2024-VirtualBox:~/Escritorio/windows$ ls
carpeta contenido1.txt contenido2.txt contenido.txt jueves.odt notas_1.txt notas.txt prueba_1.txt prueba.txt
ds-2024@ds2024-VirtualBox:~/Escritorio/windows$ cp contenido1.txt /home/ds-2024/Documentos/practica
ds-2024@ds2024-VirtualBox:~/Escritorio/windows$ cp contenido2.txt /home/ds-2024/Documentos/practica
ds-2024@ds2024-VirtualBox:~/Escritorio/windows$ ls
carpeta contenido1.txt contenido2.txt contenido.txt jueves.odt notas_1.txt notas.txt prueba_1.txt prueba.txt
ds-2024@ds2024-VirtualBox:~/Escritorio/windows$ cd ~
ds-2024@ds2024-VirtualBox:~$ cd Documentos/practica
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ ls
contenido1.txt contenido2.txt contenido.txt
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ █
```

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$
```

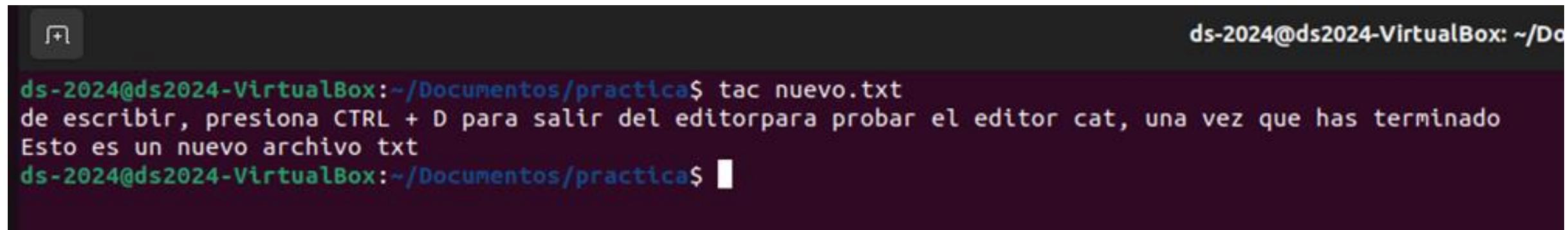
```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$
```

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 visible: "ds-2024@ds2024-VirtualBox: ~/Do". The main content of the terminal is the output of the "tac" command on a file named "nuevo.txt". The output reads:

```
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 word o pdf. Se debe hacer al menos un ejemplo de cada comando.

Edición de archivos: vi y nano

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

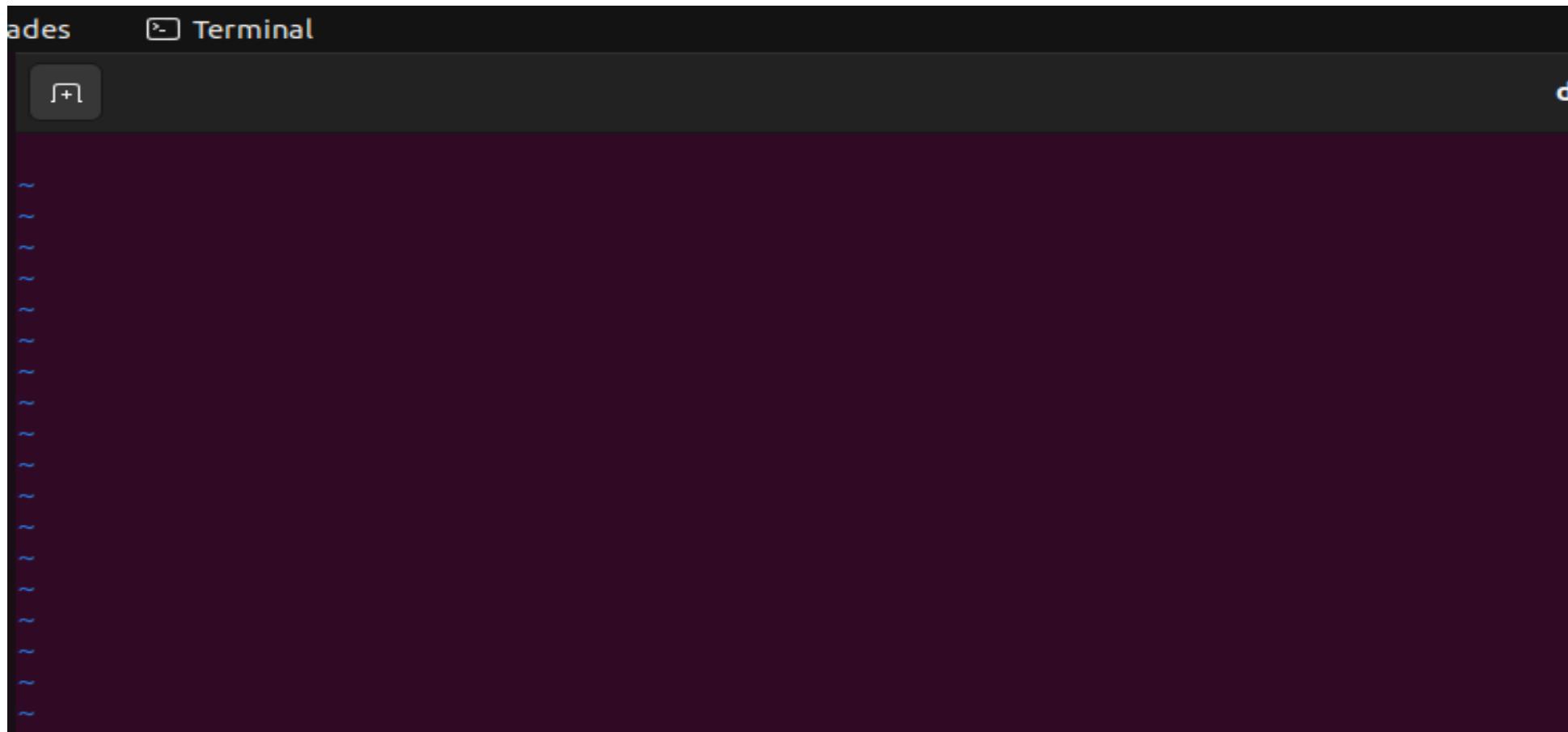
vi opera en dos modos principales: el **modo de comando** y el **modo de inserción**. En el modo de comando, los usuarios pueden navegar por el texto, realizar búsquedas, realizar ediciones y ejecutar comandos. En el modo de inserción, los usuarios pueden escribir y editar el texto de la manera habitual. 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).

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

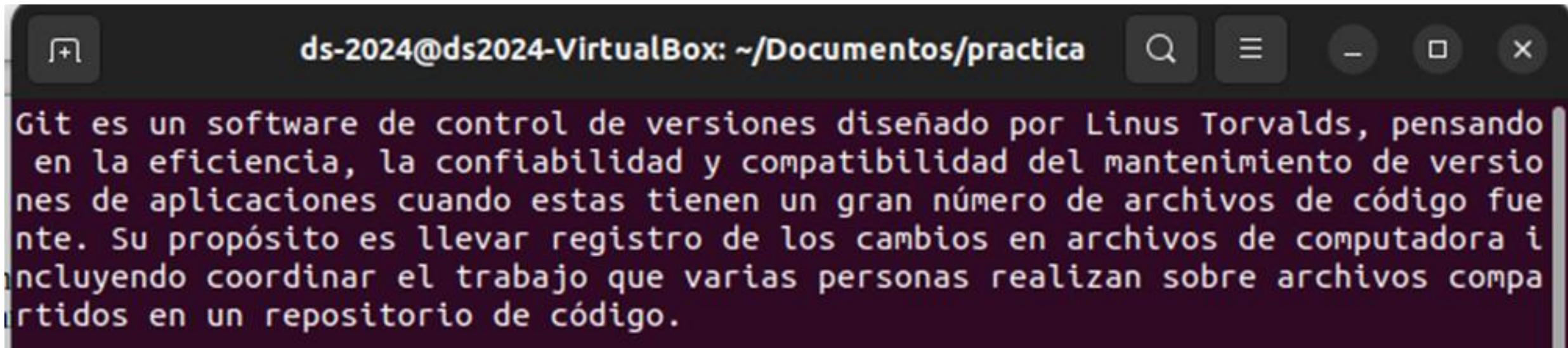
Para abrir un fichero con el editor vi basta con escribir:
vi nombre_archivo.algo
Si el fichero no existe lo crea también.

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:



The screenshot shows a terminal window with a dark background. The title bar reads "ds-2024@ds2024-VirtualBox: ~/Documentos/practica". The main area of the terminal contains the following text:

```
Git es un software de control de versiones diseñado por Linus Torvalds, pensando  
en la eficiencia, la confiabilidad y compatibilidad del mantenimiento de versio  
nes de aplicaciones cuando estas tienen un gran número de archivos de código fue  
nte. Su propósito es llevar registro de los cambios en archivos de computadora i  
ncluyendo coordinar el trabajo que varias personas realizan sobre archivos compa  
rtidos 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.



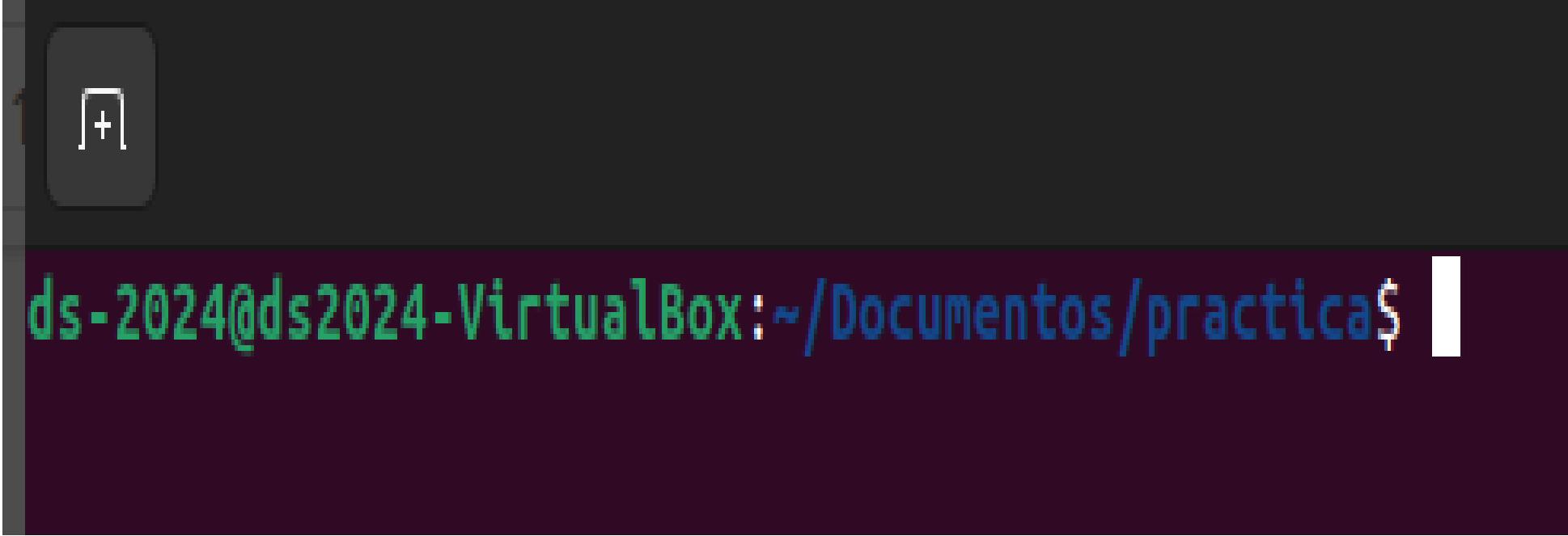
A screenshot of a terminal window with a dark background. On the left side, there is a vertical scroll bar with blue tick marks. In the bottom right corner, the text ':wq' is visible, indicating a command being entered. At the top of the window, there is a small amount of white text that is mostly illegible but includes the word 'Git'.

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



A screenshot of a terminal window with a dark background. On the left side, there is a vertical bar divided into several colored segments: grey, brown, grey, grey, and black. In the black segment, the command ':wq' is typed in white text. The terminal is otherwise empty, with no other text or output visible.

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

A screenshot of a terminal window. At the top, there is a dark header bar with a small icon containing a plus sign and a number '1'. The main area of the terminal shows a command-line interface with a green background and white text. The text reads: 'ds-2024@ds2024-VirtualBox:~/Documentos/practica\$'. A cursor arrow is visible at the end of the command line.

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: Practicando con vi

Crear un fichero con un nombre cualquiera desde la terminal , añadir contenido y luego probar todos los comandos dados arriba.

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.

Esperamos que disfrutes aprendiendo 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.

Actividad 7.

- . 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.

- . Guarda los cambios y cierra el archivo.

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.

Ejemplo:

Escribir en la consola: pr samurai.txt y verás como separa el contenido con la estructura de 66 líneas. Subir y bajar a través de la terminal.

pr samurai.txt

ds-2024@ds2024-VirtualBox:~/Documentos/practica\$ pr samurai.txt

2024-04-05 11:04 samurai.txt Página 1

What is a Shell Samurai?
A Shell Samurai has an intuitive understanding of the CLI and Linux fundamentals. They can navigate the Command Line and configure the Linux 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!
Let's learn Linux and kick ass at it together!
- Stetson Blake

Actividad

Crear un archivo txt y pegar o escribir contenido considerable (~ 50 líneas) para practicar los comandos dados anteriormente.

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.

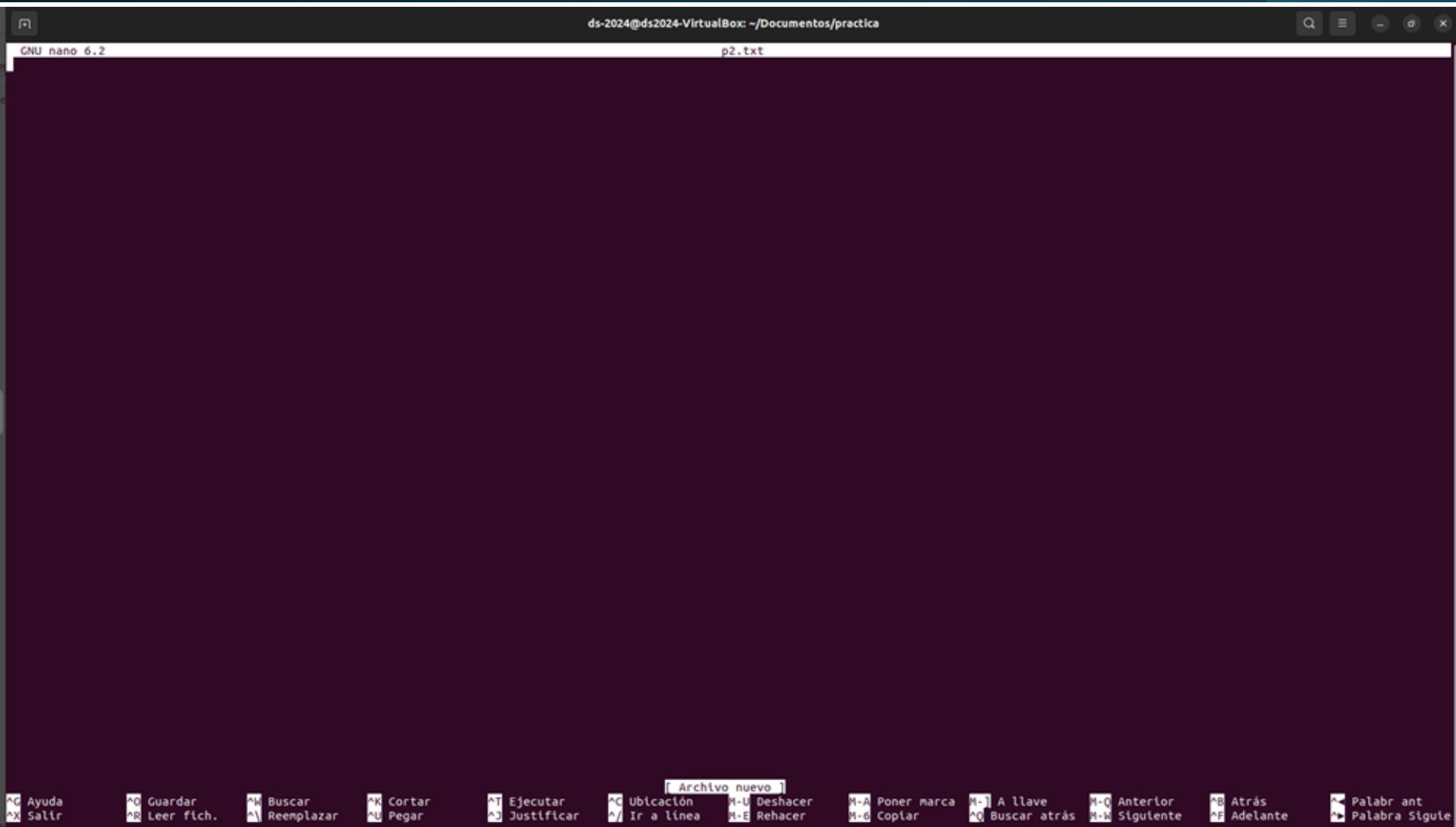
Comandos básicos de nano:

- 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.

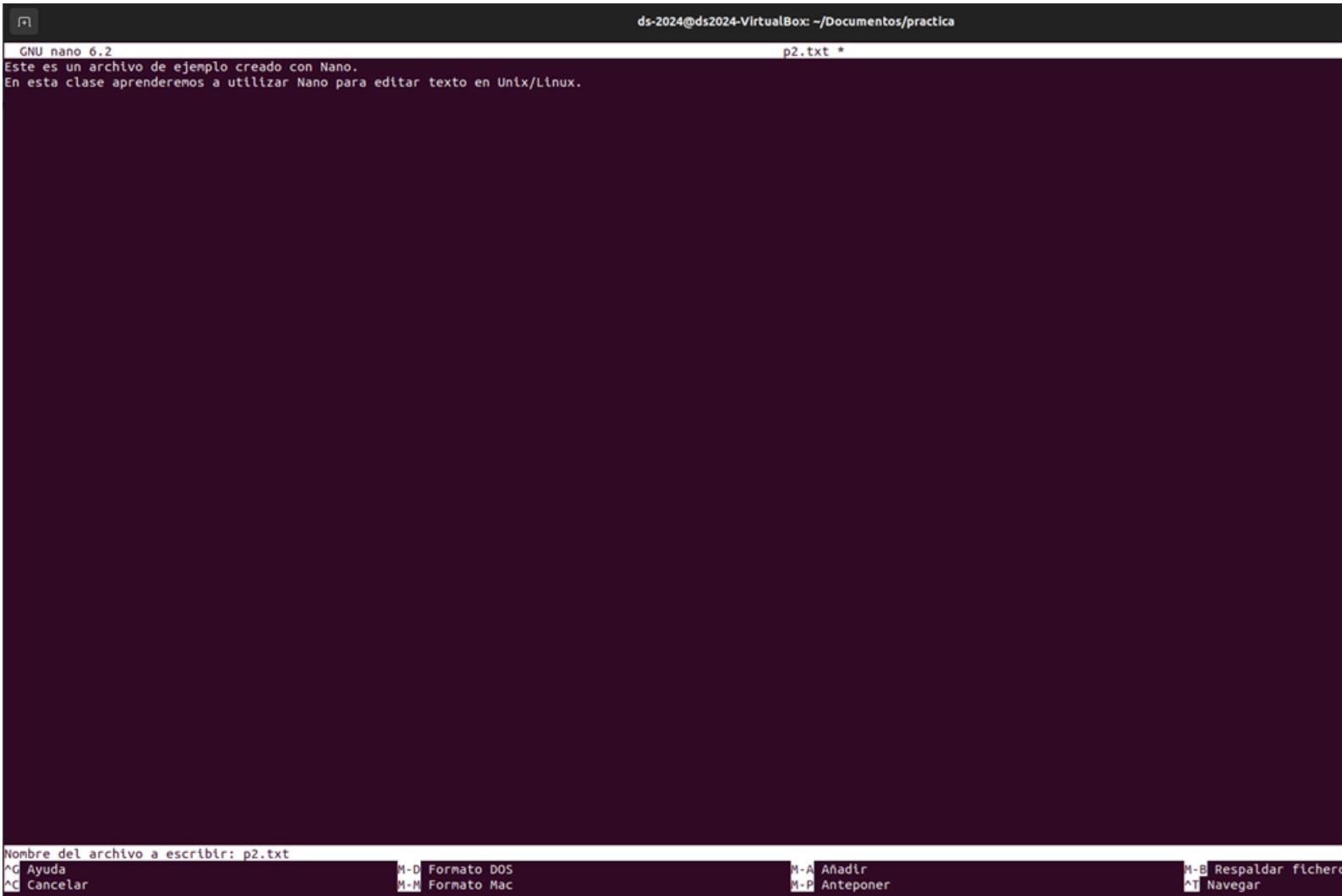
En esta clase aprenderemos a utilizar Nano para editar texto en Unix/Linux.

```
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.



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.

```
Nombre del archivo a escribir: p2.txt
^G Ayuda
^C Cancelar
```

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 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:

Con un archivo de texto .txt cualquiera practica con los comandos dados antes.

Actividad:

Con un archivo de texto .txt cualquiera practica con los comandos dados antes.

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.

Búsqueda en archivos: grep

El comando grep 'conjuntocaracteres' 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 $5/month. For some
server, be sure to use the latest version of Ubuntu, which is the Linux
The path you take to get access to a Linux Shell doesn't matter too much. All
ds-2024@ds2024-VirtualBox:~/Documentos/practica$
```

Este comando buscará todas las líneas que contienen la palabra "error" en el archivo

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 "install" /home/ds-2024/Documentos/practica
/home/ds-2024/Documentos/practica/samurai.txt:not offer a GUI. If you decide to install Linux on an old computer or laptop that
/home/ds-2024/Documentos/practica/samurai.txt:deep end. Just be cautious as many newbies may install Linux on their "main
/home/ds-2024/Documentos/practica/samurai.txt:can simply reinstall and the only cost is your time. If you go this route, be sure to
/home/ds-2024/Documentos/practica/samurai.txt:install Linux on a machine that isn't your primary computer.
/home/ds-2024/Documentos/practica/samurai.txt:You'll need a USB drive to install Ubuntu from a bootable disk or if you're a
/home/ds-2024/Documentos/practica/samurai.txt:that. Ubuntu has setup instructions for installing on a Bootable USB stick here.
/home/ds-2024/Documentos/practica/samurai.txt:- Click OK and wait for it to install.
/home/ds-2024/Documentos/practica/samurai.txt:Another great option is to use software like VirtualBox or VMWare to install a
/home/ds-2024/Documentos/practica/samurai.txt:For more info on using VirtualBox, reference the installation docs Ubuntu has
/home/ds-2024/Documentos/practica/samurai.txt:Dual Booting is a way of installing Linux alongside another operating system
/home/ds-2024/Documentos/practica/samurai.txt:on the same machine. This is a similar route to just installing Linux on a
/home/ds-2024/Documentos/practica/samurai.txt:that is the primary way you'll connect to a cloud server! When you install your
/home/ds-2024/Documentos/practica/samurai2.txt:not offer a GUI. If you decide to install Linux on an old computer or laptop that
/home/ds-2024/Documentos/practica/samurai2.txt:deep end. Just be cautious as many newbies may install Linux on their "main
/home/ds-2024/Documentos/practica/samurai2.txt:can simply reinstall and the only cost is your time. If you go this route, be sure to
/home/ds-2024/Documentos/practica/samurai2.txt:install Linux on a machine that isn't your primary computer.
/home/ds-2024/Documentos/practica/samurai2.txt:You'll need a USB drive to install Ubuntu from a bootable disk or if you're a
/home/ds-2024/Documentos/practica/samurai2.txt:that. Ubuntu has setup instructions for installing on a Bootable USB stick here.
/home/ds-2024/Documentos/practica/samurai2.txt:- Click OK and wait for it to install.
/home/ds-2024/Documentos/practica/samurai2.txt:Another great option is to use software like VirtualBox or VMWare to install a
/home/ds-2024/Documentos/practica/samurai2.txt:For more info on using VirtualBox, reference the installation docs Ubuntu has
/home/ds-2024/Documentos/practica/samurai2.txt:Dual Booting is a way of installing Linux alongside another operating system
/home/ds-2024/Documentos/practica/samurai2.txt:on the same machine. This is a similar route to just installing Linux on a
/home/ds-2024/Documentos/practica/samurai2.txt:that is the primary way you'll connect to a cloud server! When you install your
ds-2024@ds2024-VirtualBox:~/Documentos/practica$
```

Opciones comunes

- Otro ejemplo, grep -r “terminal” /home/ds-2024/Documentos/practica

```
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 linea

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

```
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

Operador pipe “|”

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 de tubería

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 de tubería

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, ejemplo: ver página 167

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 10. Ejercicios con pipe |

Crear una carpeta con nombre act10. Crear los ficheros necesarios para replicar cada uno de los siguientes ejercicios. Realizar la entrega de los comandos utilizados en .docx .txt o .pdf dentro de la carpeta act 10. Debe hacer la entrega de la carpeta en formato comprimido en .zip.

A continuación los ejercicios que debes replicar:

Ejercicios

1. Contar el número de líneas en un archivo de texto.

Sintaxis:

```
cat archivo.txt | wc -l
```

Este comando leerá el contenido del archivo **archivo.txt** utilizando **cat** y luego contará el número de líneas en ese contenido usando **wc -l**.

2. Ordenar un listado de nombres de archivos por orden alfabético

Sintaxis:

ls | sort

Este comando listará todos los archivos en el directorio actual con ls y luego los ordenará alfabéticamente con sort.

3. Buscar archivos con una extensión específica en un directorio y contarlos

Sintaxis:

```
ls *.txt | wc -l
```

Este comando listará todos los archivos con extensión .txt en el directorio actual y luego contará el número de líneas en esa lista.

4. Mostrar solo las líneas únicas en un archivo de texto

Sintaxis:

```
sort archivo.txt | uniq
```

Este comando ordenará las líneas del archivo archivo.txt con sort y luego mostrará solo las líneas únicas con uniq.

5. Mostrar solo los nombres de los archivos con extensión ".txt" en un directorio y ordenarlos alfabéticamente:

Sintaxis:

```
ls | grep ".txt" | sort | uniq
```

Este comando primero lista todos los archivos en el directorio actual con ls, luego utiliza grep para filtrar solo los archivos con extensión ".txt", luego los ordena alfabéticamente con sort, y finalmente elimina los nombres de archivos duplicados con uniq.

6. Buscar archivos modificados hoy en un directorio y contarlos

Sintaxis:

```
find . -type f -mtime -1 | sort | wc -l
```

Este comando utiliza find para buscar archivos (-type f) en el directorio actual (.) que fueron modificados en las últimas 24 horas (-mtime -1), luego los ordena alfabéticamente con sort, y finalmente cuenta el número total de líneas con wc -l.

7. Mostrar las líneas únicas en un archivo de texto y contarlas

Sintaxis:

```
cat archivo.txt | sort | uniq | wc -l
```

Este comando lee el contenido del archivo archivo.txt con cat, luego lo ordena con sort, elimina líneas duplicadas con uniq, y finalmente cuenta el número total de líneas con wc -l.

8. Filtrar las líneas que contienen una palabra específica en un archivo, ordenarlas alfabéticamente y contarlas

Sintaxis:

```
grep "palabra" archivo.txt | sort | uniq | wc -l
```

Este comando utiliza grep para encontrar todas las líneas en el archivo archivo.txt que contienen la palabra "palabra", luego las ordena alfabéticamente con sort, elimina líneas duplicadas con uniq, y finalmente cuenta el número total de líneas con wc -l.

9. Contar la cantidad de palabras únicas en un archivo de texto, luego ordenarlas alfabéticamente

Sintaxis:

```
cat archivo.txt | tr -s ' ' '\n' | sort | uniq | wc -l
```

Donde:

- cat archivo.txt: Lee el contenido del archivo archivo.txt.
- tr -s ' ' '\n': Transforma los espacios en blanco en saltos de línea, separando así cada palabra en una línea diferente. La opción -s de tr elimina los espacios en blanco repetidos.
- sort: Ordena alfabéticamente las palabras.

10. Filtrar líneas que contienen una dirección IP válida en un archivo de registro y contarlas:

Sintaxis:

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

Este comando primero lee el contenido del archivo archivo.log con cat, luego utiliza grep con una expresión regular para encontrar direcciones IP válidas, luego las ordena con sort, elimina líneas duplicadas con uniq, y finalmente cuenta el número total de líneas con wc -l.

Detalles ejercicio 10:

Para poder replicar el contenido del 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)

Cambio de modo en los archivos:

Llegado este punto, conviene hacer un alto en el camino para comentar una de las principales características de los sistemas Linux:

Están concebidos para que los utilicen diferentes usuarios de manera simultánea. Es decir, son multiusuario.

Por tanto, se hace imperativo definir un sistema de permisos que limite la propiedad y el uso que se hace de los ficheros que pertenecen a cada usuario, de manera que cada uno pueda decidir con quién comparte cada fichero dentro del propio sistema de ficheros.

Se definen por tanto tres tipos de permisos:

- Lectura, el usuario puede leer el contenido del fichero.
- Escritura, el usuario puede modificar el contenido del fichero.
- Ejecución, si el fichero es un script (contiene sentencias de comandos) o un programa ejecutable, el usuario puede ejecutarlo.

ls -l

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ ls -l
total 84
-rw-rw-r-- 1 ds-2024 ds-2024      569 abr  8 12:09 archivo.log
-rw-rw-r-- 1 ds-2024 ds-2024     822 abr  4 09:47 comprimido.zip
-rwxrwx--- 1 ds-2024 ds-2024     174 abr  4 08:31 contenido1.txt
-rwxrwx--- 1 ds-2024 ds-2024      75 abr  4 08:29 contenido2.txt
-rwxrwx--- 1 ds-2024 ds-2024    583 abr  3 20:17 contenido.txt
-rwxrwx--- 1 ds-2024 ds-2024   2242 abr  5 09:15 data.txt
-rwxrwx--- 1 ds-2024 ds-2024     336 abr  5 09:22 ejemplo.csv
-rw-rw-r-- 1 ds-2024 ds-2024     176 abr  8 10:42 lista_archivos.txt
-rw-rw-r-- 1 ds-2024 ds-2024     718 abr  4 09:22 nuevo2.txt
-rw-rw-r-- 1 ds-2024 ds-2024 10240 abr  4 10:57 nuevo_comprimido.tar
-rw-rw-r-- 1 ds-2024 ds-2024     135 abr  4 09:08 nuevo.txt
-rw-rw-r-- 1 ds-2024 ds-2024     119 abr  5 10:48 p-1.txt
-rw-rw-r-- 1 ds-2024 ds-2024     123 abr  5 11:42 p2.txt
drwxrwxr-x 2 ds-2024 ds-2024   4096 abr  4 10:27 prueba2
-rw-rw-r-- 1 ds-2024 ds-2024     441 abr  5 10:06 p.txt
-rwxrwx--- 1 ds-2024 ds-2024   7162 abr  5 11:04 samurai2.txt
-rwxrwx--- 1 ds-2024 ds-2024   7162 abr  5 11:04 samurai.txt
ds-2024@ds2024-VirtualBox:~/Documentos/practica$
```

r: read
w: write
x: execute
d: directory

¿Que significa? drwxrwxr-x

```
drwxrwxr-x 2 ds-2024 ds-2024 4096 abr 4 10:27 prueba2
```

d: Indica que el elemento es un directorio.

rw-rw-rwx: Los siguientes nueve caracteres representan los permisos del archivo o directorio. En este caso, rw-rw-rwx significa que el propietario, el grupo y otros usuarios tienen permisos de lectura, escritura y ejecución en el directorio.

Para desglosar los permisos:

- Los primeros tres caracteres (rwx) representan los permisos del **propietario** del archivo o directorio. En este caso, rwx significa que el propietario tiene permisos de lectura, escritura y ejecución en el directorio.
- Los siguientes tres caracteres (rwx) representan los permisos del **grupo** al que pertenece el archivo o directorio. En este caso, rwx significa que el grupo tiene permisos de lectura, escritura y ejecución en el directorio.
- Los últimos tres caracteres (rwx) representan los permisos para **otros usuarios** que no son el propietario ni están en el grupo. En este caso, rwx significa que otros usuarios tienen permisos de lectura, escritura y ejecución en el directorio.

Dicho de otra forma, se definen tres ámbitos de aplicación de esos permisos:

- Owner, el permiso se aplica al propietario del fichero.
- Group, el permiso se aplica a un grupo de usuarios (los usuarios pueden pertenecer a varios grupos).
- Global, el permiso se aplica a todos los usuarios del sistema.

Los permisos de cada archivo se pueden ver con el comando ls -l. La salida de este comando ofrece una lista de ficheros con la siguiente información

- La primera columna muestra el tipo de fichero y sus permisos.
- La segunda muestra el número de enlaces a ese fichero.
- La tercera indica qué usuario es el propietario del fichero.
- La cuarta indica el grupo al que pertenece el fichero.
- La quinta muestra el tamaño en bytes del fichero. Los directorios, según qué sistema de ficheros se use, suelen ocupar un bloque de datos completo (4096 bytes).
- Las columnas sexta a octava muestran la fecha y hora de última modificación.
- La novena muestra el nombre del fichero.

Desgranando la columna de permisos:

- El primer carácter indica el tipo de fichero: “d” se trata de un directorio, “l” un enlace simbólico y “-“ un fichero ordinario.
- Los siguientes caracteres definen todos los permisos, agrupados en tres conjuntos de tres caracteres cada uno. El primer grupo define los permisos del usuario propietario del fichero. El segundo los del grupo y el tercero los permisos globales.
- Cada grupo de permisos se define mediante los caracteres “rwx”, donde r indica permiso de lectura, w de escritura y x de ejecución. Si en una de esas posiciones aparece el carácter “-“, quiere decir que esa entidad (usuario, grupo o global) no tiene el permiso que se define en esa posición.

Para cambiar los permisos de un archivo se emplea el comando **chmod** [quien] oper, permiso, files, donde:

- **[quien]** indica a quién afecta el permiso que se desea cambiar. Es una combinación de las letras **u** para el usuario, **g** para el grupo del usuario, **o** para los otros usuarios y **a** para todos los anteriores. Si no se indica nada, se supone **a**.
- **oper** indica si el permiso se da usando un **+** o se quita usando un **-**.
- **permiso** indica el permiso que se quiere dar o quitar. Es una combinación de las letras: **r** (leer), **w** (escribir), **x** (ejecutar). Estos permisos son diferentes cuando se aplican a directorios: **r** da la posibilidad de ver el contenido del directorio con el comando **ls**; el permiso **w** da la posibilidad de crear y borrar archivos en ese directorio; el permiso **x** autoriza a buscar y utilizar un archivo concreto.
- **files** son los nombres de los archivos o directorios cuyos modos de acceso se quieren cambiar.

Ejemplo:

- Vamos a crear un archivo con nombre **prueba.txt**
- Escribimos: **ls -l prueba.txt** para ver los permisos que han sido creados por defecto.

```
comprimidos.zip  contenido2.txt  datos.txt  lista_de_comandos.txt  
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ ls -l prueba.txt  
-rwx-rwxr-- 1 ds-2024 ds-2024 0 abr  8 13:17 prueba.txt
```

- Vamos a darle permisos de ejecución a todos:

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ chmod +x prueba.txt  
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ ls -l prueba.txt  
-rwxrwxr-x 1 ds-2024 ds-2024 0 abr  8 13:17 prueba.txt  
ds-2024@ds2024-VirtualBox:~/Documentos/practica$
```

Ejemplo:

- Vamos a quitarle todos los permisos a los ‘otros usuarios’

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ chmod o-rwx prueba.txt
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ ls -l prueba.txt
-rwxrwx--- 1 ds-2024 ds-2024 0 abr  8 13:17 prueba.txt
ds-2024@ds2024-VirtualBox:~/Documentos/practica$
```

Cambio de propietario a un conjunto de archivos:

- El comando **chown** se emplea para cambiar de propietario a un determinado conjunto de archivos:
 - **chown newowner file1 file2...**
 - “La lista completa de usuarios del sistema se puede ver en el fichero “**/etc/passwd**”

Cambio de grupo de un archivo

- El comando **chgrp** se emplea para cambiar el grupo al que pertenece un archivo:
 - **chgrp newgroup file1 file2...**
 - Los grupos de usuarios están almacenados en el archivo “**/etc/group**”.

Actividad 11.

- Investigar cómo se crean nuevos usuarios y nuevos grupos desde una terminal de Linux (o Mac para los usuarios Mac). Crear por lo menos dos usuarios y tres grupos.
- Una vez creado los usuarios y grupos desarrolle varios ejemplos sobre el uso de los comandos **chown** y **chgrp**.
- **Realizar la entrega en un documento word, powerpoint o pdf, añada las capturas de los comandos usados también.**

Comando ln. Enlaces

Un enlace es una conexión entre un nombre de archivo y los datos en el disco. Hay dos tipos principales de enlaces: enlaces simbólicos y enlaces duros.

Tipos de Enlaces:

- Enlaces Simbólicos (Soft Links):

También conocidos como enlaces suaves o soft links. Son referencias a otro archivo o directorio mediante su ruta relativa o absoluta. Imagina un acceso directo

- Enlaces Duros (Hard Links):

También conocidos como enlaces físicos o hard links. Son referencias directas a los datos del archivo en el disco. Imagina una copia exacta de un archivo.

Ejemplo de enlace duro:

Crea un archivo llamado original.txt con contenido, escribir en el terminal:

```
echo "Este es el archivo original." > original.txt
```

Crea un enlace duro llamado enlace-duro.txt, escribe en el terminal:

```
ln original.txt enlace-duro.txt
```

Verificar con un cat, escribir en el terminal: cat original.txt

Verificar con un cat el archivo enlace-duro.txt, escribir en el terminal cat enlace-duro.txt. Verás que el contenido es el mismo.

Captura de terminal

```
ds-2024@ds2024-VirtualBox: ~/Documentos/practica
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ echo "Este es el archivo original." > original.txt
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ ln original.txt enlace-duro.txt
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ ls
archivo.log  contenido1.txt  contenido.txt  ejemplo.csv    lista_archivos.txt  nuevo_comprimido.tar  original.txt  p2.txt  prueba.txt  samurai2.txt
comprimido.zip  contenido2.txt  data.txt    enlace-duro.txt  nuevo2.txt        nuevo.txt      p-1.txt   prueba2  p.txt   samurai.txt
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ cat original.txt
Este es el archivo original.
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ cat enlace-duro.txt
Este es el archivo original.
ds-2024@ds2024-VirtualBox:~/Documentos/practica$
```

Editar el ‘archivo’ enlace-duro.txt

nano enlace-duro.txt

Cuando se abra el editor escribir en una línea: “Esta es una edición desde el enlace-duro” guardar y salir.

Abre el archivo original.txt escribiendo cat original.txt y veras que aparece: “Esta es una edición desde el enlace-duro”

Similarmente, abre el archivo original.txt desde nano: nano original.txt.

Cuando se abra el editor escribir en una línea: “Esta es una edición desde el archivo original.txt” guardar y salir.

Abre enlace-duro.txt escribiendo cat enlace-duro.txt y veras que aparece: “Esta es una edición desde el archivo original.txt”

Captura de terminal

```
+  
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ nano enlace-duro.txt
```

```
+  
GNU nano 6.2  
Este es el archivo original.  
  
Esta es una edición desde el enlace-duro
```

Captura de terminal

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ cat original.txt  
Este es el archivo original.
```

Esta es una edición desde el enlace-duro

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

Capturas del terminal

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ nano original.txt
```

```
GNU nano 6.2
Este es el archivo original.

Esta es una edición desde el enlace-duro

Esta es una edición desde el archivo original.txt
```

Capturas del terminal

+

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ cat enlace-duro.txt  
Este es el archivo original.
```

Esta es una edición desde el enlace-duro

```
Esta es una edición desde el archivo original.txt  
ds-2024@ds2024-VirtualBox:~/Documentos/practica$
```

Ejemplo de enlace simbólico

Un enlace simbólico es un tipo de acceso directo al archivo original. Si el archivo original se elimina, el enlace simbólico no funcionará.

Crea un enlace simbólico llamado enlace-simbolico.txt

```
ln -s original.txt enlace-simbolico.txt
```

Verifica que el enlace simbólico apunta al archivo original.

```
ls -l enlace-simbolico.txt
```

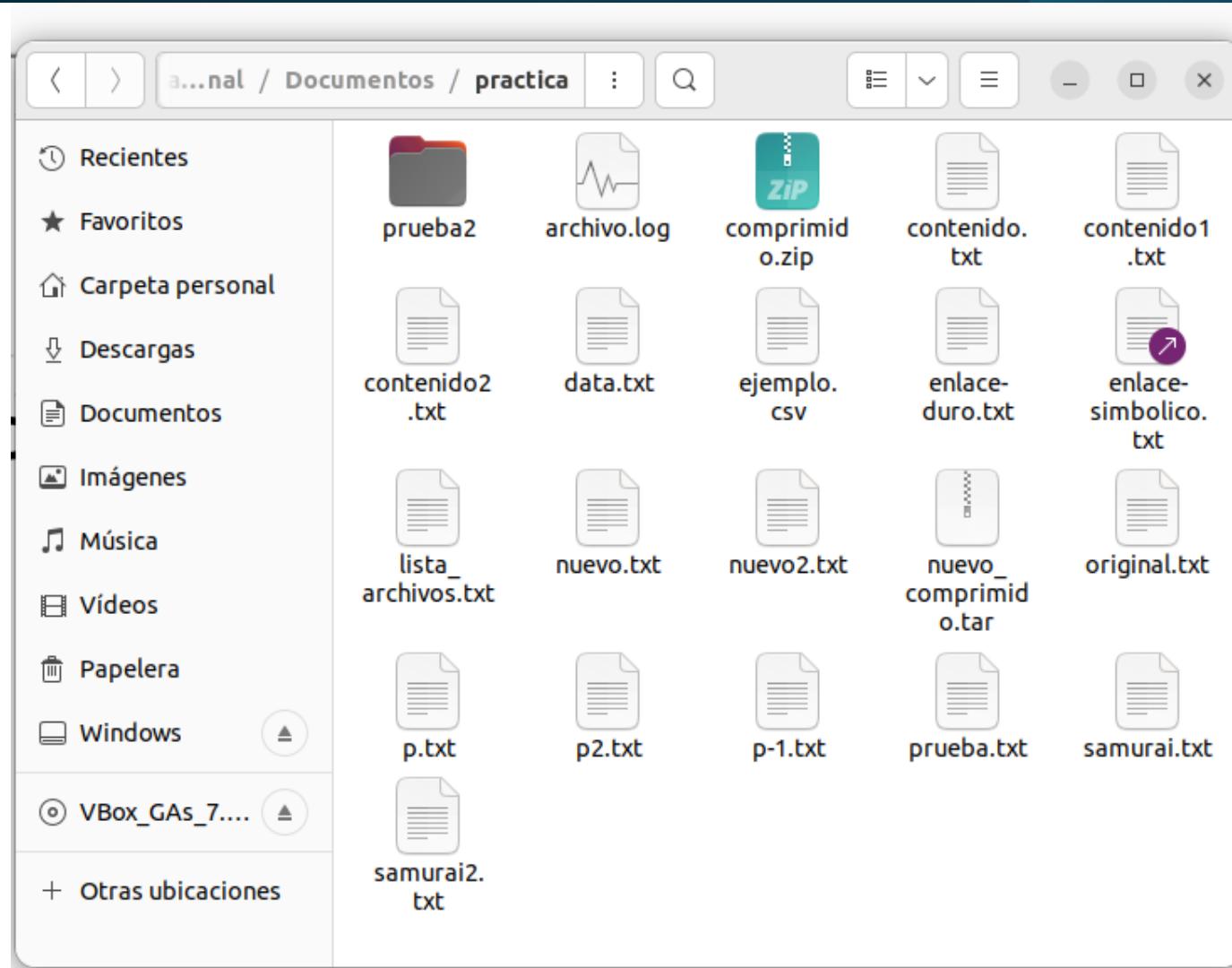
Captura del terminal

```
ds-2024@ds-2024-VirtualBox:~/Documentos/practica$ ln -s original.txt enlace-simbolico.txt
ds-2024@ds-2024-VirtualBox:~/Documentos/practica$ ls -l enlace-simbolico.txt
lrwxrwxrwx 1 ds-2024 ds-2024 12 abr 9 09:59 enlace-simbolico.txt -> original.txt
ds-2024@ds-2024-VirtualBox:~/Documentos/practica$ █
```

Captura del terminal

```
ds-2024@ds2024-VirtualBox:~/Documentos/practica$ ls -l
total 92
-rw-rw-r-- 1 ds-2024 ds-2024      569 abr  8 12:09 archivo.log
-rw-rw-r-- 1 ds-2024 ds-2024     822 abr  4 09:47 comprimido.zip
-rwxrwx--- 1 ds-2024 ds-2024     174 abr  4 08:31 contenido1.txt
-rwxrwx--- 1 ds-2024 ds-2024      75 abr  4 08:29 contenido2.txt
-rwxrwx--- 1 ds-2024 ds-2024    583 abr  3 20:17 contenido.txt
-rwxrwx--- 1 ds-2024 ds-2024   2242 abr  5 09:15 data.txt
-rwxrwx--- 1 ds-2024 ds-2024     336 abr  5 09:22 ejemplo.csv
-rw-rw-r-- 2 ds-2024 ds-2024     126 abr  9 09:53 enlace-duro.txt
lrwxrwxrwx 1 ds-2024 ds-2024      12 abr  9 09:59 enlace-simbolico.txt -> original.txt
-rw-rw-r-- 1 ds-2024 ds-2024     176 abr  8 10:42 lista_archivos.txt
-rw-rw-r-- 1 ds-2024 ds-2024    718 abr  4 09:22 nuevo2.txt
-rw-rw-r-- 1 ds-2024 ds-2024  10240 abr  4 10:57 nuevo_comprimido.tar
-rw-rw-r-- 1 ds-2024 ds-2024     135 abr  4 09:08 nuevo.txt
-rw-rw-r-- 2 ds-2024 ds-2024     126 abr  9 09:53 original.txt
-rw-rw-r-- 1 ds-2024 ds-2024     119 abr  5 10:48 p-1.txt
-rw-rw-r-- 1 ds-2024 ds-2024     123 abr  5 11:42 p2.txt
drwxrwxr-x 2 ds-2024 ds-2024   4096 abr  4 10:27 prueba2
-rwxrwx--- 1 ds-2024 ds-2024       0 abr  8 13:17 prueba.txt
-rw-rw-r-- 1 ds-2024 ds-2024     441 abr  5 10:06 p.txt
-rwxrwx--- 1 ds-2024 ds-2024   7162 abr  5 11:04 samurai2.txt
-rwxrwx--- 1 ds-2024 ds-2024   7162 abr  5 11:04 samurai.txt
ds-2024@ds2024-VirtualBox:~/Documentos/practica$
```

Captura desde la carpeta practica.



Cuando trabajas con enlaces en Linux, es importante tener en cuenta las siguientes diferencias y limitaciones:

Enlaces Duros:

- Restricción de Sistema de Archivos: Los enlaces duros no pueden cruzar diferentes sistemas de archivos o particiones. Esto significa que no puedes crear un enlace duro a un archivo que se encuentra en una partición diferente a la del enlace.
- Persistencia de Datos: Si eliminas el archivo original, el enlace duro seguirá existiendo y contendrá los datos, ya que ambos apuntan al mismo inode.

Cuando trabajas con enlaces en Linux, es importante tener en cuenta las siguientes diferencias y limitaciones:

Enlaces Simbólicos:

- Flexibilidad: Los enlaces simbólicos pueden apuntar a cualquier archivo o directorio, independientemente de la ubicación o el sistema de archivos.
- Enlaces Rotos: Si el archivo o directorio original se mueve o se elimina, el enlace simbólico se rompe, ya que su referencia ya no es válida.

Expresiones regulares

Las expresiones regulares (ER) son una forma de describir cadenas de caracteres.

Una expresión regular es un **patrón** capaz de reconocer o filtrar cadenas de caracteres según ciertos criterios. El uso de comodines * para indicar cadenas de caracteres cualesquiera o ? para indicar un carácter único son ejemplos de uso de expresiones regulares.

Expresiones regulares

Por ejemplo, el patrón **aba*** reconoce cadenas como abaco, abajo, abatimiento, abalorio, aba-23;

el patrón **do?** reconoce cadenas como doy dos, dot, don, do\$;

el patrón **aba*-txt** describe el conjunto de cadenas de caracteres que comienzan con aba, contienen cualquier otro grupo de caracteres, luego un guión, y finalmente la cadena txt.

Expresiones regulares

Los patrones construidos como expresiones regulares permiten reconocer cadenas de caracteres de estructura compleja; esto las hace muy útiles para realizar búsquedas o sustituciones en textos.

Las expresiones regulares son reconocidas por muchos lenguajes de programación, editores y otras herramientas. El editor vi y sus variantes vim y gvim las aceptan.

Expresiones regulares

Las expresiones regulares se construyen como las expresiones aritméticas, usando **operadores** para combinar expresiones más pequeñas. Analizaremos esos operadores y las reglas de construcción de expresiones regulares, atendiendo siempre al conjunto de cadenas que representa cada patrón

Metacaracteres

La construcción de expresiones regulares depende de la asignación de significado especial a algunos caracteres.

En el patrón **aba*-txt** el carácter ***** no vale por sí mismo, como el carácter asterisco, sino que indica un "conjunto de caracteres cualesquiera". Asimismo, el carácter **?** no se interpreta como el signo de interrogación, sino que representa "**un carácter cualquiera y uno solo**".

Estos caracteres a los que se asigna significado especial se denominan metacaracteres.

El conjunto de metacaracteres para expresiones regulares es el siguiente:

\ ^ \$. [] { } | () * + ?

Estos caracteres, en una expresión regular, son interpretados en su significado especial y no como los caracteres que normalmente representan. Una búsqueda que implique alguno de estos caracteres obligará a "escaparlo" de la interpretación. Esto puede hacerse anteponiéndole el símbolo \, como se hace para evitar la interpretación por el shell de los metacaracteres propios del shell.

Metacaracteres

En una expresión regular, el carácter ? representa "**un carácter cualquiera**"; si queremos representar el carácter tal cual, como signo de interrogación, debemos escribir \?.

Los caracteres anteriores, para ser interpretados como tales y no como metacaracteres, deben escribirse anteponiendo \.

Versiones de sintaxis

Existen varias versiones de sintaxis para expresiones regulares. Las más sencillas son las expresiones regulares básicas (BRE, Basic Regular Expressions). Una sintaxis ampliada son las llamadas expresiones regulares extendidas (ERE, Extended Regular Expressions).

En Linux, comandos como **grep** para filtrar líneas de texto según un patrón soportan las expresiones regulares básicas y extendidas sin diferenciación.

Versiones de sintaxis

Existen varias versiones de sintaxis para expresiones regulares. Las más sencillas son las expresiones regulares básicas (BRE, Basic Regular Expressions). Una sintaxis ampliada son las llamadas expresiones regulares extendidas (ERE, Extended Regular Expressions).

En Linux, comandos como **grep** para filtrar líneas de texto según un patrón soportan las expresiones regulares básicas y extendidas sin diferenciación.

Expresiones regulares básicas (BRE)

Una expresión regular (ER) determina un conjunto de cadenas de caracteres. Un miembro de este conjunto de cadenas se dice que aparece (**match**), equipara o satisface la expresión regular.

ERs de un solo caracter

Las expresiones regulares se componen de expresiones regulares elementales que aparean con un único carácter:

Expresión	aparea con
c	caracter ordinario c
.	(punto) aparea con un carácter cualquiera excepto nueva línea
[abc]	ER de un carácter que aparea con uno de a, b, c
[^abc]	ER de un carácter que no sea uno de a, b, c
[0-9] [a-z]	ERs de un carácter que aparea con cualquier carácter en el intervalo indicado
[A-Z]	El signo – indica un intervalo de caracteres consecutivos
\e	ER que aparea con alguno de estos caracteres (en lugar de la e): • * [\] cuando no están dentro de [] ^ al principio de la ER, o al principio dentro de [] \$ al final de una ER / usado para delimitar una ER

Corchetes

Los corchetes [] delimitan listas de caracteres individuales. Muchos metacaracteres pierden su significado si están dentro de estas listas: los caracteres especiales . * [\ valen por sí dentro de [].

Para incluir un caracter] en una lista, colocarlo al principio; para incluir un ^ colocarlo en cualquier lugar menos al principio; para incluir un - colocarlo al final.

Dentro de los conjuntos de caracteres individuales, se reconocen las siguientes categorías:

[:alnum:]	caracteres alfanuméricos
[:alpha:]	caracteres alfabéticos
[:cntrl:]	caracteres de control
[:digit:]	dígitos
[:graph:]	caracteres gráficos
[:lower:]	minúsculas
[:print:]	caracteres imprimibles
[:punct:]	signos de puntuación
[:space:]	espacios
[:upper:]	mayúsculas
[:xdigit:]	dígitos hexadecimales

En estos nombres de categorías, los paréntesis rectos forman parte del nombre de la categoría, no pueden ser omitidos.

Las categorías también pueden escribirse en forma explícita. Por ejemplo, `[[:alnum:]]` significa `[0-9A-Za-z]`.

Construcción de Expresiones Regulares

Una expresión regular se construye con uno o más operadores que indican, cada uno, el carácter a buscar. Los operadores más comunes y aceptados son los siguientes:

Operador	Significado
c	un carácter no especial concuerda consigo mismo
\c	elimina significado especial de un carácter c; el \ escapa el significado especial
^	indica ubicado al comienzo de la línea (cadena nula al principio de línea)
\$	indica ubicado al final de la línea (cadena nula al final de línea)
.	(punto) un carácter individual cualquiera
[...]	uno cualquiera de los caracteres incluidos; acepta listas del tipo a-z, 0-9, A-Z
[^...]	un carácter distinto de los indicados; acepta intervalos del tipo a-z, 0-9, A-Z
r*	0, 1 o más ocurrencias de la ER r (repetición)
r1r2	la ER r1 seguida de la ER r2 (concatenación)

Ejemplos de Expresiones Regulares Básicas

Una expresión regular se construye con uno o más operadores que indican, cada uno, el carácter a buscar. Los operadores más comunes y aceptados son los siguientes:

Operador	Significado
c	un carácter no especial concuerda consigo mismo
\c	elimina significado especial de un carácter c; el \ escapa el significado especial
^	indica ubicado al comienzo de la línea (cadena nula al principio de línea)
\$	indica ubicado al final de la línea (cadena nula al final de línea)
.	(punto) un carácter individual cualquiera
[...]	uno cualquiera de los caracteres incluidos; acepta listas del tipo a-z, 0-9, A-Z
[^...]	un carácter distinto de los indicados; acepta intervalos del tipo a-z, 0-9, A-Z
r*	0, 1 o más ocurrencias de la ER r (repetición)
r1r2	la ER r1 seguida de la ER r2 (concatenación)

Las expresiones regulares se aprenden mejor con los ejemplos y el uso.

Expresión	aparea con
a . b	axb aab abb aSb a#b ...
a .. b	axxb aaab abbb a4\$b ...
[abc]	a b c (cadenas de un carácter)
[aA]	a A (cadenas de un carácter)
[aA] [bB]	ab Ab aB AB (cadenas de dos caracteres)
[0123456789]	uno de 0 1 2 3 4 5 6 7 8 9
[0-9]	uno de 0 1 2 3 4 5 6 7 8 9
[A-Za-z]	uno de A B C ... Z a b c ... z
[0-9] [0-9] [0-9]	tres dígitos 000 001 .. 009 010 .. 019 100 .. 999
[0-9] *	cadena_vacía 0 1 9 00 99 123 456 999 9999 ...
[0-9] [0-9] *	0 1 9 00 99 123 456 999 9999 99999999 ...
[0-9] [0-9] +	00 01 29 474 99 123 456 999 9999 99999999 ...
^ . * \$	cualquier línea completa
^ \$	una línea vacía

Expresiones Regulares Extendidas

Las expresiones regulares extendidas agregan a las expresiones regulares básicas algunos operadores que permiten construcciones más complejas:

Operador	Significado
r^+	1 o más ocurrencias de la ER r
$r^?$	0 o una ocurrencia de la ER r, y no más
$r\{n\}$	n ocurrencias de la ER r
$r\{n, \}$	n o más ocurrencias de la ER r
$r\{, m\}$	0 o a lo sumo m ocurrencias de la ER r
$r\{n, m\}$	n o más ocurrencias de la ER r, pero a lo sumo m
$r_1 r_2$	la ER r_1 o la ER r_2 (alternativa)
(r)	ER anidada
$"r"$	evita que los caracteres de la ER r sean interpretados por el shell

Ejemplos de Expresiones Regulares Extendidas

Expresión	aparea con
$[0-9]^+$	0 1 9 00 99 123 456 999 9999 99999 99999999 ..
$[0-9]?$	cadena_vacía 0 1 2 .. 9
$^a b$	a b
$(ab)^*$	cadena_vacía ab abab ababab ...
$^*[0-9]b$	b 0b 1b 2b .. 9b
$([0-9]^+ab)^*$	cadena_vacía 1234ab 9ab9ab9ab 9876543210ab 99ab99ab ...

Expresiones regulares en Linux

En Linux no se distinguen expresiones regulares básicas de extendidas; los comandos aceptan todas las expresiones regulares. En ese caso, como siempre se están usando extendidas, los metacaracteres:

? + { | ()

deben ser escapados cuando se quieren usar como caracteres normales, escribiendo:

\? \+ \{ \| \(\)

El comando grep

Para experimentar con expresiones regulares, es útil el comando grep (Global Regular Expression and Print). Este comando busca un patrón expresado como una expresión regular dentro de uno o varios archivos de texto o de su entrada estándar, extrayendo las líneas donde se encuentra el patrón indicado. Su sintaxis es:

```
grep [opciones] patrón [archivo] ...
```

donde el patrón a buscar es una expresión regular.

```
grep bash /etc/passwd
```

```
cat /etc/passwd | grep bash
```

en ambos casos, se obtienen las líneas del archivo /etc/passwd donde figura la palabra bash, y solo éas.

Ejemplos:

Los siguientes ejemplos enfatizan la funcionalidad de las expresiones regulares. Se necesitarán datos dentro del fichero, por lo tanto, el siguiente conjunto de comandos sólo agrega diferentes cadenas de texto al fichero **text.txt**.

Ejemplos:

Los siguientes ejemplos enfatizan la funcionalidad de las expresiones regulares. Se necesitarán datos dentro del fichero, por lo tanto, el siguiente conjunto de comandos sólo agrega diferentes cadenas de texto al fichero **text.txt**.

Escribir en la terminal:

```
$ echo "aaabbb1" > text.txt  
$ echo "abab2" >> text.txt  
$ echo "noone2" >> text.txt  
$ echo "class1" >> text.txt  
$ echo "alien2" >> text.txt  
$ cat text.txt  
aaabbb1  
abab2  
noone2  
class1  
alien2
```

Ejemplo 1:

El primer ejemplo es una combinación de búsqueda a través del fichero sin y con expresiones regulares.

Para entender completamente las expresiones regulares es muy importante mostrar la diferencia. El primer comando busca la cadena exacta, en cualquier parte de la línea, mientras que el segundo busca conjuntos de caracteres que contengan cualquiera de los caracteres entre corchetes, por lo que los resultados de los comandos son diferentes.

```
$ grep "ab" text.txt
aaabb1
abab2
$ grep "[ab]" text.txt
aaabb1
abab2
class1
alien2
```

Ejemplo 2:

El segundo grupo de ejemplos muestran el uso de los metacaracteres inicio y fin de línea. Es muy importante especificar la necesidad de colocar los 2 caracteres en el lugar correcto de la expresión.

Cuando se especifica el comienzo de la línea, el metacaracter debe estar antes de la expresión, mientras que cuando se especifica el final de la línea, el metacaracter debe estar después de la expresión

```
$ grep "^a" text.txt
```

```
aaabbb1
```

```
abab2
```

```
alien2
```

```
$ grep "2$" text.txt
```

```
abab2
```

```
noone2
```

```
alien2
```

Además de los metacaracteres explicados anteriormente, las expresiones regulares también incluyen metacaracteres que permiten la multiplicación del patrón previamente especificado:

- * Cero o más del patrón precedente
- + Uno o más del patrón precedente
- ? Cero o uno del patrón precedente

Mas ejemplos:

Para los metacaracteres, el siguiente comando busca una cadena que contenga ab, un solo carácter y uno o más de los caracteres encontrados previamente.

El resultado muestra que grep encontró la cadena aaabbb1, que coincide con abbb y abab2. Dado que el carácter + es un carácter de expresión regular extendida, necesitamos pasar la opción -E al comando grep.

```
$ grep -E "ab.+\" text.txt
aaabbb1
abab2
```

Analicemos cada parte del comando:

- grep: Este comando se utiliza para buscar líneas en archivos de texto plano que coincidan con un patrón determinado.
- -E: Esta opción indica a grep que el patrón que se va a buscar es una expresión regular.
- "ab.+": La expresión regular que se va a buscar. En este caso, busca líneas que:
 - Comienzan con la letra "a".
 - Le sigue la letra "b".
 - Le sigue cualquier carácter (.), una o más veces (+).
- text.txt: El nombre del archivo donde se realizará la búsqueda.

La mayoría de los metacaracteres son muy fáciles de entender, pero pueden volverse complicados cuando se usan por primera vez. Los ejemplos anteriores representan una pequeña parte de la funcionalidad de las expresiones regulares. Intente utilizar todos los metacaracteres de las tablas anteriores para entender sus funcionalidades.

Actividad 12. Expresiones Regulares

Crear una carpeta con nombre act12 y guardar en dicha carpeta las soluciones a las preguntas que aparecen en las siguientes páginas, puede ser Word, txt, powerpoint o pdf. Una vez finalizado comprimir todo en un archivo zip.

1- Basado en el contenido del fichero tel2.txt, replicar los ejercicios que se muestran en las siguientes páginas:

```
$ cat tel2.txt
```

Méndez Roca, Gisela|calle Ruiñor|28023|Madrid|915351478

Ruiz del Castillo, Marcos|calle Balmes|08020|Barcelona|932282177

Hernández Darín, Alberto|plaza mayor|13190|Corral de Calatrava|
926448829

Gómez Bádenas, Josefina|calle Sagasta|13190|Corral de Calatrava|
926443602

Martínez Parra, Marta|calle de la Santa Trinidad|38870|La Calera|
984122119

Expósito Heredia, Pedro|calle del castillo|38870|La Calera|
984122369

Búsqueda de la cadena "calatrava" en mayúsculas o minúsculas:

```
$ grep -i 'calatrava' tel2.txt
```

```
Hernández Darín, Alberto|plaza mayor|13190|Corral de  
Calatrava|926448829
```

```
Gómez Bádenas, Josefina|calle Sagasta|13190|Corral de  
Calatrava|926443602
```

```
$
```

Búsqueda de las líneas que comiencen por la letra 'G':

```
$ grep '^G' tel2.txt
```

```
Gómez Bádenas, Josefina|calle Sagasta|13190|Corral de  
Calatrava|926443602
```

```
$
```

Búsqueda de las líneas que terminen por 9:

```
$ grep '9$' tel2.txt
```

```
Hernández Darín, Alberto|plaza mayor|13190|Corral de  
Calatrava|926448829
```

```
Martínez Parra, Marta|calle de la Santa Trinidad|38870|  
La Calera|984122119
```

```
Expósito Heredia, Pedro|calle del castillo|38870|  
La Calera|984122369
```

```
$
```

Búsqueda de las líneas que contengan dos ocurrencias sucesivas de la letra 'r':

```
$ grep 'rr' tel2.txt
```

Hernández Darín, Alberto|plaza mayor|13190|Corral de
Calatrava|926448829

Gómez Bádenas, Josefina|calle Sagasta|13190|Corral de
Calatrava|926443602

Martínez Parra, Marta|calle de la Santa Trinidad|38870|
La Calera|984122119

Mismo ejemplo usando una ERb:

```
$ grep 'r\{2\}' tel2.txt
Hernández Darín, Alberto|plaza mayor|13190|Corral de Calatrava|
926448829
Gómez Bádenas, Josefina|calle Sagasta|13190|Corral de Calatrava|
926443602
Martínez Parra, Marta|calle de la Santa Trinidad|38870|
La Calera|984122119
```

Mismo ejemplo usando una ERe:

```
$ grep -E 'r{2}' tel2.txt
Hernández Darín, Alberto|plaza mayor|13190|Corral de
Calatrava|926448829
Gómez Bádenas, Josefina|calle Sagasta|13190|Corral de
Calatrava|926443602
Martínez Parra, Marta|calle de la Santa Trinidad|38870|
La Calera|984122119
$
```

Mostrar las líneas que contengan las cadenas calatrava o madrid (sin diferenciar mayúsculas y minúsculas):

```
$ grep -iE 'Calatrava|madrid' tel2.txt
Méndez Roca, Gisela|calle Ruiseñor|28023|Madrid|915351478
Hernández Darín, Alberto|plaza mayor|13190|Corral de Calatrava|
926448829
Gómez Bádenas, Josefina|calle Sagasta|13190|Corral de Calatrava|
926443602
$
```

2. Ejercicios guiados

Usando `grep` y el fichero `/usr/share/hunspell/en_US.dic`, busque las líneas que coincidan con los siguientes criterios:

1. Todas las líneas que contengan la palabra `cat` en cualquier parte de la línea

2. Todas las líneas que no contengan ninguno de los siguientes caracteres: `sawgtfixkgts`.

3. Todas las líneas que comiencen con 3 letras cualesquieras y la palabra `dig`.

4. Todas las líneas que terminen al menos en una `e`.

5. Todas las líneas que contengan una de las siguientes palabras: `org`, `kay` o `tuna`.

6. Cantidad de líneas que comiencen con una o ninguna `c` seguida de la cadena `ati`.

Respuestas a los ejercicios guiados

Usando `grep` y el fichero `/usr/share/hunspell/en_US.dic`, busque las líneas que coincidan con los siguientes criterios:

1. Todas las líneas que contengan la palabra `cat` en cualquier parte de la línea

```
$ grep "cat" /usr/share/hunspell/en_US.dic
Alcatraz/M
Decatur/M
Hecate/M
...
```

2. Todas las líneas que no contengan ninguno de los siguientes caracteres: `sawgtfixkgs`.

```
$ grep -v "[sawgtfixk]" /usr/share/hunspell/en_US.dic
49269
0/nm
1/n1
2/nm
2nd/p
3/nm
3rd/p
4/nm
5/nm
6/nm
7/nm
8/nm
...
```

3. Todas las líneas que comiencen con 3 letras cualesquieras y la palabra dig.

```
$ grep "^.{3}dig" /usr/share/hunspell/en_US.dic
cardigan/SM
condign
predigest/GDS
...
```

4. Todas las líneas que terminen al menos en una e.

```
$ grep -E "e+$" /usr/share/hunspell/en_US.dic
Anglicize
Anglophobe
Anthropocene
...
```

5. Todas las líneas que contengan una de las siguientes palabras: org, kay o tuna.

```
$ grep -E "org|kay|tuna" /usr/share/hunspell/en_US.dic
Borg/SM
George/MS
Tokay/M
fortunate/UY
...
```

6. Cantidad de líneas que comiencen con una o ninguna c seguida de la cadena ati.

```
$ grep -cE "^c?ati" /usr/share/hunspell/en_US.dic  
3
```

3. Mas ejercicios

Contenido del archivo.txt (use otro nombre):

Esta es una frase.

Otra frase con la palabra "clave".

Y otra frase más.

- a) Contar líneas que contienen una palabra específica.

Comando: grep -c "clave" archivo.txt

Salida: 1

Explicación:

grep -c: Cuenta las líneas que coinciden con la expresión regular.

"clave": La palabra que queremos buscar.

archivo.txt: El archivo donde se realiza la búsqueda.

b. Buscar nombres de usuario en un archivo de contraseñas

Comando: **grep -E "^[^:]+:" /etc/passwd**

Salida:

```
root  
daemon  
...  
...
```

Explicación:

grep -E: Indica que se utiliza una expresión regular.

^: Coincide con el inicio de la línea.

[^:]+: Coincide con cualquier carácter que no sea : (uno o más).

: Coincide con el carácter :.

/etc/passwd: El archivo donde se realiza la búsqueda.

c. Filtrar direcciones de correo electrónico

- Archivo: correos.txt (cambiar nombre)

Contenido del archivo:

Nombre: Juan Pérez

Correo: juan.perez@ejemplo.com

Nombre: Ana García

Correo: anagarcia@otroejemplo.com

- Comando:

```
grep -E "[^@]+@[^@]+\.[^@]+" correos.txt
```

- Salida:

juan.perez@ejemplo.com

anagarcia@otroejemplo.com

Explicación:

- grep -E: Indica que se utiliza una expresión regular.
- [^@]+: Coincide con cualquier carácter que no sea @ (uno o más).
- @: Coincide con el carácter @.
- [^@]+: Coincide con cualquier carácter que no sea @ (uno o más).
- \.: Coincide con el carácter . (punto).
- [^@]+: Coincide con cualquier carácter que no sea @ (uno o más).

d. Reemplazar texto en un archivo

- Archivo: texto.txt (cambiar nombre)

Contenido del archivo:

Esta frase contiene la palabra "ejemplo".

Y otra frase con la palabra "ejemplo".

- Comando:

```
sed -i "s/ejemplo/reemplazo/g" texto.txt
```

- Salida:

Esta frase contiene la palabra "reemplazo".

Y otra frase con la palabra "reemplazo".

Explicación:

- sed -i: Edita el archivo en línea (sin necesidad de crear un archivo temporal).
- "s/ejemplo/reemplazo/g": La expresión de sed.
- s: Indica que se va a realizar una sustitución.
- /ejemplo/: La palabra que queremos reemplazar.
- /reemplazo/: La palabra que queremos usar como reemplazo.
- g: Indica que se reemplazan todas las apariciones, no solo la primera.

e. Eliminar líneas vacías de un archivo

- Archivo: archivo.txt (cambiar nombre)

Contenido del archivo (incluir las líneas en blanco):

Esta frase tiene contenido.

Y esta también.

Pero esta está vacía.

- Comando:

```
sed '/^$/d' archivo.txt
```

- Salida:

Esta frase tiene contenido.

Y esta también.

Explicación:

- sed: Edita el archivo en línea (sin necesidad de crear un archivo temporal).
- '/^\$/d': La expresión de sed.
- /^\$/: Coincide con una línea que está vacía (no tiene caracteres).
- d: Elimina la línea que coincide con la expresión.

f. Extraer números de teléfono de un archivo

- Archivo: telefonos.txt (cambiar nombre)

Contenido del archivo:

Nombre: Ana García

Teléfono: 1234567890

Nombre: Juan Pérez

Teléfono: +54 9 11 1234-5678

- Comando:

```
grep -E "[0-9+() -]+[0-9]" telefonos.txt
```

- Salida:

1234567890

+54 9 11 1234-5678

Explicación:

- grep -E: Indica que se utiliza una expresión regular.
- [0-9+() -]+: Coincide con cualquier número del 0 al 9, +, (,), - (uno o más).
- [0-9]: Coincide con cualquier número del 0 al 9 (uno).

g. Extraer fechas de un archivo de texto

- Archivo: fechas.txt (cambiar nombre)

Contenido del archivo:

Fecha de nacimiento: 14/05/1980

Fecha de vencimiento: 03-08-2024

Próximo evento: 2023-12-31

Cita médica: 2024-04-09

- Comando:

```
grep -E "(0[1-9]|1[2][0-9]|3[01])[- /.](0[1-9]|1[012])[- /.]([0-9]{4})" fechas.txt
```

- Salida:

14/05/1980

03-08-2024

2023-12-31

2024-04-09

Explicación:

- grep -E: Indica que se utiliza una expresión regular.
- (0[1-9]|[12][0-9]|3[01]): Coincide con el día del mes (entre 1 y 31).
- [- ./]: Coincide con un guion, barra o punto.
- (0[1-9]|1[012]): Coincide con el mes del año (entre 1 y 12).
- [- ./]: Coincide con un guion, barra o punto.
- ([0-9]{4}): Coincide con el año (4 dígitos).

h. Extraer nombres de archivos con una extensión específica

- Archivo: archivos.txt (cambiar nombre)

Contenido del archivo:

imagen.jpg

documento.pdf

musica.mp3

codigo.py

archivo.txt

otro_archivo.txt

- Comando:

```
grep -Eo "[^ ]+\.txt" archivos.txt
```

- Salida:

archivo.txt

otro_archivo.txt

Explicación:

- grep -Eo: Indica que se utiliza una expresión regular y que solo se debe mostrar la parte que coincide.
- [^]+: Coincide con cualquier carácter que no sea un espacio (uno o más).
- \.: Coincide con el carácter . (punto).
- txt: Coincide con la extensión .txt.

4. Propuestos. Partiendo del fichero **historia.txt** responde las siguientes preguntas:

1. Muestra todas las líneas que contengan la palabra “España”
2. Recupera el comando anterior y mediante un pipe junto a ‘wc’ comprueba que la palabra “España” aparece en 22 líneas.
3. Muestra todas las líneas que contengan primero la cadena “España” y luego la cadena “guerra”
4. Recupera el comando anterior y por medio de un pipe y de ‘wc’ comprueba que la expresión regular aparece solo en una línea.
5. Muestra las líneas que contengan primero la cadena “guerra” y luego la cadena “España”
6. Por medio de una tubería , partiendo del comando anterior comprueba que aparece en dos líneas con la ayuda de ‘wc’
7. Muestra todas las líneas que contengan las cadenas “Francia” o “Italia”.
8. Recuperando el comando anterior con la ayuda de un pipe y de ‘wc’ comprueba que aparece en 6 líneas.

4. Propuestos. Partiendo del fichero **historia.txt** responde las siguientes preguntas:

9. Muestra las líneas que contengan ambas cadenas “España” y “guerra” en cualquier orden (puedes usar pipes)
10. Por medio de una tubería y con ‘wc’ comprueba que aparece en tres líneas
11. Muestras las líneas que contengan la cadena “guerra” pero no “España” (la opción grep –v hace búsquedas inversas, es decir, todas las líneas que no contengan la expresión regular especificada. Investigue.
12. Por medio de un pipe y con ‘wc’ comprueba que la expresión buscada aparece en 2 líneas.
13. Muestra las líneas que contengan palabras de cinco letras terminadas en ‘ia’ (recordar que \b representa el separador de palabras)
14. Por medio de un pipe y ‘wc’ comprueba que aparecen en dos líneas.
15. Muestra las líneas que contengan números de cuatro cifras.
16. Por medio de un pipe y ‘wc’ comprueba que aparecen en 19 líneas.
17. Muestra las líneas que empiecen con ‘a’ (el carácter de principio de línea en expresiones regulares es ‘^’)
18. Muestra con un pipe y con ‘wc’ que aparecen en 6 líneas

4. Propuestos. Partiendo del fichero **historia.txt** responde las siguientes preguntas:

19. Muestra las líneas que terminen con ‘s’. (el carácter para fin de línea en expresiones regulares es \$). Además, por medio de un pipe y con ‘wc’ comprueba que son 20 líneas
20. Muestra las líneas que comiencen con la cadena ‘la’. Además, por medio de un pipe y con ‘wc’ comprueba que son 5 líneas.
21. Muestra las líneas que terminen con la cadena ‘as’. Además, por medio de un pipe y con ‘wc’ comprueba que son 6 líneas.
22. Muestra las líneas que comiencen y terminen con ‘s’. Además, por medio de un pipe y con ‘wc’ comprueba que ninguna línea cumple el requisito.
23. Muestra las líneas que empiecen con ‘a’ o ‘A’ y terminen con ‘s’ o ‘S’. Además, por medio de un pipe y con ‘wc’ comprueba que solo una línea cumple el requisito.
24. Muestra las líneas que contengan fechas relativas al siglo XX (puedes usar la clase `[:digit:]`). Además, por medio de un pipe y con ‘wc’ comprueba que son 12 líneas.
25. Muestra las líneas que contengan alguna palabra de 15 letras (no de más, así que deberás usar ‘\b’; puedes usar la clase `[:alpha:]`), que es equivalente al rango [a-zA-Z]. Además, por medio de un pipe y ‘wc’ comprueba que son 3 líneas (no 6)

Crear un script a partir de una serie de comandos

¿Qué es un script?

Un script en Linux es un archivo de texto que contiene una serie de comandos que se ejecutan secuencialmente por el intérprete de comandos, también conocido como shell. Estos comandos pueden ser cualquier cosa que se pueda escribir en la línea de comandos, como:

- Comandos básicos: ls, cd, mkdir, rm, cp, mv, etc.
- Comandos más complejos: grep, find, awk, sed, sort, etc.
- Funciones personalizadas: Puedes crear tus propias funciones para reutilizar código o realizar tareas complejas.

Los scripts se utilizan para automatizar tareas repetitivas, lo que permite ahorrar tiempo y esfuerzo. Algunas ventajas son:

Automatización: Ahorran tiempo y esfuerzo al automatizar tareas repetitivas.

Eficiencia: Permiten realizar tareas complejas de forma rápida y sencilla.

Precisión: Reducen el riesgo de errores al ejecutar comandos manualmente.

Flexibilidad: Se pueden personalizar para adaptarse a tus necesidades específicas.

Scripts en Ciencia de Datos

Los científicos de datos utilizan scripts para automatizar tareas repetitivas y tediosas, lo que les permite ahorrar tiempo y centrarse en el análisis de datos. A continuación, se enumeran algunos de los casos de uso más comunes:

1. Limpieza y preprocessamiento de datos:

- Los scripts se pueden usar para cargar datos de diferentes fuentes, como archivos CSV, bases de datos o APIs.
- Se pueden usar para limpiar datos, eliminar valores nulos, corregir errores y normalizar variables.
- Se pueden usar para transformar datos, como crear nuevas variables, calcular características y agrupar datos.

Scripts en Ciencia de Datos

2. Análisis de datos:

- Los scripts se pueden usar para ejecutar análisis estadísticos, como pruebas de hipótesis, regresión lineal y análisis de varianza.
- Se pueden usar para crear visualizaciones de datos, como gráficos, diagramas y mapas.
- Se pueden usar para construir modelos de aprendizaje automático, como regresión lineal, árboles de decisión y redes neuronales.

Scripts en Ciencia de Datos

3. Implementación de modelos:

- Los scripts se pueden usar para implementar modelos de aprendizaje automático en producción.
- Se pueden usar para crear APIs que permitan a otros usuarios acceder a los modelos.
- Se pueden usar para automatizar la generación de informes y alertas.

Scripts en Ciencia de Datos

4. Gestión de proyectos:

- Los scripts se pueden usar para automatizar tareas de gestión de proyectos, como la creación de informes de progreso, el seguimiento de tareas y la gestión de versiones.
- Se pueden usar para crear flujos de trabajo que combinen diferentes herramientas y scripts.

Cómo crear un script

Hasta ahora hemos aprendido a ejecutar comandos desde la shell, pero también podemos introducir comandos en un archivo, y luego establecer que ese archivo sea ejecutable. Cuando este se ejecuta, los comandos también se ejecutarán uno tras otro. Estos archivos se llaman scripts, y son una herramienta absolutamente crucial para cualquier persona que desea automatizar tareas o procesos.

Imprimiendo salidas (Printing Output)

Comencemos construyendo un ejemplo con el comando **echo** el cual imprime un argumento en la salida estándar.

```
$ echo "Hello World!"  
Hello World!
```

Ahora, usaremos la redirección de archivos para enviar este comando a un nuevo archivo llamado **new_script**.

Imprimiendo salidas (Printing Output)

```
$ echo 'echo "Hello World!"' > new_script
$ cat new_script
echo "Hello World!"
```

El archivo `new_script` contiene ahora el mismo comando que antes.

Cómo ejecutar un script

Vamos a demostrar algunos de los pasos necesarios para que este archivo se ejecute de la forma en que esperamos, el primer pensamiento de un usuario podría ser simplemente escribir el nombre del script. La forma en que ellos podrían escribir el nombre de cualquier otro comando:

```
$ new_script  
/bin/bash: new_script: command not found
```

Podemos asumir con seguridad que new_script existe en nuestra ubicación actual, pero note que el mensaje de error no nos está diciendo que el archivo no existe, sino que nos está diciendo que el comando no existe.

Comandos y PATH

Cuando escribimos el comando **ls** en la shell, por ejemplo, estamos ejecutando un archivo llamado ls que existe en nuestro sistema de archivos:

```
$ which ls  
/bin/ls
```

Rápidamente se volvería tedioso escribir la ruta absoluta de ls cada vez que deseemos ver el contenido de un directorio, así que Bash tiene una variable de entorno (environment) que contiene todos los directorios en los que podemos encontrar los comandos que deseamos ejecutar.

El comando which en Linux se utiliza para encontrar la ubicación del archivo ejecutable de un comando o programa. En otras palabras, te dice dónde se encuentra en el sistema de archivos el comando que estás intentando ejecutar.

Comandos y PATH

Cada una de estas ubicaciones es donde el shell espera encontrar un comando, delimitado con dos puntos lo está.

```
$ echo $PATH  
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
```

La shell buscará new_script en cada uno de estos directorios, pero no lo encontrará y por lo tanto lanzará el error que vimos antes.

Comandos y PATH

Hay tres soluciones para este problema: podemos **mover** new_script a uno de los directorios PATH, podemos **añadir** nuestro directorio actual a PATH, o podemos **cambiar** la forma en que intentamos llamar al script, esta última solución es la más sencilla, simplemente requiere que especifiquemos la ubicación actual cuando llamemos al script usando la barra de puntos (./).

```
$ ./new_script  
/bin/bash: ./new_script: Permission denied
```

El mensaje de error ha cambiado, lo que indica que hemos hecho algunos progresos.

Ejecutar Permisos

La primera indagación que un usuario debe hacer en este caso es usar ls -l para mirar el archivo:

```
$ ls -l new_script  
-rw-rw-r-- 1 user user 20 Apr 30 12:12 new_script
```

Podemos ver que los permisos para este archivo están configurados para que ningún usuario tenga permisos de ejecución. Por lo tanto:

```
$ chmod +x new_script  
$ ls -l new_script  
-rwxrwxr-x 1 user user 20 Apr 30 12:12 new_script
```

Ejecutar Permisos

Este comando ha dado permisos de ejecución a todos los usuarios, tenga en cuenta que esto puede ser un riesgo para la seguridad, pero por ahora es un nivel aceptable de permisos.

```
$ ./new_script  
Hello World!
```

Ahora podemos ejecutar nuestro script.

Definición del intérprete

Como hemos demostrado, hemos podido simplemente introducir texto en un archivo, configurarlo como ejecutable y ejecutarlo. El new_script es funcionalmente un archivo de texto normal, pero logramos que sea interpretado por Bash, pero ¿qué pasa si está escrito en Python?

Es muy recomendable especificar el tipo de intérprete que queremos usar en la primera línea de un script, este se llama **bang line** o más conocido como **shebang**; e indica al sistema cómo queremos que se execute este archivo. Como estamos aprendiendo Bash, estaremos usando la ruta absoluta a nuestro ejecutable Bash, una vez más usando **which**:

```
$ which bash  
/bin/bash
```

Definición del intérprete

Nuestro **shebang** comienza con un signo de hash (#) y un signo de exclamación de cierre (!), seguido de la ruta absoluta que vimos antes (/bin/bash). Ahora abrimos **new_script** en un editor de texto e insertemos el **shebang**, y aprovechemos la oportunidad para insertar un comentario en nuestro script, ya que los comentarios son ignorados por el intérprete y están escritos para el beneficio de otros usuarios que deseen entender su script.

```
#!/bin/bash

# Este es nuestro primer comentario. También es una buena práctica documentar todos los scripts.

echo "Hello World!"
```

Definición del intérprete

También haremos un cambio adicional al nombre del archivo: guardaremos este archivo como `new_script.sh`. El sufijo del archivo **.sh** no cambia la ejecución del archivo de ninguna manera, es una convención que los scripts bash sean etiquetados con **.sh** o **.bash** para identificarlos fácilmente, así como los scripts Python son usualmente identificados con el sufijo **.py**.

Variables

Las variables son una parte importante de cualquier lenguaje de programación y **Bash** no es diferente, cuando se inicia una nueva sesión desde la terminal, la shell ya establece algunas variables, como por ejemplo la variable PATH, a la que llamamos **variables de entorno**, ya que suelen definir las características de nuestro entorno de shell, por lo que se pueden modificar y añadir variables de entorno, pero por ahora vamos a centrarnos en la configuración de variables dentro de nuestro script.

Modificaremos nuestro script para que se vea así:

```
#!/bin/bash

# Este es nuestro primer comentario. También es una buena práctica comentar todos los
scripts.

username=Carol

echo "Hello $username!"
```

En este caso, hemos creado una variable llamada `username` y le hemos asignado el valor de Carol. Tenga en cuenta que no hay espacios entre el nombre de la variable, el signo igual o el valor asignado.

Modificaremos nuestro script para que se vea así:

En la siguiente línea, hemos utilizado el comando echo con la variable, pero hay un signo de dólar (\$) adelante del nombre de la variable, esto es importante, ya que indica a la shell que queremos tratar el nombre de usuario como una variable y no sólo como una palabra normal. Al introducir \$username en nuestro comando, indicamos que queremos realizar una sustitución, reemplazando el nombre de una variable por el valor asignado a esa variable.

Ejecutando el nuevo script, obtenemos esta salida:

```
$ ./new_script.sh  
Hello Carol!
```

Ejemplo básico de un script para crear un directorio

- Vamos a crear una carpeta (desde la terminal) dentro del directorio *practica* que se llame: *ej_script*
- Con ayuda del editor nano crear un script que contenga este código:

```
#!/bin/bash

# Crear la carpeta principal "Proyecto"
mkdir -p Proyecto

# Crear las subcarpetas dentro de "Proyecto"
for i in {1..5}; do
    mkdir -p Proyecto/subcarpeta_$i

    # Crear los archivos dentro de cada subcarpeta
    touch Proyecto/subcarpeta_$i/archivo.py
    touch Proyecto/subcarpeta_$i/archivo.ipynb
    touch Proyecto/subcarpeta_$i/documentación.docx
    touch Proyecto/subcarpeta_$i/readme.md
    touch Proyecto/subcarpeta_$i/.git

done

echo "Estructura del proyecto creada exitosamente en la carpeta 'Proyecto'."
```

Ejemplo básico de un script para crear un directorio

- Guardar el script con nombre carpetas.sh y salir del editor nano.
- Una vez en el terminal, verificar los permisos del archivo carpetas.sh
- Dar los permisos de ejecución con chmod
- Ejecutar el script escribiendo: ./carpetas.sh

La estructura de ese sistema de ficheros es esta:

```
Proyecto
└── subcarpeta_1
    ├── archivo_1.py
    ├── archivo_1.ipynb
    ├── documentación.docx
    └── readme.md
        └── .git
            └── subcarpeta_2
                ├── archivo_2.py
                ├── archivo_2.ipynb
                ├── documentación.docx
                └── readme.md
                    └── .git
                        └── subcarpeta_3
                            ├── archivo_3.py
                            ├── archivo_3.ipynb
                            ├── documentación.docx
                            └── readme.md
                                └── .git
                                    └── subcarpeta_4
                                        ├── archivo_4.py
                                        ├── archivo_4.ipynb
                                        ├── documentación.docx
                                        └── readme.md
                                            └── .git
                                                └── subcarpeta_5
                                                    ├── archivo_5.py
                                                    ├── archivo_5.ipynb
                                                    ├── documentación.docx
                                                    └── readme.md
                                                        └── .git
```

Para ver la estructura se puede escribir en el terminal el comando: **tree -a**.





