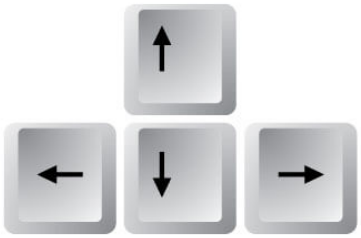
1.- Probando el terminal

Escribimos unos caracteres y se recibe un mensaje de error diciendo que no encontró la orden.

```
alumno@administrador-20VE:~$ kdkjflajfks
kdkjflajfks: no se encontró la orden
alumno@administrador-20VE:~$
```



- Cuando se presiona la **tecla de flecha hacia arriba** lo que pasa es que regresa el comando anterior que en este caso es “kdkjflajfks”.
- Si presionamos la **tecla de la flecha hacia abajo** lo que pasa es que tenemos la línea de comando en blanco.
- Por último, las **flechas izquierda y derecha**, ayuda a colocar el cursor en cualquier parte de la línea de comando, permitiendo la corrección de errores.

2.- Comandos de Navegación:

pwd: Permite imprimir directorio de trabajo. En este caso se llama /home/alumno.

```
alumno@administrador-20VE:~$ pwd
/home/alumno
```

ls: Este comando ayuda a enumerar las listar archivos y directorios del directorio de trabajo.

```
alumno@administrador-20VE:~$ ls
app-web     docker_ubuntu  Imágenes        nginx.yaml   snap
cliente.py  Documentos     informarmacion  Plantillas   start.sh
Descargas   Escritorio     kubectl         Público      user_space_report.txt
Dockerfile  hola.py        Música          servidor.py  Videos
```

cd: Nos permite cambiar directorio.Para ello se escribe primero el comando **cd** seguido con un la ruta del directorio donde se desea trabajar.

Existe dos tipos de nombres rutas:

- Nombres de rutas absoluto
- Nombre de rutas relativos

Nombre de rutas absoluto: Comienza con el directorio raíz y sigue el árbol rama por rama hasta completar la ruta al directorio o archivo deseado.

EJEMPLO: La ruta del directorio es **/usr/bin**. Esto significa que desde el directorio raíz que se representa con **/** hay un directorio llamado **"usr"** que contiene un directorio llamado **"bin".** Luego vemos el directorio donde estamos por el comando **pwd** y se observa que estamos en el directorio **/usr/bin.** Por último, imprimimos lo que contiene este directorio con el comando **ls.**

```
alumno@administrador-20VE:~$ cd /usr/bin
alumno@administrador-20VE:/usr/bin$ pwd
/usr/bin
alumno@administrador-20VE:/usr/bin$ ls
'['                                 mpif77
 aa-enabled                         mpif77.openmpi
 aa-exec                            mpif90
 aa-features-abi                    mpif90.openmpi
 aconnect                           mpifort
 add-apt-repository                 mpifort.openmpi
 addpart                            mpijavac
 addr2line                          mpijavac.pl
 aggregate_profile                  mpirun
 airscan-discover                   mpirun.openmpi
 alsabat                            mscompress
 alsaloop                           msexpand
 alsamixer                          mt
 alsatplg                           mt-gnu
 alsaucm                            mtr
 amidi                              mtrace
 amixer                             mtr-packet
 antlr4                             mv
 apg                                namei
 apgbfm                             nano
 aplay                              nautilus
 aplaymidi                          nautilus-autorun-software
 apport-bug                         nawk
 apport-cli                         nc
 apport-collect                     nc.openbsd
 apport-unpack                      neqn
 appres                             netcat
 appstreamcli                       netstat
 apropos                            networkctl
 apt                                networkd-dispatcher
 apt-add-repository                 newgidmap
 apt-cache                          newgrp
 apt-cdrom                          newuidmap
 apt-config                         ngettext
```

Nombres de rutas relativas: Comienza desde el directorio raíz y conduce a su destino. Para ello, utiliza dos notaciones especiales: **“.”** se refiere al directorio de trabajo y **“..”** se refiere al directorio principal del directorio de trabajo.

Cómo funciona:

1.- Primero nos situamos en el directorio de trabajo a /usr/bin.

```
alumno@administrador-20VE:/$ cd /usr/bin
alumno@administrador-20VE:/usr/bin$ pwd
/usr/bin
alumno@administrador-20VE:/usr/bin$
```

2.- Si queremos ingresar al directorio padre del directorio de trabajo a /usr/bin, que es /usr. Existen dos forma:

Primera forma:

```
alumno@administrador-20VE:/$ cd /usr
alumno@administrador-20VE:/usr$ pwd
/usr
alumno@administrador-20VE:/usr$
```

Segunda Forma:

```
alumno@administrador-20VE:/usr/bin$ cd ..
alumno@administrador-20VE:/usr$ pwd
/usr
```

3.- Asimismo si queremos cambiar de directorio de /usr a /usr/bin. Hay dos formas:
Primera forma:

```
alumno@administrador-20VE:/$ cd /usr/bin
alumno@administrador-20VE:/usr/bin$ pwd
/usr/bin
```

Segunda forma:

```
alumno@administrador-20VE:/usr$ cd ./bin
alumno@administrador-20VE:/usr/bin$ pwd
/usr/bin
```

En la mayoría de casos podemos omitir **./bin,** lo cual haría lo mismo.

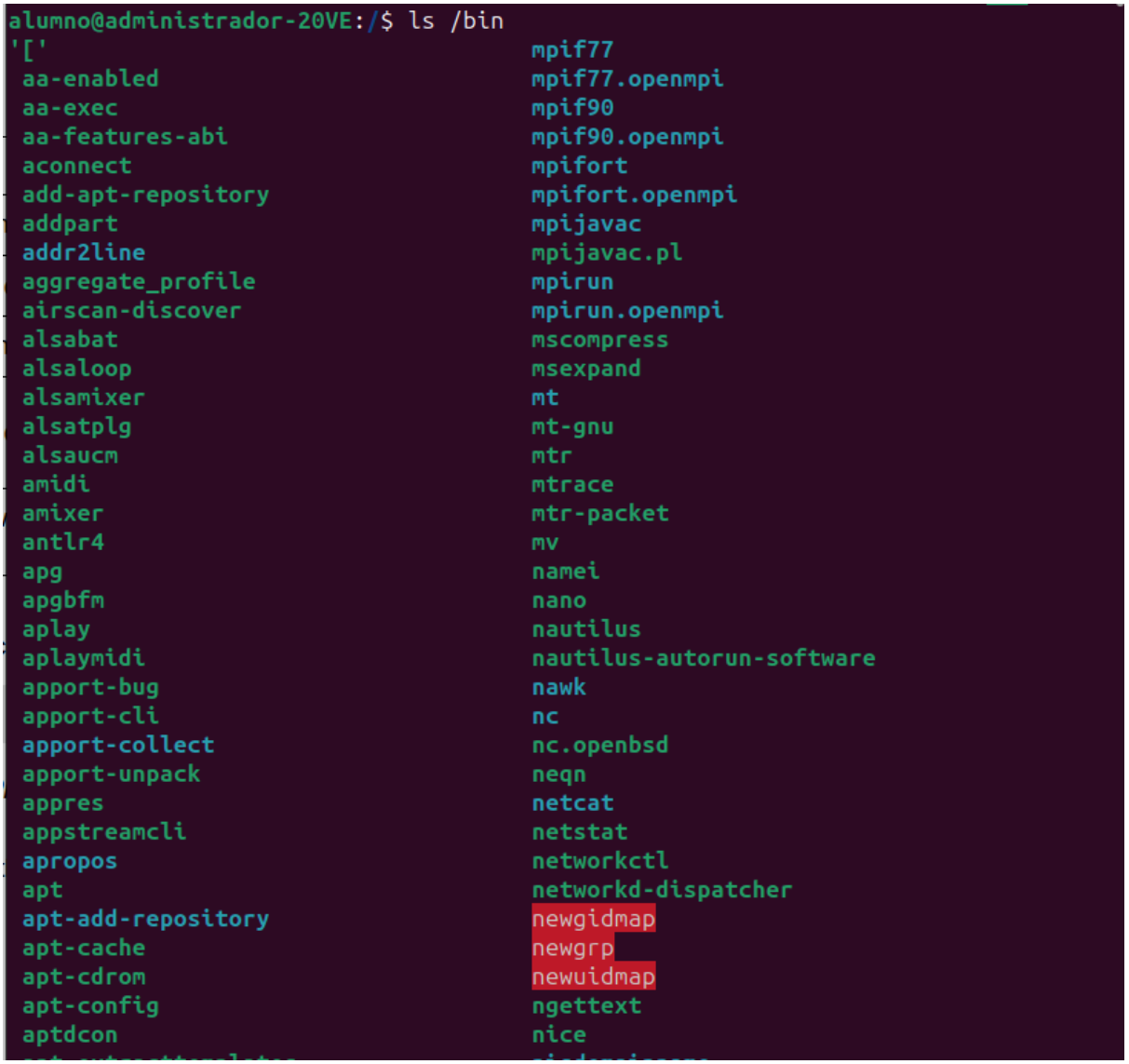
```
alumno@administrador-20VE:/usr$ cd bin
alumno@administrador-20VE:/usr/bin$
```

Diferentes manera de utilizar el comando “ls”:

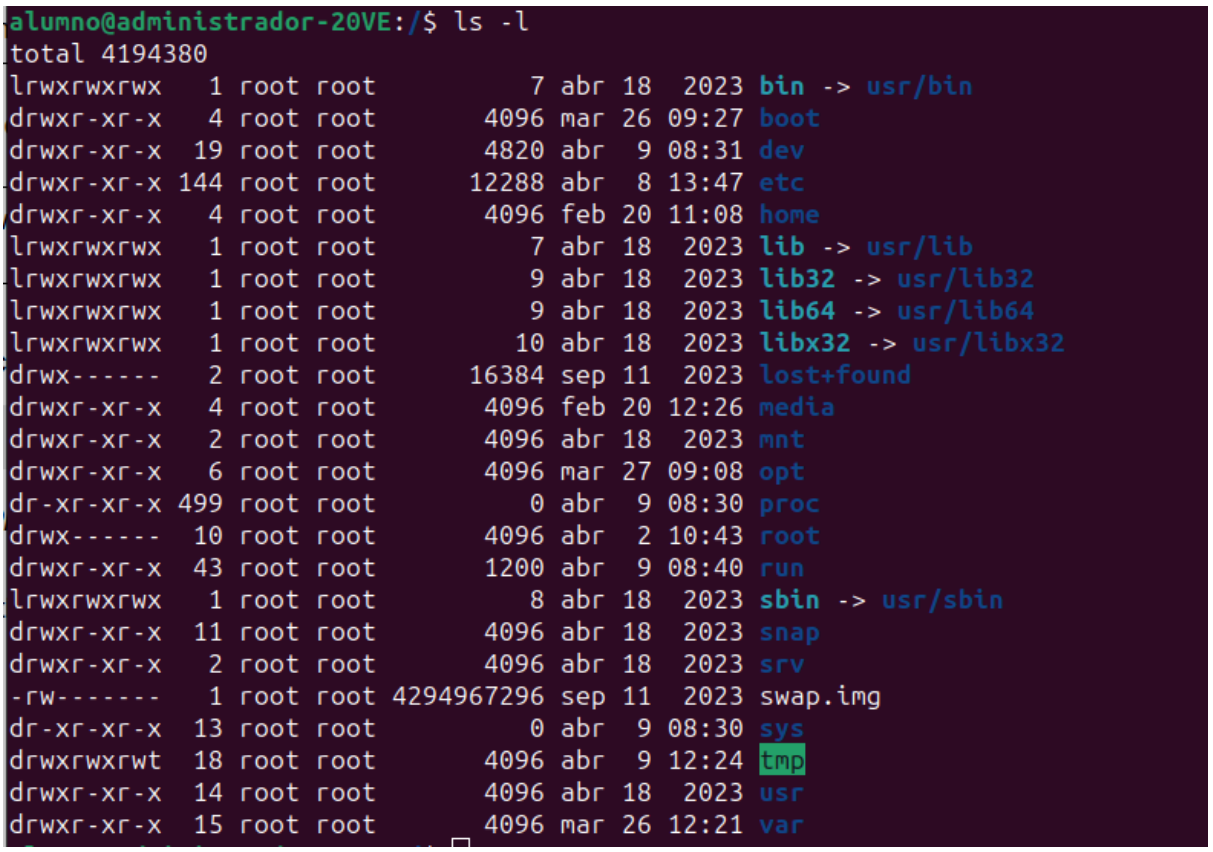
ls: Muestra la lista de los archivos en el directorio de trabajo.

```
alumno@administrador-20VE:/$ ls
bin   dev  home  lib32  libx32      media  opt   root  sbin  srv       sys  usr
boot  etc  lib   lib64  lost+found  mnt    proc  run   snap  swap.img  tmp  var
alumno@administrador-20VE:/$
```

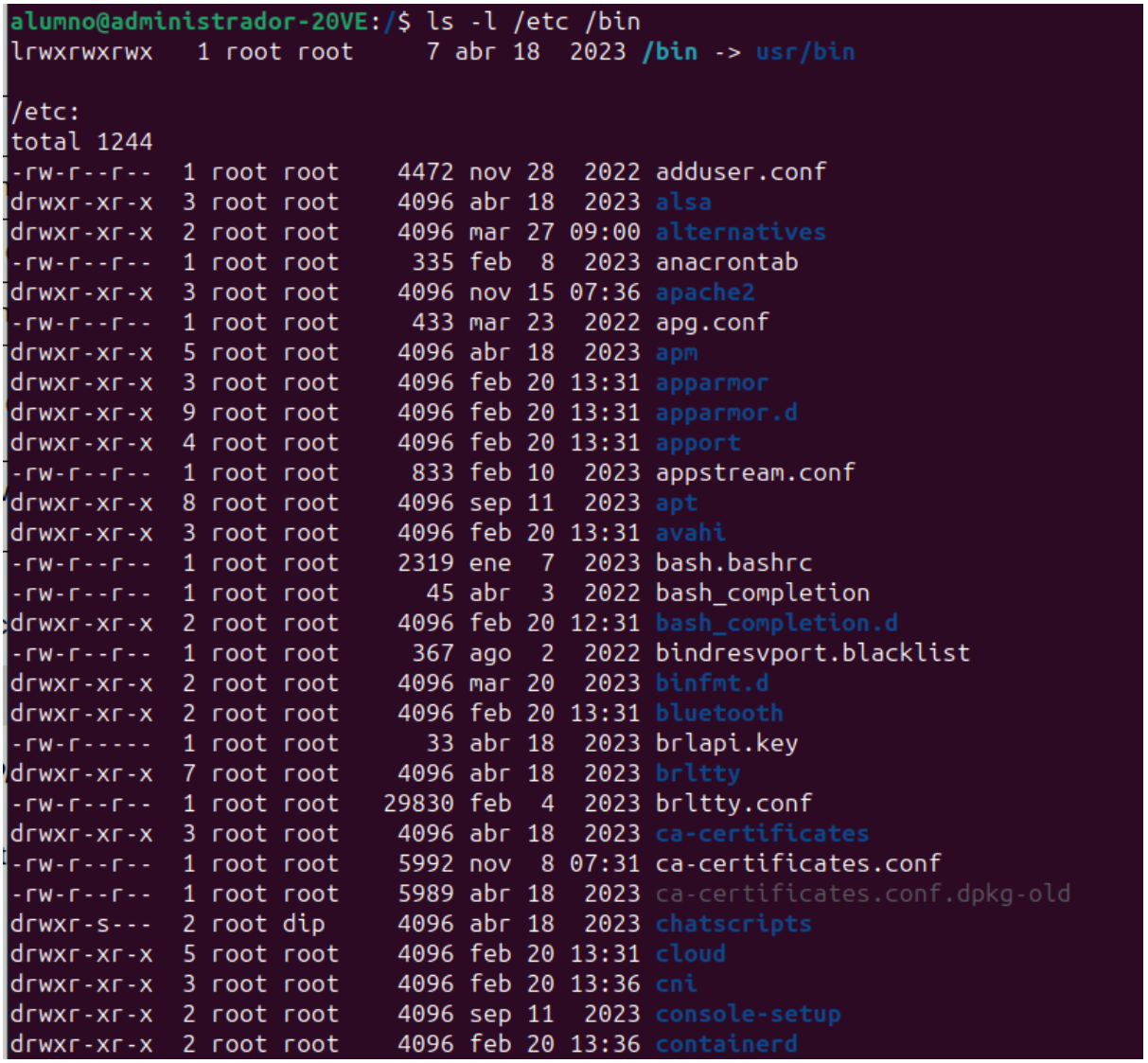
ls /bin: Enumera la lista de archivos que se encuentra en el directorio **/bin.** También, esto funciona para otros directorios que deseamos.



ls -l : Enumera los archivos del directorio de trabajo en formato largo.



ls -l /etc /bin : Enumera los archivos del directorio **/bin** y el directorio **/etc** en formato largo.



ls -la .. : Enumera todos los archivos, hasta archivos ocultos, en el directorio principal del directorio de trabajo en formato largo.

- La primera columna representa a los permisos de los archivos que puede ser de lectura, escritura y ejecución.
- La segunda columna representa el nombre del usuario propietario del archivo.
- La tercera columna representa el nombre del grupo que tiene permisos de archivo.
- La cuarta columna representa el tamaño del archivo en bytes.
- La quinta columna muestra la fecha de la última modificación.
- La sexta columna muestra el nombre del archivo.

```
alumno@administrador-20VE:/$ ls -la ..
total 4194388
drwxr-xr-x  19 root root       4096 sep 11  2023 .
drwxr-xr-x  19 root root       4096 sep 11  2023 ..
lrwxrwxrwx   1 root root          7 abr 18  2023 bin -> usr/bin
drwxr-xr-x   4 root root       4096 mar 26 09:27 boot
drwxr-xr-x  19 root root       4820 abr  9 08:31 dev
drwxr-xr-x 144 root root      12288 abr  8 13:47 etc
drwxr-xr-x   4 root root       4096 feb 20 11:08 home
lrwxrwxrwx   1 root root          7 abr 18  2023 lib -> usr/lib
lrwxrwxrwx   1 root root          9 abr 18  2023 lib32 -> usr/lib32
lrwxrwxrwx   1 root root          9 abr 18  2023 lib64 -> usr/lib64
lrwxrwxrwx   1 root root         10 abr 18  2023 libx32 -> usr/libx32
drwx------   2 root root      16384 sep 11  2023 lost+found
drwxr-xr-x   4 root root       4096 feb 20 12:26 media
drwxr-xr-x   2 root root       4096 abr 18  2023 mnt
drwxr-xr-x   6 root root       4096 mar 27 09:08 opt
dr-xr-xr-x 423 root root          0 abr  9 08:30 proc
drwx------  10 root root       4096 abr  2 10:43 root
drwxr-xr-x  43 root root       1200 abr  9 08:40 run
lrwxrwxrwx   1 root root          8 abr 18  2023 sbin -> usr/sbin
drwxr-xr-x  11 root root       4096 abr 18  2023 snap
drwxr-xr-x   2 root root       4096 abr 18  2023 srv
-rw-------   1 root root 4294967296 sep 11  2023 swap.img
dr-xr-xr-x  13 root root          0 abr  9 08:30 sys
drwxrwxrwt  18 root root       4096 abr  9 12:28 tmp
drwxr-xr-x  14 root root       4096 abr 18  2023 usr
drwxr-xr-x  15 root root       4096 mar 26 12:21 var
alumno@administrador-20VE:/$
```

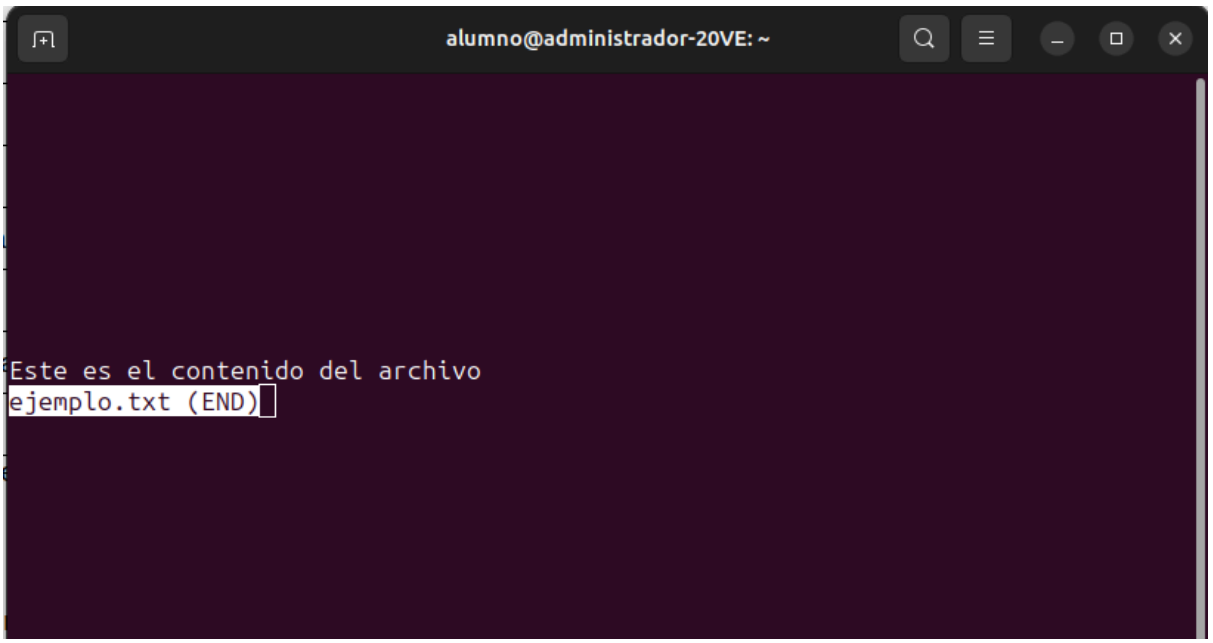
ls -a: Mostrará todos los archivos y directorios, incluyendo estos ocultos, en el directorio actual.

```
alumno@administrador-20VE:/$ ls -a
.    boot  home    lib64       media  proc  sbin  swap.img  usr
..   dev   lib     libx32      mnt    root  snap  sys       var
bin  etc   lib32   lost+found  opt    run   srv   tmp
alumno@administrador-20VE:/$
```

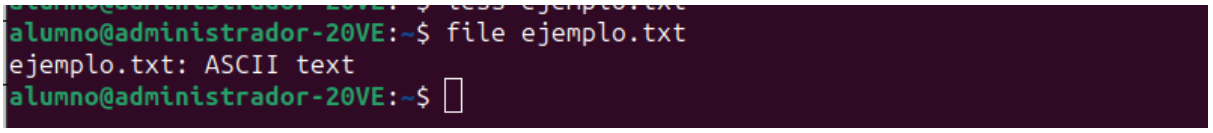
Comando “less”:

Este comando permite mostrar un archivo de texto una página a la vez.

```
alumno@administrador-20VE:~$ less ejemplo.txt
alumno@administrador-20VE:~$
```



Comando “file”: Examinará un archivo y nos dirá qué tipo de archivo es.



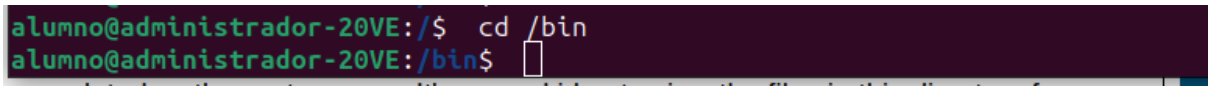
Prueba de comandos :

cd / : Nos permite ingresar al directorio raíz.

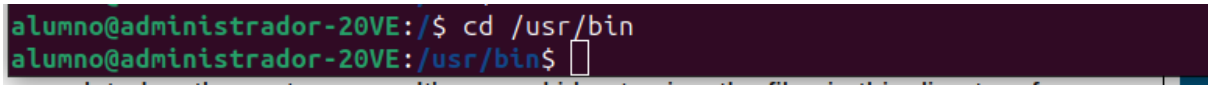
cd /boot: Nos permite ingresar al directorio **/boot** que contiene los archivos del kernel de Linux y del cargador de arranque.

cd /etc: Nos permite ingresar al directorio **/etc** que contiene los archivos de configuración del sistema.

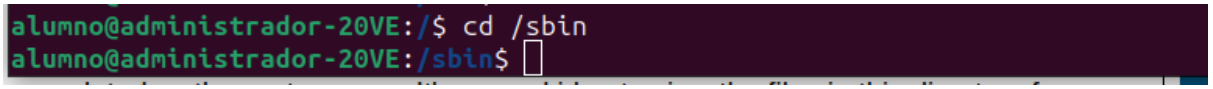
cd /bin: Nos permite ingresar al directorio **/bin** que contiene los programas esenciales que el sistema requiere para funcionar.

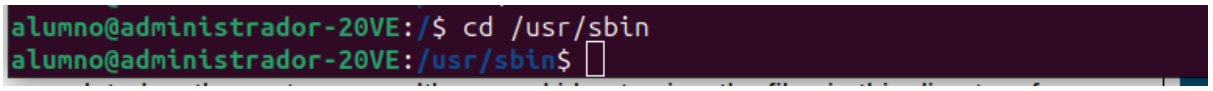


cd /usr/bin: Nos permite ingresar al directorio **/usr/bin** que contiene aplicaciones para los usuarios del sistema.

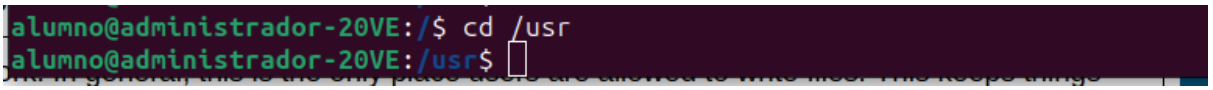


cd /sbin , cd /usr/sbin : Nos permite ingresar al directorio /**sbin** y **/usr/sbin** que contiene programas para la administración del sistema.

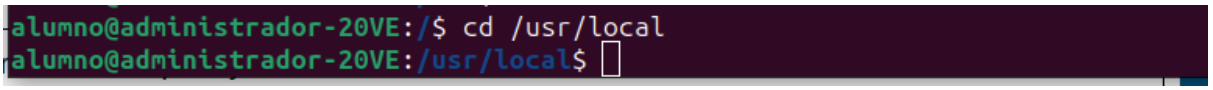




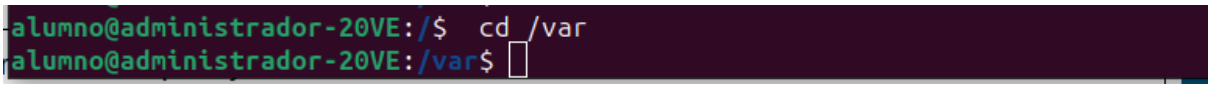
cd /usr: Nos permite ingresar al directorio **/usr** que contiene una variedad de elementos que respaldan las aplicaciones de usuario.



cd /usr/local: Nos permite ingresar al directorio **/usr/local** que contiene subdirectorios se utilizan para la instalación de software y otros archivos para su uso en la máquina local.

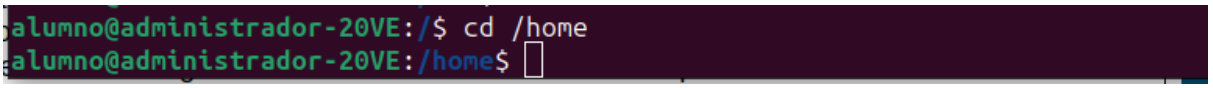


cd /var: Nos permite ingresar al directorio **/var** que contiene archivos que cambian a medida que se ejecuta el sistema.

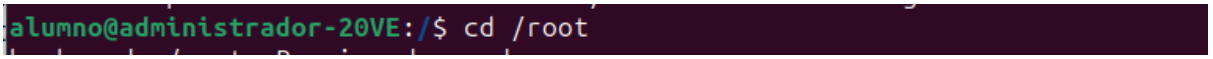


cd /lib: Nos permite ingresar al directorio **/lib** que contiene las bibliotecas compartidas.

cd /home: Nos permite ingresar al directorio **/home** que es donde los usuarios guardan su trabajo personal.



cd /root: Nos permite ingresar al directorio **/root** que es el directorio de inicio del superusuario.



cd /tmp: Nos permite ingresar al directorio **/tmp** que es un directorio en el que los programas pueden escribir sus archivos temporales.

cd /dev: Nos permite ingresar al directorio **/dev** que contiene dispositivos que están disponibles para el sistema.

```
alumno@administrador-20VE:/$ cd /dev
alumno@administrador-20VE:/dev$ ls
acpi_thermal_rel  i2c-6          nvme0      tty22  tty59      ttyS8
autofs            i2c-7          nvme0n1    tty23  tty6       ttyS9
block             i2c-8          nvme0n1p1  tty24  tty60      udmabuf
btrfs-control     i2c-9          nvme0n1p2  tty25  tty61      uhid
bus               initctl        nvme0n1p3  tty26  tty62      uinput
char              input          nvme0n1p4  tty27  tty63      urandom
console           kmsg           nvme0n1p5  tty28  tty7       userfaultfd
core              kvm            nvme0n1p6  tty29  tty8       userio
cpu               log            nvram      tty3   tty9       v4l
cpu_dma_latency   loop0          port       tty30  ttyprintk  vcs
cuse              loop1          ppp        tty31  ttyS0      vcs1
disk              loop10         psaux      tty32  ttyS1      vcs2
dma_heap          loop11         ptmx       tty33  ttyS10     vcs3
dri               loop2          pts        tty34  ttyS11     vcs4
drm_dp_aux0       loop3          random     tty35  ttyS12     vcs5
```

cd /proc: Proporciona una amplia variedad de información sobre el sistema en tiempo de ejecución.

```
alumno@administrador-20VE:/$ cd /proc
alumno@administrador-20VE:/proc$ ls
1      19160  39    5103  5582   64     768          filesystems
10     19172  390   5119  56     64601  769          fs
10020  19218  391   5143  5649   647    77           interrupts
10070  19265  392   52    5652   65     78           iomem
105    19299  4     5226  5698   65085  79           ioports
1061   193    40    5231  57     65251  791          irq
107    19349  400   5232  5714   65285  8            kallsyms
1095   199    41    5245  58     65464  80           kcore
11     2      43    5246  58107  65566  8037         keys
113    20     44    5251  5834   6574   805          key-users
12     200    45    5261  584    65751  807          kmsg
13     201    46    5276  585    6595   8072         kpagecgroup
137    202    4660  5278  58502  65987  809          kpagecount
1385   203    4682  5282  586    66208  81           kpageflags
14     204    4683  5292  59     66242  810          loadavg
140    205    4689  5293  592    66276  811          locks
```

cd /media: Nos permite ingresar al directorio **/media** que se utiliza para puntos de montaje.

```
alumno@administrador-20VE:/$ cd /media
alumno@administrador-20VE:/media$ ls
administrador  alumno
```

COMODINES:

ls *: Muestra todos los nombres de archivos.

```
alumno@administrador-20VE:~$ ls *
cliente.py  ejemplo.txt  informarmacion  nginx.yaml   start.sh
Dockerfile  hola.py      kubectl         servidor.py  user_space_report.txt

app-web:
index.html   sample_app.py  static   templates
running.txt  sample-app.sh  tempdir  venv

Descargas:
Actividad0-C8286-1.pdf
Actividad0-C8286-2.pdf
Actividad0-C8286-3.pdf
Actividad0-C8286.pdf
Actividad2-C8286.pdf
Concurrencia.ipynb
docker-desktop-4.28.0-amd64.deb
Evaluacion0-C8286-1.pdf
Evaluacion0-C8286.pdf
Evaluacion1-C8286.pdf
Evaluacion3-C8286.pdf
Introduccion.pdf
Notas-Docker.pdf
Photoresponsive_Amide-Based_Derivatives_of_Azobenz.pdf
```

ls *.txt: Muestra todos los nombres de archivos que terminan con los caracteres ".txt"

```
alumno@administrador-20VE:~$ ls *.txt
ejemplo.txt  user_space_report.txt
alumno@administrador-20VE:~$
```

grep '[[:alnum:]]' ejemplo.txt : Muestra el texto del archivo ejemplo.txt los caracteres alfanuméricos, que en este caso se subrayan de color rojo.

```
alumno@administrador-20VE:~$ grep '[[:alnum:]]' ejemplo.txt
Este es el contenido del archivo
La evolución de los procesadores ha sido impresionante.
 Desde los primeros microprocesadores de 4 bits que revolucionaron
 la informática personal en los años 70 hasta los
 procesadores actuales de 64 bits que alimentan
supercomputadoras y servidores de alto rendimiento,
 la capacidad de procesamiento ha aumentado exponencialmente.
```

grep '[[:alpha:]]' ejemplo.txt : Muestra el texto del archivo ejemplo.txt los caracteres alfabéticos, que en este caso estan de color rojo.

```
alumno@administrador-20VE:~$ grep '[[:alpha:]]' ejemplo.txt
Este es el contenido del archivo
La evolución de los procesadores ha sido impresionante.
 Desde los primeros microprocesadores de 4 bits que revolucionaron
 la informática personal en los años 70 hasta los
 procesadores actuales de 64 bits que alimentan
supercomputadoras y servidores de alto rendimiento,
 la capacidad de procesamiento ha aumentado exponencialmente.
```

grep '[[:digit:]]' ejemplo.txt : Muestra el texto del archivo ejemplo.txt los números, en este caso estan de color rojo.

```
alumno@administrador-20VE:~$ grep '[[:digit:]]' ejemplo.txt
 Desde los primeros microprocesadores de 4 bits que revolucionaron
 la informática personal en los años 70 hasta los
 procesadores actuales de 64 bits que alimentan
```

grep '[[:upper:]]' ejemplo.txt : Muestra el texto del archivo ejemplo.txt los caracteres alfabéticos Mayúsculas , en este caso estan de color rojo.

```
alumno@administrador-20VE:~$ grep '[[:upper:]]' ejemplo.txt
Este es el contenido del archivo
La evolución de los procesadores ha sido impresionante.
 Desde los primeros microprocesadores de 4 bits que revolucionaron
```

grep '[[:lower:]]' ejemplo.txt : Muestra el texto del archivo ejemplo.txt los caracteres alfabéticos Minúsculas , en este caso estan de color rojo.

```
alumno@administrador-20VE:~$ grep '[[:lower:]]' ejemplo.txt
Este es el contenido del archivo
La evolución de los procesadores ha sido impresionante.
 Desde los primeros microprocesadores de 4 bits que revolucionaron
 la informática personal en los años 70 hasta los
 procesadores actuales de 64 bits que alimentan
supercomputadoras y servidores de alto rendimiento,
 la capacidad de procesamiento ha aumentado exponencialmente.
```

ls e*: Muestra todos los nombres de archivos que comienzan con el carácter "e"

```
alumno@administrador-20VE:~$ ls e*
ejemplo.txt
```

ls e*.txt: Muestra todos los nombres de archivos que comienzan con el carácter "b" y terminan con los caracteres ".txt".

```
alumno@administrador-20VE:~$ ls e*.txt
ejemplo.txt
```

ls [abc]*: Muestra cualquier nombre de archivo que comience con "a", "b" o "c" seguido de cualquier otro carácter.

```
alumno@administrador-20VE:~$ ls [abc]*
cliente.py

app-web:
index.html   sample_app.py  static   templates
running.txt  sample-app.sh  tempdir  venv
```

CP: Este comando permite copiar archivos y directorios.

cp ejemplo.txt ejemplo2.txt: En este caso se está copiando el contenido del ejemplo.txt y se está pegando en el ejemplo2.txt.

```
alumno@administrador-20VE:~$ cp ejemplo.txt ejemplo2.txt
alumno@administrador-20VE:~$ cat ejemplo2.txt
Este es el contenido del archivo
La evolución de los procesadores ha sido impresionante.
 Desde los primeros microprocesadores de 4 bits que revolucionaron
 la informática personal en los años 70 hasta los
 procesadores actuales de 64 bits que alimentan
supercomputadoras y servidores de alto rendimiento,
 la capacidad de procesamiento ha aumentado exponencialmente.
```

cp ejemplo.txt Documentos: En este caso se está copiando el archivo ejemplo.txt y se está dejando una copia en el directorio Documentos.

```
alumno@administrador-20VE:~$ cp ejemplo.txt  Documentos
alumno@administrador-20VE:~$ cd Documentos
alumno@administrador-20VE:~/Documentos$ ls
Dockerfile.txt  ejemplo.txt
```

mv: Mueve o cambia el nombre de archivos y directorios según cómo se utilice.

mv ejemplo.txt ejemplo2.txt: En este caso se está cambiando el nombre del archivo ejemplo.txt con como ejemplo2.txt, lo cual, contendrá su contenido del archivo ejemplo.txt.

```
alumno@administrador-20VE:~$ ls
app-web         Documentos    Imágenes        Plantillas   user_space_report.txt
cliente.py      ejemplo2.txt  informarmacion  Público      Vídeos
Descargas       ejemplo.txt   kubectl         servidor.py
Dockerfile      Escritorio    Música          snap
docker_ubuntu   hola.py       nginx.yaml      start.sh
alumno@administrador-20VE:~$ mv ejemplo.txt ejemplo2.txt
alumno@administrador-20VE:~$ ls
app-web         Documentos    informarmacion  Público               Vídeos
cliente.py      ejemplo2.txt  kubectl         servidor.py
Descargas       Escritorio    Música          snap
Dockerfile      hola.py       nginx.yaml      start.sh
docker_ubuntu   Imágenes      Plantillas      user_space_report.txt
```

rm: Elimina archivos y directorios.
rm ejemplo.txt: En este caso se elimina el archivo ejemplo.txt.

```
alumno@administrador-20VE:~/Documentos$ rm ejemplo.txt
alumno@administrador-20VE:~/Documentos$ ls
Dockerfile.txt
```

rm -r hola: En este caso se elimina el directorio hola.

```
alumno@administrador-20VE:~$ rm -r hola
alumno@administrador-20VE:~$ ls
app-web         Documentos    informarmacion  Público              Vídeos
cliente.py      ejemplo2.txt  kubectl         servidor.py
Descargas       Escritorio    Música          snap
Dockerfile      hola.py       nginx.yaml      start.sh
docker_ubuntu   Imágenes      Plantillas      user_space_report.txt
alumno@administrador-20VE:~$
```

mkdir: Se utiliza para crear directorios.
mkdir hola: En este caso se crea el directorio hola.

```
alumno@administrador-20VE:~$ mkdir hola
alumno@administrador-20VE:~$ ls
app-web         Documentos    Imágenes        Plantillas   user_space_report.txt
cliente.py      ejemplo2.txt  informarmacion   Público      Vídeos
Descargas       Escritorio    kubectl         servidor.py
Dockerfile      hola          Música          snap
docker_ubuntu   hola.py       nginx.yaml      start.sh
alumno@administrador-20VE:~$
```

type: Mostrar información sobre el tipo de comando.
type type, type ls, type cp: Muestra el tipo de comando que ejecutará el shell. En este caso de type, ls y cp.

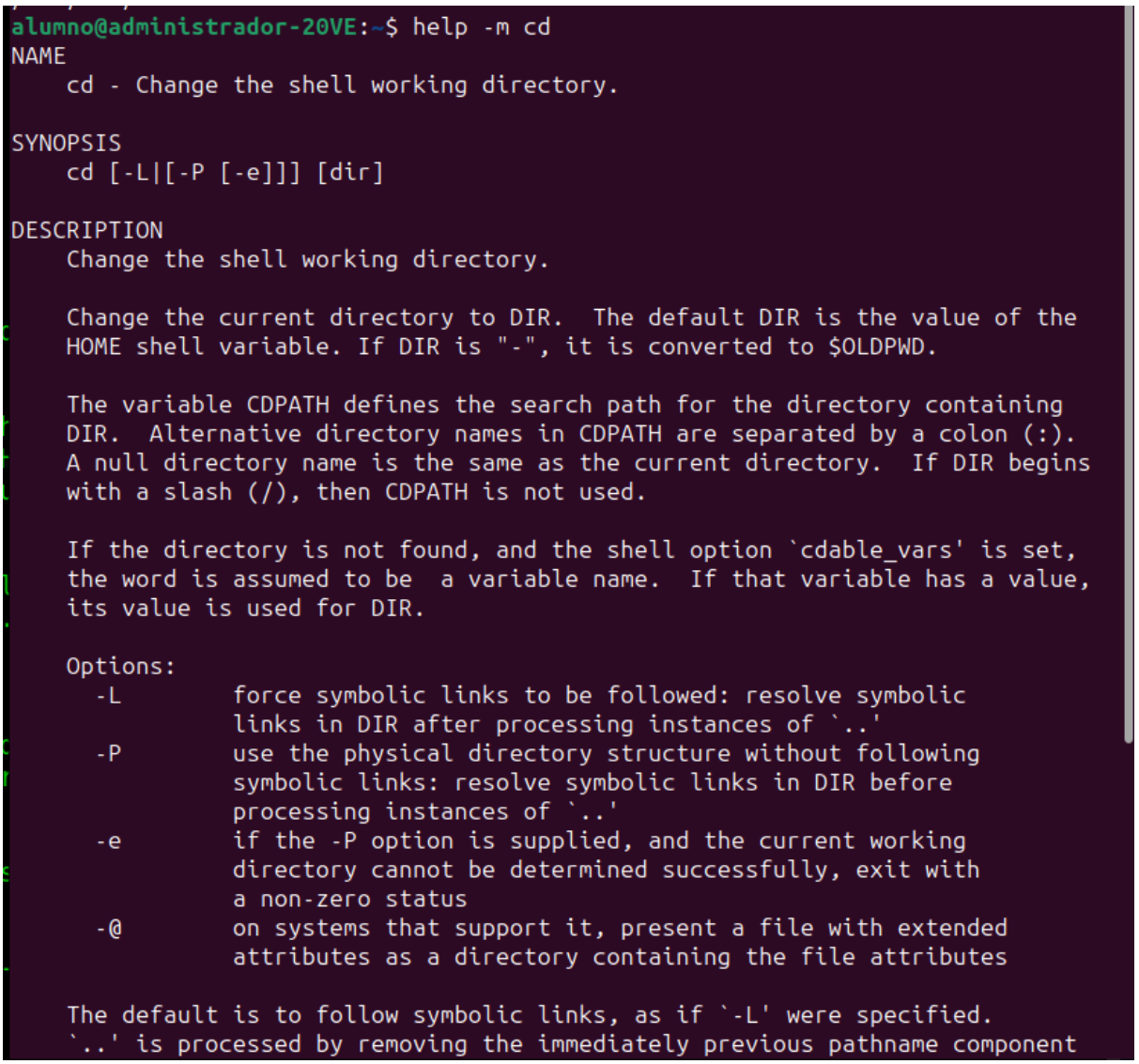
```
alumno@administrador-20VE:~$ type type
type es una orden interna del intérprete de ordenes
alumno@administrador-20VE:~$ type ls
ls es un alias de `ls --color=auto'
alumno@administrador-20VE:~$ type cp
cp está asociado (/usr/bin/cp)
```

which ls: Determinar la ubicación exacta de un ejecutable **ls .**

```
alumno@administrador-20VE:~$ which ls
/usr/bin/ls
alumno@administrador-20VE:~$
```

help: Es una función de ayuda incorporada disponible para cada una de las funciones integradas del shell.

help -m cd: Muestra información detallada sobre cómo usar el comando **cd.**



mkdir –help: Muestra información de ayuda sobre el comando **mkdir.**

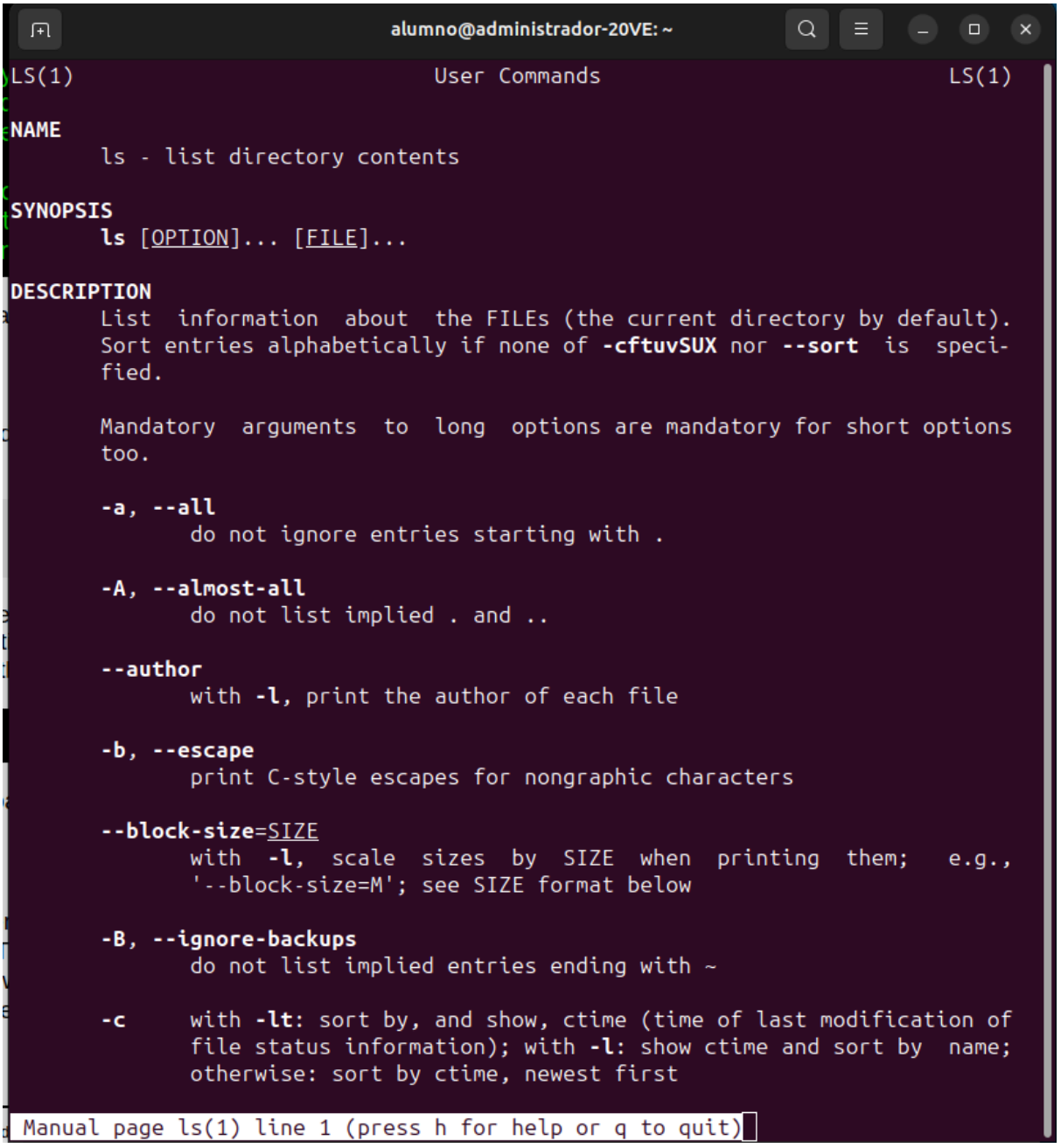
```
alumno@administrador-20VE:~$ mkdir --help
Modo de empleo: mkdir [OPCIÓN]... DIRECTORIO...
Crea los DIRECTORIO(s), si no existen ya.

Los argumentos obligatorios para las opciones largas son también obligatorios
para las opciones cortas.
  -m, --mode=MODE   set file mode (as in chmod), not a=rwx - umask
  -p, --parents     no error if existing, make parent directories as needed,
                    with their file modes unaffected by any -m option.
  -v, --verbose     print a message for each created directory
  -Z                   establece el contexto de seguridad SELinux de cada
                         directorio creado al tipo predeterminado
      --context[=CTX]  como -Z, o si se especifica CTX entonces establece el
                         contexto de seguridad SELinux o SMACK a CTX
      --help        display this help and exit
      --version     output version information and exit

ayuda en línea sobre GNU coreutils: <https://www.gnu.org/software/coreutils/>
Report any translation bugs to <https://translationproject.org/team/>
Full documentation <https://www.gnu.org/software/coreutils/mkdir>
or available locally via: info '(coreutils) mkdir invocation'
alumno@administrador-20VE:~$
```

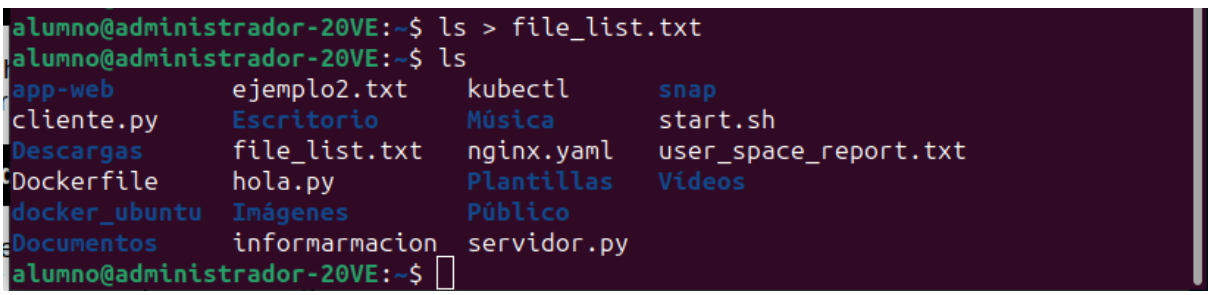
man ls: Mostrar el manual de usuario del comando **ls.**

```
alumno@administrador-20VE:~$ man ls
alumno@administrador-20VE:~$
```



Redirección de E/S:

ls > file_list.txt: El comando ls se ejecuta y los resultados se escriben en un archivo llamado **file_list.txt.**



ls >> file_list.txt: En este caso se utiliza **">>"** para agregar los nuevos resultados al archivo **file_list.txt.**

```
alumno@administrador-20VE:~$ ls >> file_list.txt
```

sort < file_list.txt: En este caso se procesa el contenido de file_list.txt.

```
alumno@administrador-20VE:~$ sort < file_list.txt
app-web
app-web
cliente.py
cliente.py
Descargas
Descargas
Dockerfile
Dockerfile
docker_ubuntu
docker_ubuntu
Documentos
Documentos
ejemplo2.txt
ejemplo2.txt
Escritorio
Escritorio
file_list.txt
file_list.txt
hola.py
hola.py
Imágenes
Imágenes
informarmacion
informarmacion
kubectl
kubectl
Música
Música
nginx.yaml
```

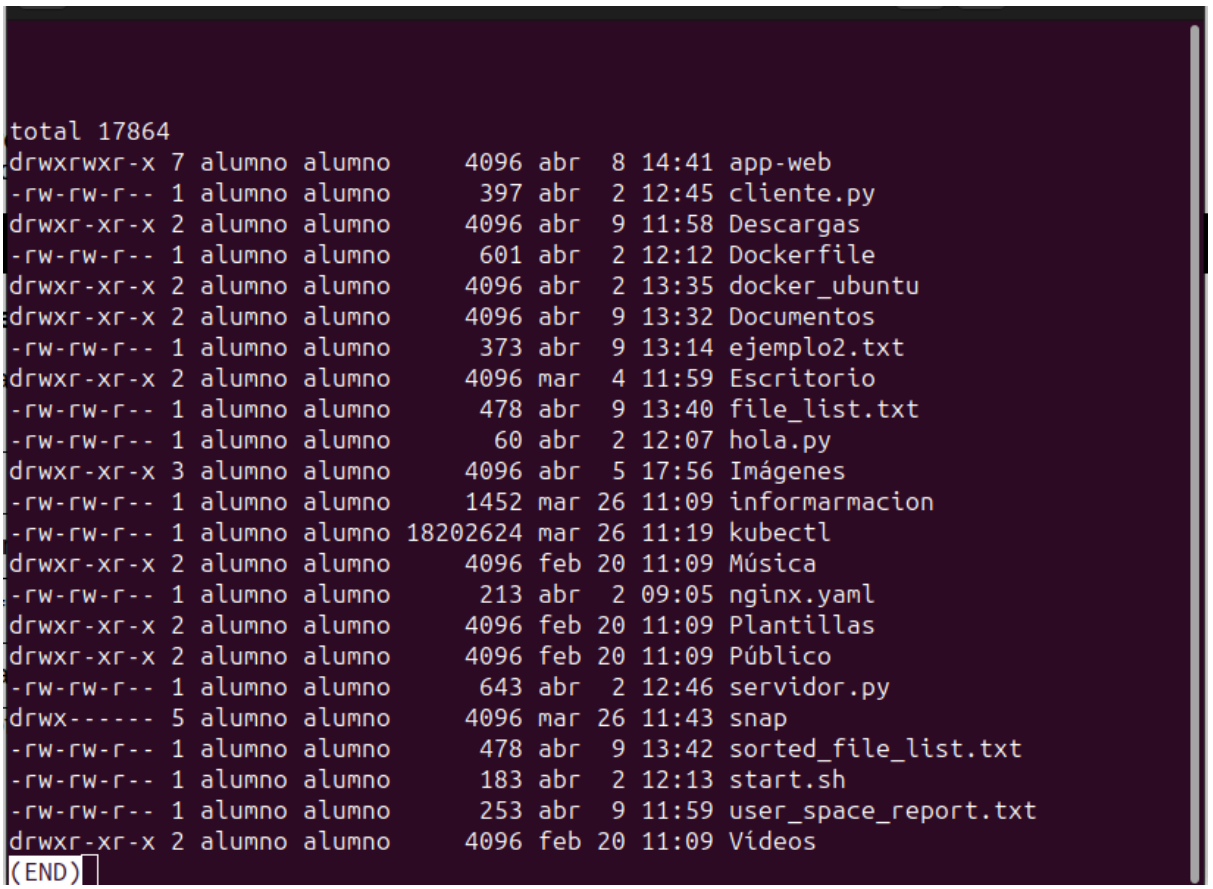
sort < file_list.txt > sorted_file_list.txt : En este caso se está redirigiendo la salida estándar a otro archivo llamado **sorted_file_list.txt.**

```
alumno@administrador-20VE:~$ sort < file_list.txt  > sorted_file_list.txt
alumno@administrador-20VE:~$ ls
app-web         ejemplo2.txt    kubectl      snap
cliente.py      Escritorio      Música       sorted_file_list.txt
Descargas       file_list.txt   nginx.yaml   start.sh
Dockerfile      hola.py         Plantillas   user_space_report.txt
docker_ubuntu   Imágenes        Público      Videos
Documentos      informarmacion  servidor.py
```

Pipelines: Permite conectar múltiples comandos.

ls -l | less: En este caso la salida del comando **ls** se introduce en **less.**

```
alumno@administrador-20VE:~$ ls -l | less
```



echo: Permite imprimir texto en la salida estándar, o redirigir el texto a un archivo.
echo this is a test: Este comando imprime el texto **this is a test.**
echo*: Cualquier argumento pasado con **echo** y se muestra.
echo D*: Muestra nombres de archivos y directorios en el directorio actual que comiencen con la letra "D".
echo *s:Muestra nombres de archivos y directorios en el directorio actual que termine con la letra "s".
echo [[:upper:]]*: Muestra los nombres de archivos y directorios en el directorio actual que comiencen con una letra mayúscula.
echo /usr/*/share: Muestra una lista de directorios dentro del directorio /usr/ que tienen un subdirectorio llamado share.
echo ~: Imprime la ruta al directorio de inicio del usuario actual.

```
alumno@administrador-20VE:~$ echo this is a test
this is a test
alumno@administrador-20VE:~$ echo *
app-web cliente.py Descargas Dockerfile docker_ubuntu Documentos ejemplo2.txt Es
critorio file_list.txt hola.py Imágenes informarmacion kubectl Música nginx.yaml
 Plantillas Público servidor.py snap sorted_file_list.txt start.sh user_space_re
port.txt Vídeos
alumno@administrador-20VE:~$ ls
app-web         ejemplo2.txt    kubectl      snap
cliente.py      Escritorio      Música       sorted_file_list.txt
Descargas       file_list.txt   nginx.yaml   start.sh
Dockerfile      hola.py         Plantillas   user_space_report.txt
docker_ubuntu   Imágenes        Público      Vídeos
Documentos      informarmacion  servidor.py
alumno@administrador-20VE:~$ echo D*
Descargas Dockerfile Documentos
alumno@administrador-20VE:~$ echo *s
Descargas Documentos Imágenes Plantillas Vídeos
alumno@administrador-20VE:~$ echo [[:upper:]]*
Descargas Dockerfile Documentos Escritorio Imágenes Música Plantillas Público Ví
deos
alumno@administrador-20VE:~$ echo /usr/*/share
/usr/local/share
alumno@administrador-20VE:~$ echo ~
/home/alumno
alumno@administrador-20VE:~$
```

echo ~foo: Imprime la ruta al directorio del usuario **~foo.**
echo $((2+2)): Imprime el resultado de la suma.
echo $(($((52)) *3)):** Imprime el resultado de la operación.

```
alumno@administrador-20VE:~$ echo ~foo
~foo
alumno@administrador-20VE:~$ echo $((2+2))
4
alumno@administrador-20VE:~$ echo $(($((5**2)) * 3))
75
alumno@administrador-20VE:~$
```

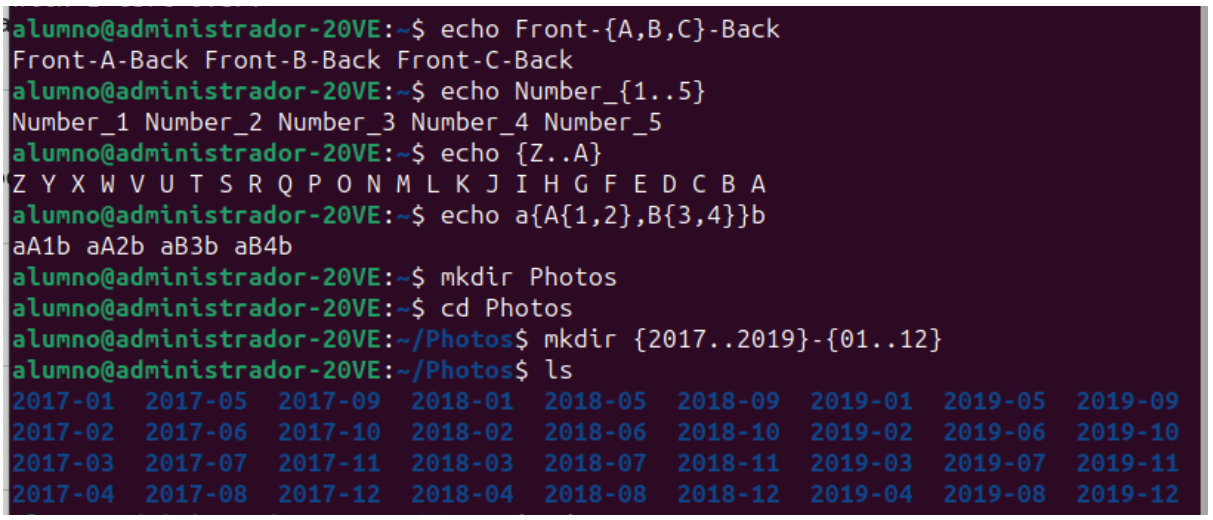
echo Five divided by two equals $((5/2)): Devuelve el resultado de 5 entre 2.
echo with $((5%2)) left over.: Devuelve el resultado del resto de 5 entre 2.

```
alumno@administrador-20VE:~$ echo Five divided by two equals $((5/2))
Five divided by two equals 2
alumno@administrador-20VE:~$ echo with $((5%2)) left over.
with 1 left over.
```

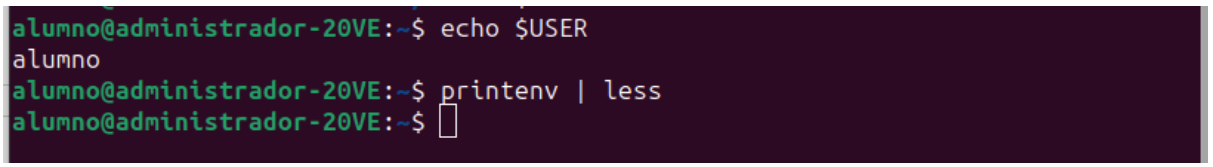
Expansión de llaves

echo Front-{A,B,C}-Back: Muestra cadenas de texto con Front-{A,B,C}-Back.
echo Number_{1..5}: Muestra cadena de texto con la palabra Number_ seguido con los números del 1 al 5.
echo {Z..A}: Muestra el abecedario en orden inverso.
echo a{A{1,2},B{3,4}}b: Muestra cadenas de texto anidadas como aA1b.

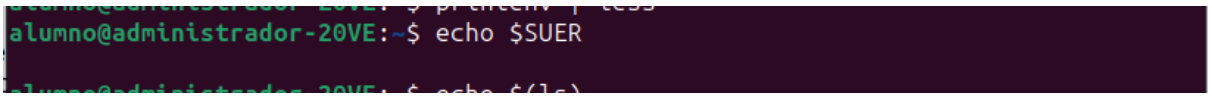
mkdir Photos: Se crea un archivo llamado Photos.
mkdir {2017..2019}-{01..12}: Se crear una serie de directorios nombrados en formato numérico “Año - Mes”.



Expansión de parámetros:
echo $USER: Muestra el contenido de USER, lo cual contiene el nombre de usuario.
printenv | less: Muestra una lista de variables disponibles.



echo $SUER: Es esta situación se escribió mal el patrón por el cual sale como resultado una cadena vacía.



echo $(ls): Muestra todos los archivos y directorios.
ls -l $(which cp): Mostrará información detallada sobre el archivo ejecutable del comando cp en tu sistema.
file $(ls /usr/bin/* | grep bin/zip): Listará todos los archivos en el directorio /usr/bin/ y luego va a filtrar solo aquellos que contengan la cadena "bin/zip". Luego, el resultado se pasará como argumento al comando file, que se utiliza para determinar el tipo de archivo.
ls -l `which cp`: Devuelve la ubicación del comando cp.

```
alumno@administrador-20VE:~$ echo $(ls)
1jnb.cif 3pl1.cif 3ugl.cif 4y16.cif 6i86.cif 6i96.cif 7laa.cif app-web cliente.p
y Descargas Dockerfile docker_ubuntu Documentos ejemplo2.txt Escritorio file_lis
t.txt hola.py Imágenes informarmacion kubectl Música nginx.yaml Photos Plantilla
s Público servidor.py snap sorted_file_list.txt start.sh user_space_report.txt V
ideos
alumno@administrador-20VE:~$ ls -l $(which cp)
-rwxr-xr-x 1 root root 141832 ene 10  2023 /usr/bin/cp
alumno@administrador-20VE:~$ file $(ls /usr/bin/* | grep bin/zip)
/usr/bin/zip:        ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dy
namically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=b23ff3e
6fd430570d529f6d5ede75cc0c71fbfdc, for GNU/Linux 3.2.0, stripped
/usr/bin/zipcloak:   ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dy
namically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=a715c8c
07d0ad4f96a8ffac467caee47e7c2447d, for GNU/Linux 3.2.0, stripped
/usr/bin/zipdetails: Perl script text executable
/usr/bin/zipgrep:    POSIX shell script, ASCII text executable
/usr/bin/zipinfo:    ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dy
namically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=8d14fc8
be748831194e4e07ad45feb0c46e80c5e, for GNU/Linux 3.2.0, stripped
/usr/bin/zipnote:    ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dy
namically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=1a146bc
153f9a4ca45bfa55684d2f7ecd4f48c8f, for GNU/Linux 3.2.0, stripped
/usr/bin/zipsplit:   ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dy
namically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=5b75c26
b725add8c3e0b9ab8b2a86fcc5d7352b2, for GNU/Linux 3.2.0, stripped
alumno@administrador-20VE:~$ ls -l `which cp`
-rwxr-xr-x 1 root root 141832 ene 10  2023 /usr/bin/cp
```

Quoting:

echo this is a test: Muestra el texto pero sin espacios en blanco.
echo The total is $100.00: Muestra el texto pero sustituye $1 como una cadena vacía.

```
alumno@administrador-20VE:~$ echo this is a     test
this is a test
alumno@administrador-20VE:~$ echo The total is $100.00
The total is 00.00
alumno@administrador-20VE:~$
```

Double Quotes:
Si tuviéramos un archivo llamado “hola mundo.txt” cuando intentamos acceder a este archivo con el comando **ls -l two words.txt,** saldra error de acceso ya que lo entiendo como si fuera dos archivos por separado.
Mientras, si el comando lleva doble comilla en el nombre del archivo: **ls -l “two words.txt”,** este mostrará el resultado del comando.

```
onworks@onworks:~$ ls
 Desktop     Downloads          Music      Public   Templates
 Documents  'hola mundo.txt'    Pictures   snap     Videos
onworks@onworks:~$ ls -l hola mundo.txt
ls: cannot access 'hola': No such file or directory
ls: cannot access 'mundo.txt': No such file or directory
onworks@onworks:~$ ls -l "hola mundo.txt"
-rw-rw-r-- 1 onworks onworks 5 Apr 14 00:13 'hola mundo.txt'
onworks@onworks:~$
```

echo "$USER $((2+2)) $(cal)": Muestra el nombre de usuario, el resultado de la sumatoria de 2+2 y el calendario de este mes y año.

```
alumno@administrador-20VE:~$ echo "$USER $((2+2)) $(cal)"
alumno 4      Abril 2024
do lu ma mi ju vi sá
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30

alumno@administrador-20VE:~$
```

echo this is a test: Muestra el texto **this is a test** eliminado los espacios.
echo "this is a test": Muestra el texto **this is a test,** pero sin eliminar los espacios.
echo $(cal): Muestra el calendario pero suprimiendo la división de las palabras y los espacios.
echo "$(cal)": Muestra el calendario sin suprimir.

```
alumno@administrador-20VE:~$ echo this is a   test
this is a test
alumno@administrador-20VE:~$ echo "this is a   test"
this is a   test
alumno@administrador-20VE:~$ echo $(cal)
Abril 2024 do lu ma mi ju vi sá 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
20 21 22 23 24 25 26 27 28 29 30
alumno@administrador-20VE:~$ echo "$(cal)"
     Abril 2024
do lu ma mi ju vi sá
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30

alumno@administrador-20VE:~$
```

Single Quotes:
echo text ~/*.txt {a,b} $(echo foo) $((2+2)) $USER: Mostrará una concatenación de todas estas cadenas y valores. Donde mostrar el texto text, luego, todos los archivos con extensión ".txt", mostrará las cadenas las cadenas "a" y "b".Asimismo imprimirá el comando “echo foo” y por último, el resultado de la sumatoria y el nombre del usuario.
echo "text ~/*.txt {a,b} $(echo foo) $((2+2)) $USER": En este caso se mostrará el siguiente resultado ***text ~/*.txt {a,b} foo 4 me***.

echo 'text ~/*.txt {a,b} $(echo foo) $((2+2)) $USER': Mostrará el siguiente resultado, ***text ~/*.txt {a,b} $(echo foo) $((2+2)) $USER***

```
alumno@administrador-20VE:~$ echo text ~/*.txt {a,b} $(echo foo) $((2+2)) $USER
text /home/alumno/ejemplo2.txt /home/alumno/file_list.txt /home/alumno/sorted_file_list.txt /
home/alumno/user_space_report.txt a b foo 4 alumno
alumno@administrador-20VE:~$ echo "text ~/*.txt {a,b} $(echo foo) $((2+2)) $USER"
text ~/*.txt {a,b} foo 4 alumno
alumno@administrador-20VE:~$
```

```
alumno@administrador-20VE:~$ echo 'text ~/*.txt {a,b} $(echo foo) $((2+2)) $USER'
text ~/*.txt {a,b} $(echo foo) $((2+2)) $USER
alumno@administrador-20VE:~$
```

Escaping Characters:

echo "The balance for user $USER is: \$5.00": Mostrará el texto ***The balance for user alumno is: $5.00.***

```
alumno@administrador-20VE:~$ echo "The balance for user $USER is : \$5.00"
The balance for user alumno is : $5.00
alumno@administrador-20VE:~$
```

ls -r : Muestra todos los archivos en orden inverso.
ls - - reverse: Muestra todos los archivos en orden inverso.
ls -l : Muestra muestre los archivos mostrando información detallada de cada uno.

```
alumno@administrador-20VE:~$ ls -r
Vídeos                  Plantillas      hola.py        Descargas    3ugl.cif
user_space_report.txt   Photos          file_list.txt  cliente.py   3pl1.cif
start.sh                nginx.yaml      Escritorio     app-web      1jnb.cif
sorted_file_list.txt    Música          ejemplo2.txt   7laa.cif
snap                    kubectl         Documentos     6i96.cif
servidor.py             informarmacion  docker_ubuntu  6i86.cif
Público                 Imágenes        Dockerfile     4y16.cif
alumno@administrador-20VE:~$ ls --reverse
Vídeos                  Plantillas      hola.py        Descargas    3ugl.cif
user_space_report.txt   Photos          file_list.txt  cliente.py   3pl1.cif
start.sh                nginx.yaml      Escritorio     app-web      1jnb.cif
sorted_file_list.txt    Música          ejemplo2.txt   7laa.cif
snap                    kubectl         Documentos     6i96.cif
servidor.py             informarmacion  docker_ubuntu  6i86.cif
Público                 Imágenes        Dockerfile     4y16.cif
alumno@administrador-20VE:~$ ls -l
total 28052
-rw-rw-r--  1 alumno alumno  2508288 abr 12 15:02 1jnb.cif
-rw-rw-r--  1 alumno alumno   205572 abr 12 14:30 3pl1.cif
-rw-rw-r--  1 alumno alumno   300722 abr 12 15:08 3ugl.cif
-rw-rw-r--  1 alumno alumno   731559 abr 12 15:00 4y16.cif
-rw-rw-r--  1 alumno alumno   769718 abr 12 14:29 6i86.cif
-rw-rw-r--  1 alumno alumno  1129704 abr 12 14:12 6i96.cif
-rw-rw-r--  1 alumno alumno  4770925 abr 12 14:56 7laa.cif
drwxrwxr-x  7 alumno alumno     4096 abr  8 14:41 app-web
-rw-rw-r--  1 alumno alumno      397 abr  2 12:45 cliente.py
drwxr-xr-x  2 alumno alumno     4096 abr 15 09:06 Descargas
-rw-rw-r--  1 alumno alumno      601 abr  2 12:12 Dockerfile
drwxr-xr-x  2 alumno alumno     4096 abr  2 13:35 docker_ubuntu
drwxr-xr-x  2 alumno alumno     4096 abr  9 13:32 Documentos
-rw-rw-r--  1 alumno alumno      373 abr  9 13:14 ejemplo2.txt
drwxr-xr-x  2 alumno alumno     4096 abr 12 15:53 Escritorio
```

ls –human-readable: Muestra los archivos pero los tamaños de los archivos serán en un formato como Kilobytes, megabytes, etc.

```
alumno@administrador-20VE:~$ ls --reverse
Vídeos                  Plantillas      hola.py         Descargas   3ugl.cif
user_space_report.txt   Photos          file_list.txt   cliente.py  3pl1.cif
start.sh                nginx.yaml      Escritorio      app-web     1jnb.cif
sorted_file_list.txt    Música          ejemplo2.txt    7laa.cif
snap                    kubectl         Documentos      6i96.cif
servidor.py             informarmacion  docker_ubuntu   6i86.cif
Público                 Imágenes        Dockerfile      4y16.cif
alumno@administrador-20VE:~$ ls --human-readable
1jnb.cif  cliente.py     hola.py         Público
3pl1.cif  Descargas      Imágenes        servidor.py
3ugl.cif  Dockerfile     informarmacion  snap
4y16.cif  docker_ubuntu  kubectl         sorted_file_list.txt
6i86.cif  Documentos     Música          start.sh
6i96.cif  ejemplo2.txt   nginx.yaml      user_space_report.txt
7laa.cif  Escritorio     Photos          Vídeos
app-web   file_list.txt  Plantillas
```

ls –full-time: Muestre la fecha y hora de modificación de cada archivo.

```
alumno@administrador-20VE:~$ ls --full-time
total 28052
-rw-rw-r--  1 alumno alumno  2508288 2024-04-12 15:02:06.459082975 -0500 1jnb.cif
-rw-rw-r--  1 alumno alumno   205572 2024-04-12 14:30:53.233849243 -0500 3pl1.cif
-rw-rw-r--  1 alumno alumno   300722 2024-04-12 15:08:06.446223873 -0500 3ugl.cif
-rw-rw-r--  1 alumno alumno   731559 2024-04-12 15:00:23.416502277 -0500 4y16.cif
-rw-rw-r--  1 alumno alumno   769718 2024-04-12 14:29:38.979854622 -0500 6i86.cif
-rw-rw-r--  1 alumno alumno  1129704 2024-04-12 14:12:28.112687071 -0500 6i96.cif
-rw-rw-r--  1 alumno alumno  4770925 2024-04-12 14:56:33.875652238 -0500 7laa.cif
drwxrwxr-x  7 alumno alumno     4096 2024-04-08 14:41:24.301299429 -0500 app-web
-rw-rw-r--  1 alumno alumno      397 2024-04-02 12:45:52.602312699 -0500 cliente.py
drwxr-xr-x  2 alumno alumno     4096 2024-04-15 09:06:35.906879719 -0500 Descargas
-rw-rw-r--  1 alumno alumno      601 2024-04-02 12:12:57.874692716 -0500 Dockerfile
drwxr-xr-x  2 alumno alumno     4096 2024-04-02 13:35:48.821182470 -0500 docker_ubuntu
drwxr-xr-x  2 alumno alumno     4096 2024-04-09 13:32:45.848245713 -0500 Documentos
-rw-rw-r--  1 alumno alumno      373 2024-04-09 13:14:47.546752039 -0500 ejemplo2.txt
drwxr-xr-x  2 alumno alumno     4096 2024-04-12 15:53:23.306868836 -0500 Escritorio
-rw-rw-r--  1 alumno alumno      478 2024-04-09 13:40:15.138125863 -0500 file_list.txt
-rw-rw-r--  1 alumno alumno       60 2024-04-02 12:07:14.615055967 -0500 hola.py
drwxr-xr-x  3 alumno alumno     4096 2024-04-05 17:56:14.786562332 -0500 Imágenes
-rw-rw-r--  1 alumno alumno     1452 2024-03-26 11:09:04.653692395 -0500 informarmacion
-rw-rw-r--  1 alumno alumno 18202624 2024-03-26 11:19:30.413797032 -0500 kubectl
drwxr-xr-x  2 alumno alumno     4096 2024-02-20 11:09:24.912157515 -0500 Música
-rw-rw-r--  1 alumno alumno      213 2024-04-02 09:05:40.962021254 -0500 nginx.yaml
drwxrwxr-x 38 alumno alumno     4096 2024-04-12 17:12:03.565286114 -0500 Photos
drwxr-xr-x  2 alumno alumno     4096 2024-02-20 11:09:24.912157515 -0500 Plantillas
drwxr-xr-x  2 alumno alumno     4096 2024-02-20 11:09:24.912157515 -0500 Público
-rw-rw-r--  1 alumno alumno      643 2024-04-02 12:46:18.665988242 -0500 servidor.py
drwx------  6 alumno alumno     4096 2024-04-12 14:10:41.669609524 -0500 snap
-rw-rw-r--  1 alumno alumno      478 2024-04-09 13:42:10.868417174 -0500 sorted_file_list.txt
-rw-rw-r--  1 alumno alumno      183 2024-04-02 12:13:14.410482530 -0500 start.sh
-rw-rw-r--  1 alumno alumno      253 2024-04-09 11:59:34.196281374 -0500 user_space_report.txt
drwxr-xr-x  2 alumno alumno     4096 2024-02-20 11:09:24.912157515 -0500 Vídeos
alumno@administrador-20VE:~$
```

echo -e "Inserting several blank lines\n\n\n: Muestra la frase "Inserting several blank lines" seguida de tres líneas en blanco.

echo -e "Words\tseparated\tby\thorizontal\ttabs.”: Muestra la frase "Words separated by horizontal tabs.", donde las palabras están separadas

echo -e "\aMy computer went \"beep\".": Muestra la frase "My computer went "beep"." y hará sonar un tono de alerta antes de que se imprima la frase.

echo -e "DEL C:\\WIN2K\\LEGACY_OS.EXE": Muestra la ruta de un archivo en el sistema Windows sin modificarla, pero mostrando las barras invertidas como parte del texto.

```
alumno@administrador-20VE:~$ echo -e "Inserting several blank lines\n\n\n"
Inserting several blank lines



alumno@administrador-20VE:~$ echo -e "Words\tseparated\tby\thorizontal\ttabs."
Words   separated       by      horizontal      tabs.
alumno@administrador-20VE:~$ echo -e "\aMy computer went \"beep\"."
My computer went "beep".
alumno@administrador-20VE:~$ echo -e "DEL C:\\WIN2K\\LEGACY_OS.EXE"
DEL C:\WIN2K\LEGACY_OS.EXE
alumno@administrador-20VE:~$
```

ls -l /bin/bash: Muestra información detallada sobre el archivo /bin/bash.

```
alumno@administrador-20VE:~$ ls -l /bin/bash
-rwxr-xr-x 1 root root 1437832 ene  7  2023 /bin/bash
```

chmod 664 ejemplo2.txt: Se establece los permisos del archivo ejemplo2.txt de la siguiente manera:
6: Esto significa que el propietario del archivo tiene permisos de lectura y escritura.
6: Los miembros del grupo pueden leer y escribir el archivo.
4: Otros usuarios solo pueden leer el archivo.

```
alumno@administrador-20VE:~$ chmod 664 ejemplo2.txt
alumno@administrador-20VE:~$
```

su: Para convertirse en superusuario.

```
alumno@administrador-20VE:~$ su
Contraseña:
```

sudo some_comand: Para ejecutar un comando como superusuario.

```
alumno@administrador-20VE:~$ sudo some_comand
[sudo] contraseña para alumno:
```

sudo -i: Permite abrir una nueva sesión de shell como el usuario root o superusuario.

```
alumno@administrador-20VE:~$ sudo -i
[sudo] contraseña para alumno:
root@administrador-20VE:~#
```

sudo chown alumno ejemplo2.txt: Se cambia el propietario del archivo ejemplo2.txt.

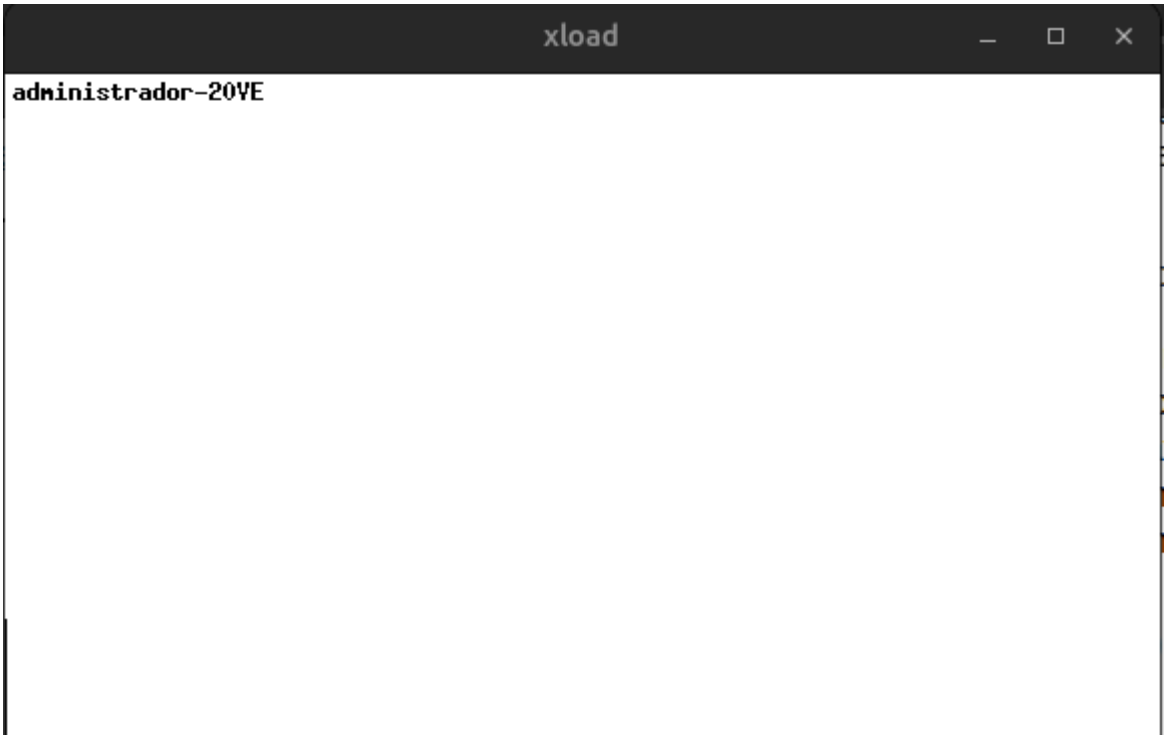
```
alumno@administrador-20VE:~$ sudo chown alumno ejemplo2.txt
alumno@administrador-20VE:~$
```

chgrp alumno ejemplo.txt: Se cambia la propiedad del grupo de un archivo o directorio.

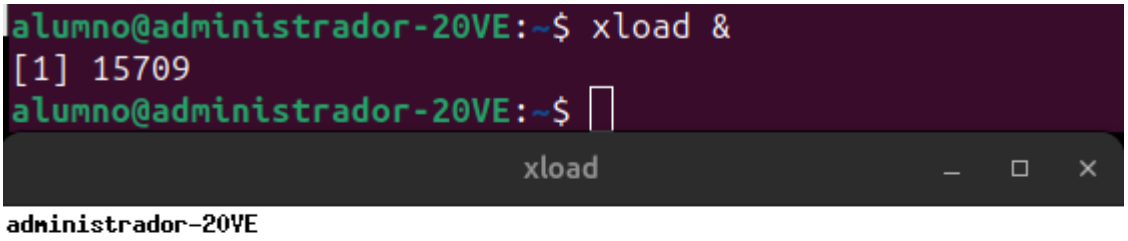
```
alumno@administrador-20VE:~$ chgrp alumno ejemplo2.txt
alumno@administrador-20VE:~$
```

xload: Muestra gráficamente la carga promedio del sistema.

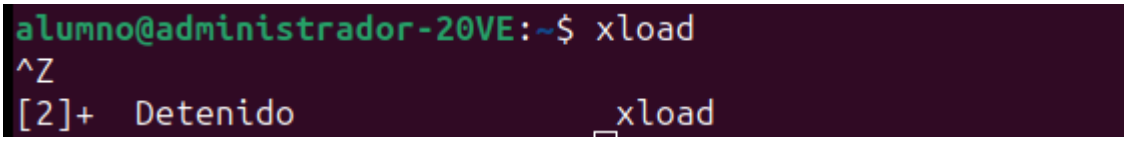
```
alumno@administrador-20VE:~$ xload
```



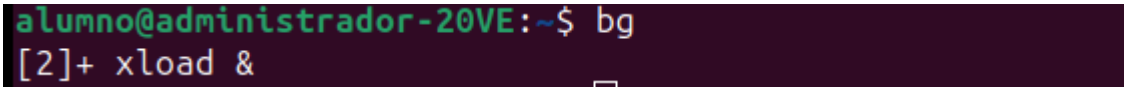
xload &: Muestra gráficamente la carga promedio del sistema pero lo pondremos en segundo plano.



Al escribir Ctrl-zy el proceso quedará suspendido.



bg: Sirve para reanudar el proceso en segundo plano.



jobs: Muestra una lista de los procesos que hemos iniciado.
ps: Muestra una lista de los procesos que hemos iniciado.

```
alumno@administrador-20VE:~$ jobs
[1]-  Ejecutando                 xload &
[2]+  Ejecutando                 xload &
alumno@administrador-20VE:~$ ps
    PID TTY          TIME CMD
   8132 pts/0    00:00:00 bash
  15709 pts/0    00:00:00 xload
  15866 pts/0    00:00:00 xload
  15984 pts/0    00:00:00 ps
```

xload & : Muestra gráficamente la carga promedio del sistema pero lo pondremos en segundo plano.

```
alumno@administrador-20VE:~$ xload &
[4] 16085
```

kill %4: Sirve para matar un proceso en este caso se termina el proceso 4

```
alumno@administrador-20VE:~$ kill %4
alumno@administrador-20VE:~$
```

```
alumno@administrador-20VE:~$ xload &
[2] 16435
[1]   Terminado                  xload
```

ps: Muestra una lista de los procesos que hemos iniciado,

```
alumno@administrador-20VE:~$ ps
    PID TTY          TIME CMD
   8132 pts/0    00:00:00 bash
  16085 pts/0    00:00:00 xload
  16225 pts/0    00:00:00 ps
```

kill 16435: Termina el proceso con el PID: 16435.

```
alumno@administrador-20VE:~$ kill 16435
alumno@administrador-20VE:~$
```

ps x | grep bad_program: Permite encontrar y listar procesos que contienen el texto "bad_program".

```
alumno@administrador-20VE:~$ ps x | grep bad_program
  18183 pts/0    S+     0:00 grep --color=auto bad_program
```

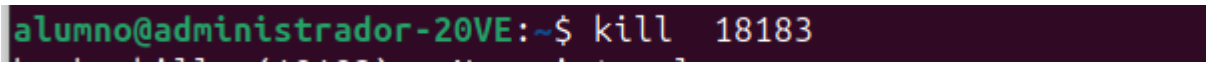
kill -SIGTERM 18183: Está solicitando al sistema operativo que envíe la señal SIGTERM al proceso con el PID 18183.

```
alumno@administrador-20VE:~$ kill -SIGTERM 18183
```

kill -SIGKILL 18183: Envía la señal SIGKILL al proceso con el ID de proceso (PID) 18183.

```
alumno@administrador-20VE:~$ kill -SIGKILL 18183
```

kill 18183: Intentará terminar el proceso con el PID 18183.



kill -9 18183: Termina el proceso con el PID 18183 de manera inmediata e irrecuperable