

## Introducción a GNU/Linux

**GNU/Linux** es un sistema operativo tipo Unix compuesto por **software libre** y de **código abierto**.

### Software libre:

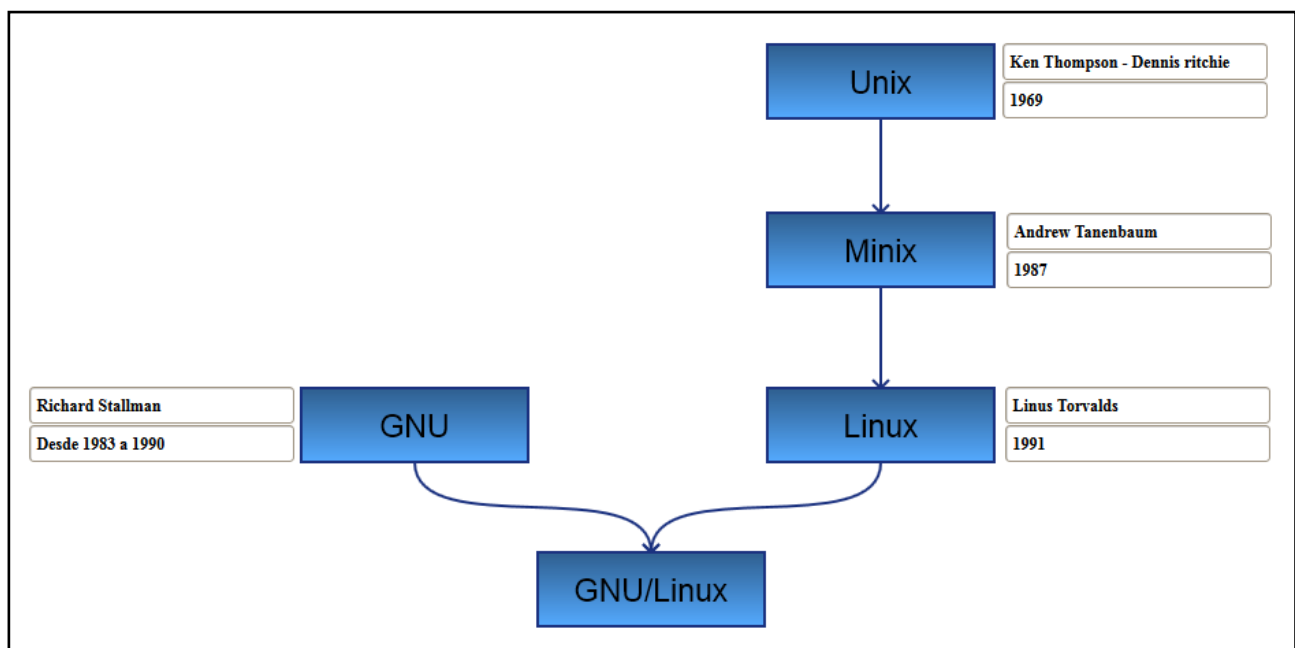
Su código fuente puede ser estudiado, modificado y utilizado libremente con cualquier finalidad y redistribuido con cambios o mejoras. No necesariamente implica que sea gratuito, libre hace referencia a las libertades que se tiene sobre el software; usar, estudiar, distribuir y mejorar.

### Código abierto:

Modelo de desarrollo de software basado en la colaboración abierta, apunta a los beneficios de compartir el código fuente en pos de ampliar la participación buscando la mejora continua.

Este sistema operativo surge de las contribuciones de varios proyectos de software, los mas destacados son; GNU iniciado por Richard Stallman en 1983 y el kernel, Linux, iniciado por Linus Torvalds en 1991.

Si bien comúnmente se lo llama Linux, ese en realidad es el nombre del kernel o núcleo, quién básicamente es el encargado de facilitar a los distintos programas acceso al hardware a través de llamadas al sistema.

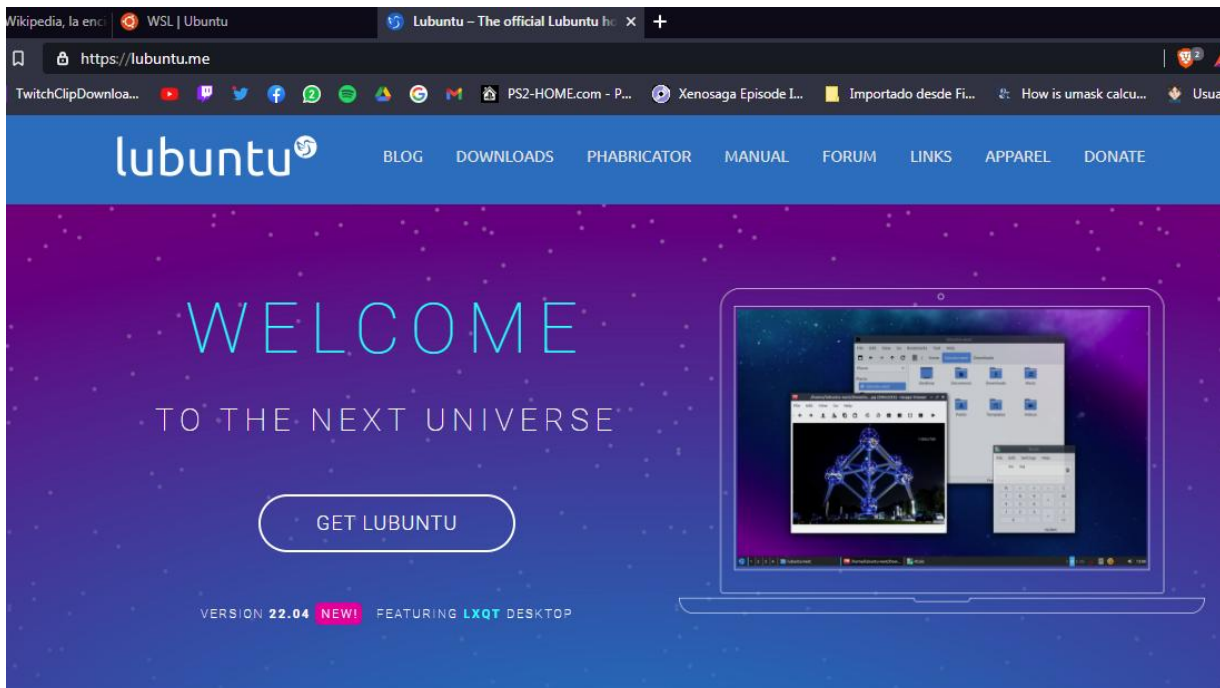


Los componentes básicos del proyecto GNU incluyeron:

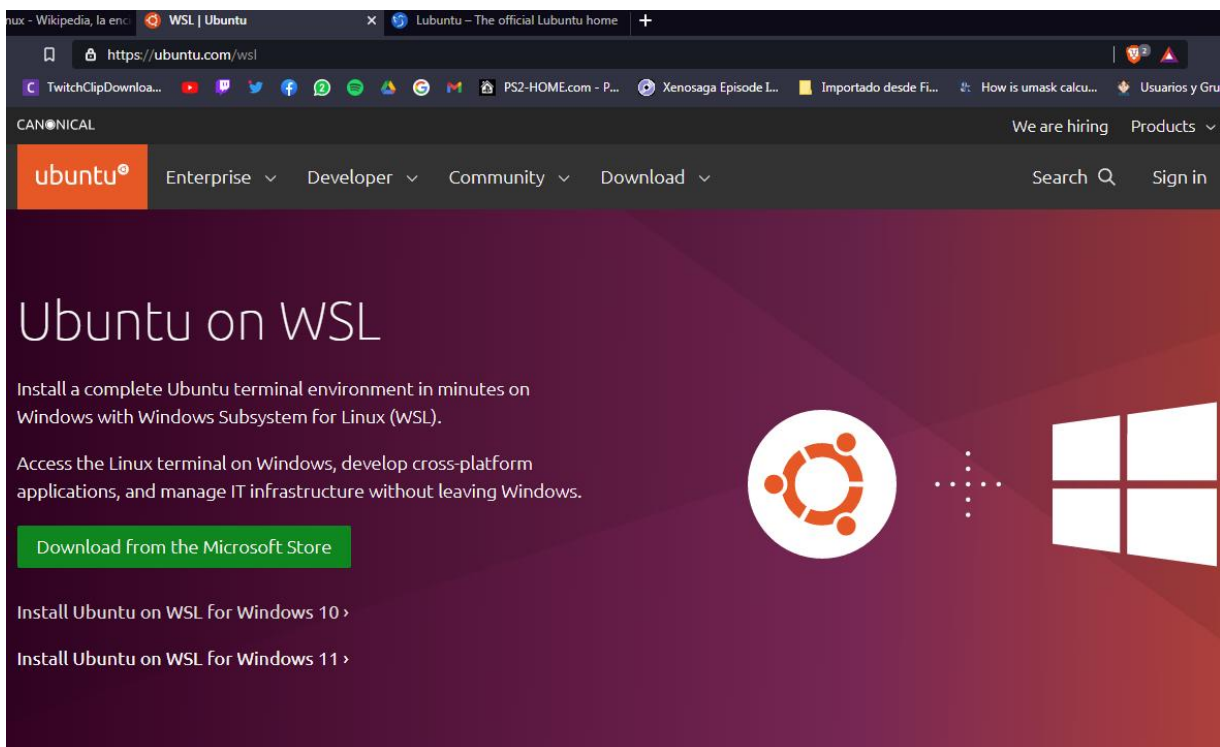
- El GCC (GNU Compiler Collection)
- La biblioteca glibc (aprox 500 llamadas al sistema)
- Las utilidades del núcleo coreutils (comandos básicos, cat, ls, rm, etc.)
- El debugger GDB
- Las utilidades binarias binutils (herramientas usadas por el GCC, make y GDB)
- El GNU Bash Shell

Hoy en día existen muchas distribuciones basadas en GNU/Linux, quizá una de las mas utilizadas UBUNTU, iniciado por Mark Shuttleworth y basada en el sistema operativo Debian GNU/Linux de Ian Murdock.

El contenido didáctico de éste apunte esta realizado sobre una versión del Lubuntu, la cual se pueden descargar desde <https://lubuntu.me>

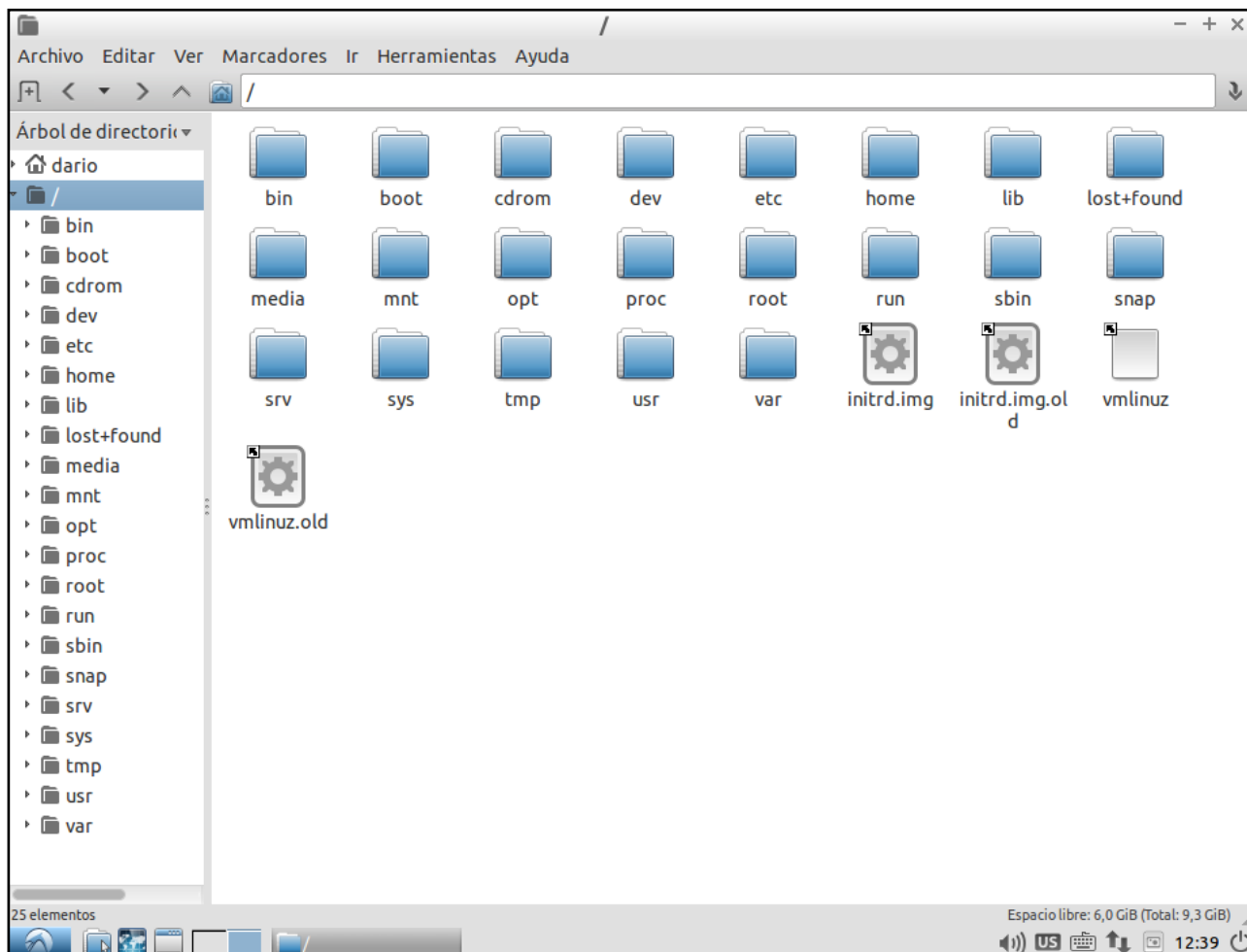


En el caso de tener Microsoft Windows 10/11 y no querer/poder trabajar con una maquina virtual, pueden instalar Ubuntu sobre Windows desde <https://ubuntu.com/wsl>















## Sistema de archivos

A continuación se presenta la estructura típica del sistema de archivos de Linux (Lubuntu)



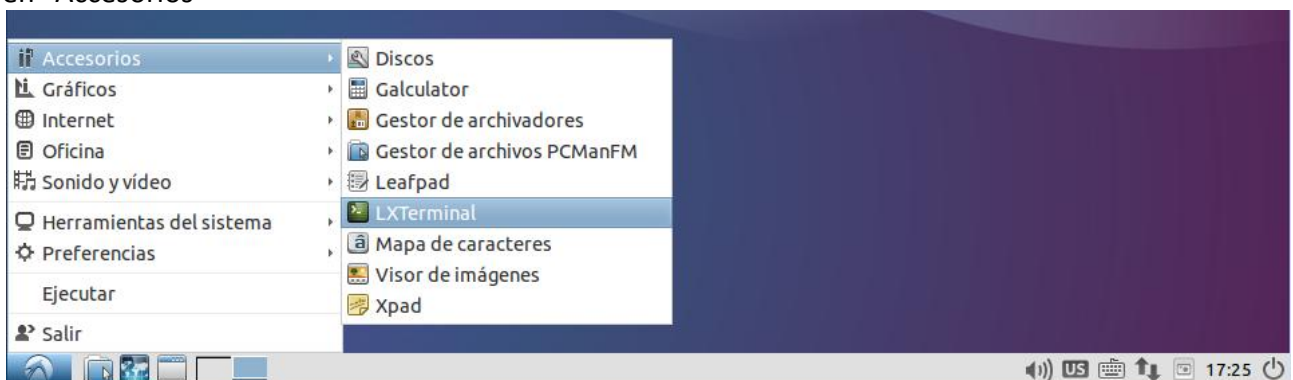
La raíz del árbol de directorios se denota con /, sería como por ejemplo C:\ en Microsoft Windows, de allí se desprende un conjunto de directorios, entre los cuales destacaremos los siguientes:

 bin	Almacena archivos binarios (Ejecutables) que pueden ser ejecutados por cualquier usuario del sistema. Por ejemplo los comandos que permiten la ejecución de utilidades básicas de la terminal de Linux, como <b>cp</b> , <b>cat</b> , <b>chmod</b> entre otros.
 boot	Contiene archivos necesarios para el arranque, antes de que el kernell empiece a ejecutar en modo usuario.
 dev	En Linux, los dispositivos son representados a través de archivos, cuando accedemos un dispositivo, lo hacemos leyendo o escribiendo sobre un archivo asociado con dicho dispositivo. Un archivo sda por ejemplo identifica a un disco duro.
 etc	Contiene archivos de configuración tanto a nivel sistema operativo como programas y aplicaciones. Por ejemplo los archivos <b>sudoers</b> , <b>passwd</b> , <b>shadow</b> y <b>bash.bashrc</b> entre otros.

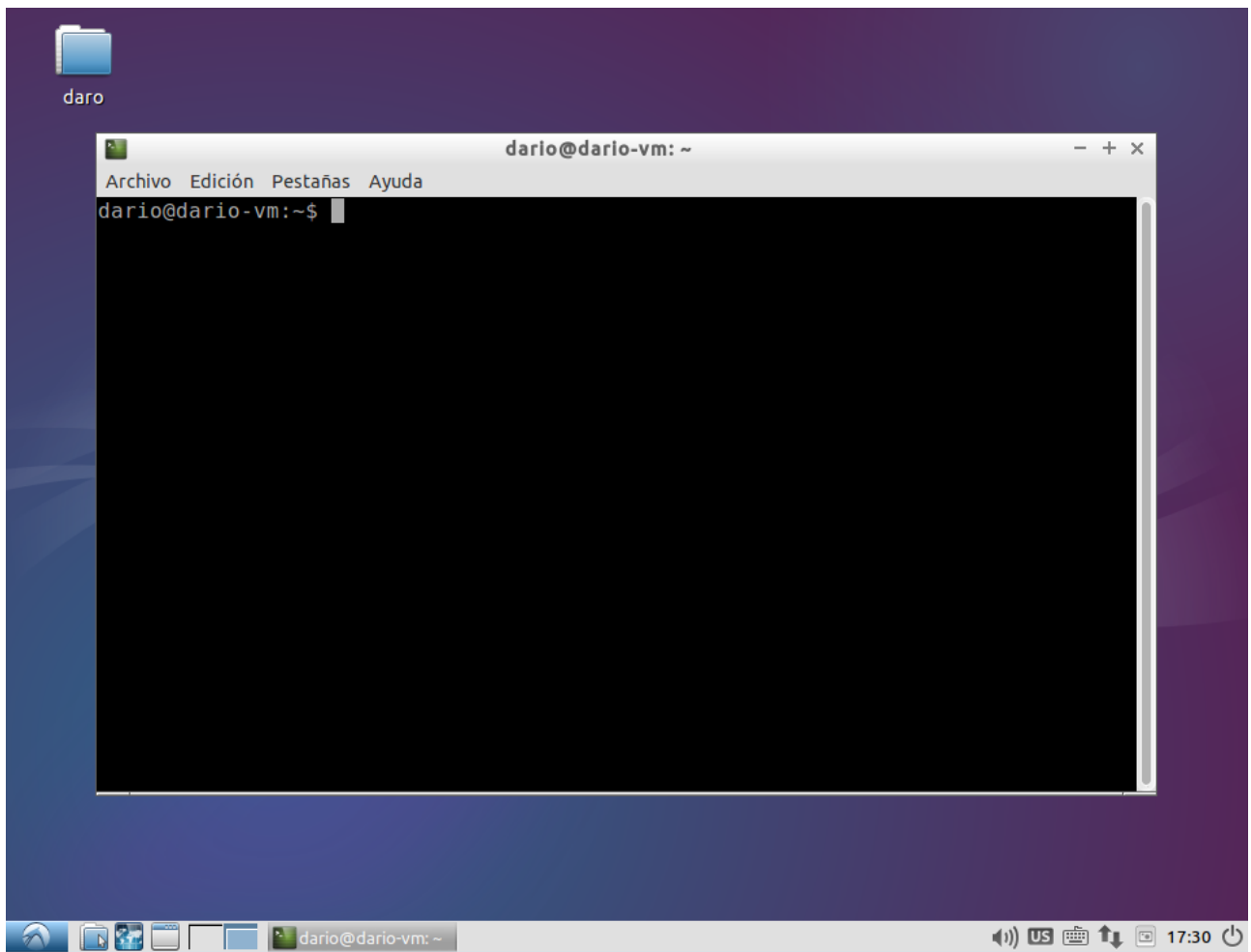
 home	Mantiene los archivos personales de los distintos usuarios del sistema a excepción del usuario root. Si por ejemplo tenemos los usuarios daro y pepe, ambos tendrán su propio espacio en el home, siendo su ubicación <b>/home/daro</b> y <b>/home/pepe</b> .
 media    mnt	Son puntos de montaje para anexar un sistema de archivo externo al propio sistema de archivos del sistema operativo, ya sean unidades usb, cdrom, etc., que se montarán automáticamente en /media, o se traten por ejemplo de sistemas de archivos que montaremos manualmente en /mnt.
 proc	Provee información sobre los procesos que se están ejecutando en el sistema operativo. Cabe destacar que es un directorio virtual, es decir el contenido está almacenado en memoria RAM y no en el disco.
 sbin	Así como el directorio /bin, almacena binarios (Ejecutables), pero que solo pueden ser ejecutados por el root (Administrador), por ejemplo herramientas de arranque, restauración, reparación, etc.
 sys	Similar a /proc, solo que aquí encontraremos información estructurada y jerárquica acerca del kernel, particiones, sistemas de archivo, drivers, etc.
 tmp	Dedicado a la creación de archivos temporales para el funcionamiento de los programas. Su contenido suele eliminarse tras cada reinicio del sistema.
 usr	Contiene binarios y archivos de solo lectura relativos a los programas de usuario. Bibliotecas, compiladores, herramientas de superusuario, entre otras.
 var	Contiene archivos de datos variables y temporales como por ejemplo los registros del sistema (logs), los registros de programas que tenemos instalados en el sistema operativo, archivos spool, etc.

### **Trabajando con la terminal**

La mayoría del tiempo estaremos utilizando la de Terminal de Linux, en general la encontraremos en “Accesorios”



A continuación aparecerá la gloriosa terminal



Lo que vemos en la terminal es el prompt, y será donde ingresaremos los comandos para comunicarnos con el interprete ( en nuestro caso BASH ). Dicho prompt está compuesto por el nombre de usuario + @ + el nombre del equipo en la red local + el directorio donde estoy parado + \$ (En este caso \$ indica que el usuario no tiene privilegios, si fuera el superusuario aparecería # ).

### Comando MAN:

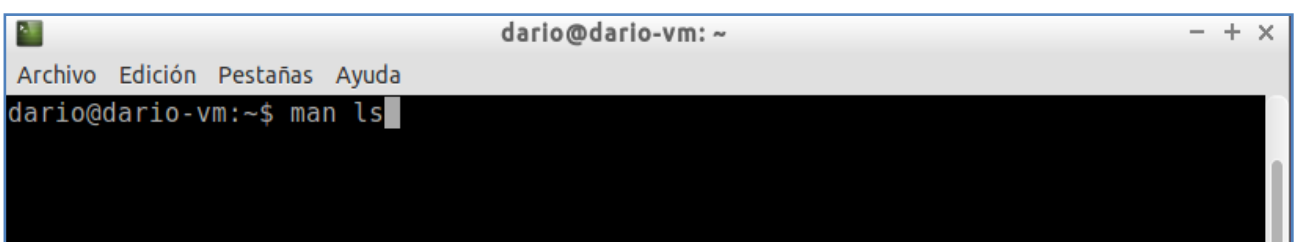
Muestra las paginas de ayuda del manual para el comando proporcionado

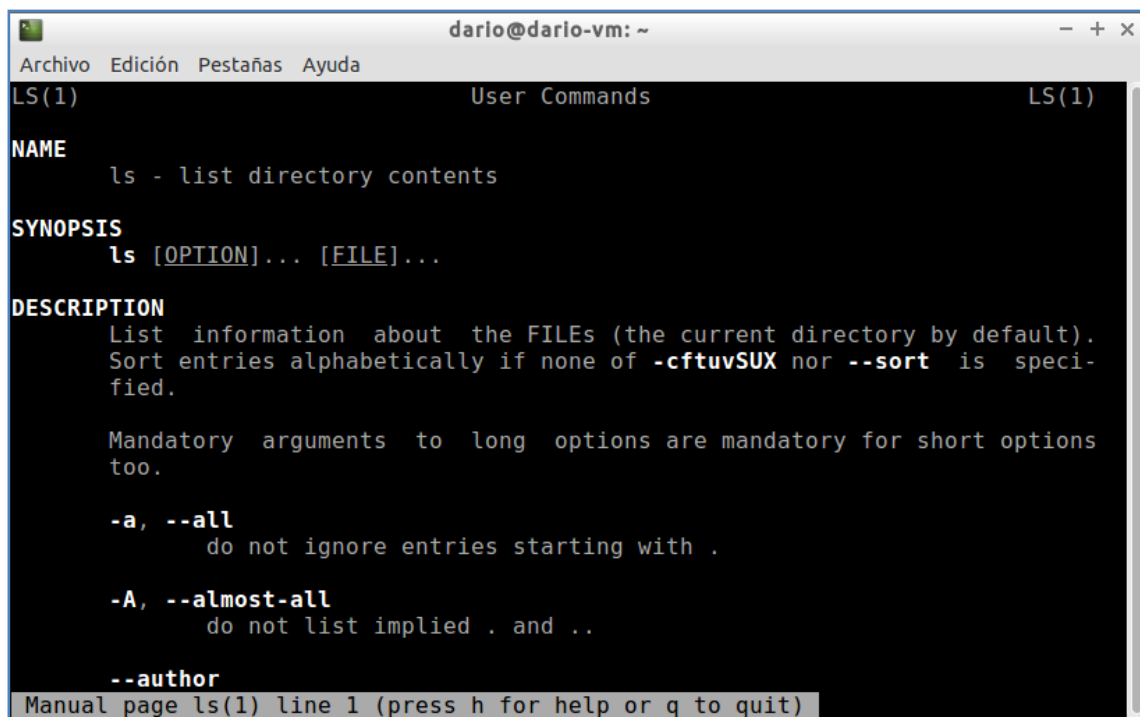
### Comando LS:

Lista el contenido de un directorio

Ingrese el siguiente comando: **man ls** [ENTER]

En este caso el comando es **man**, y **ls** está siendo pasado como parámetro al comando man, con lo cual lo que queremos es obtener ayuda sobre el comando ls, por último ingresamos el comando con la tecla ENTER.





```
darlo@darlo-vm: ~
Archivo Edición Pestañas Ayuda
LS(1) User Commands LS(1)

NAME
    ls - list directory contents

SYNOPSIS
    ls [OPTION]... [FILE]...

DESCRIPTION
    List information about the FILES (the current directory by default).
    Sort entries alphabetically if none of -cftuvSUX nor --sort is speci-
    fied.

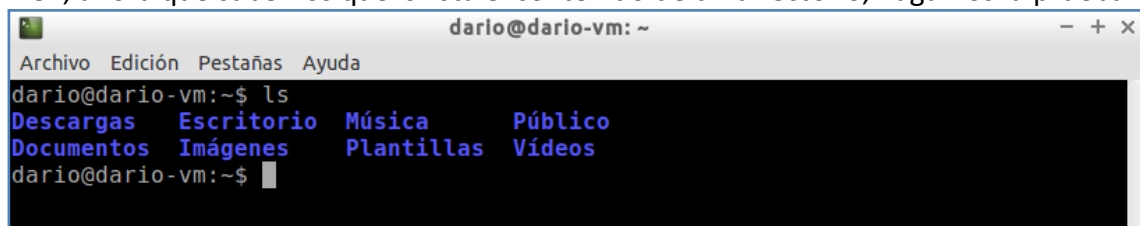
    Mandatory arguments to long options are mandatory for short options
    too.

    -a, --all
        do not ignore entries starting with .

    -A, --almost-all
        do not list implied . and ..

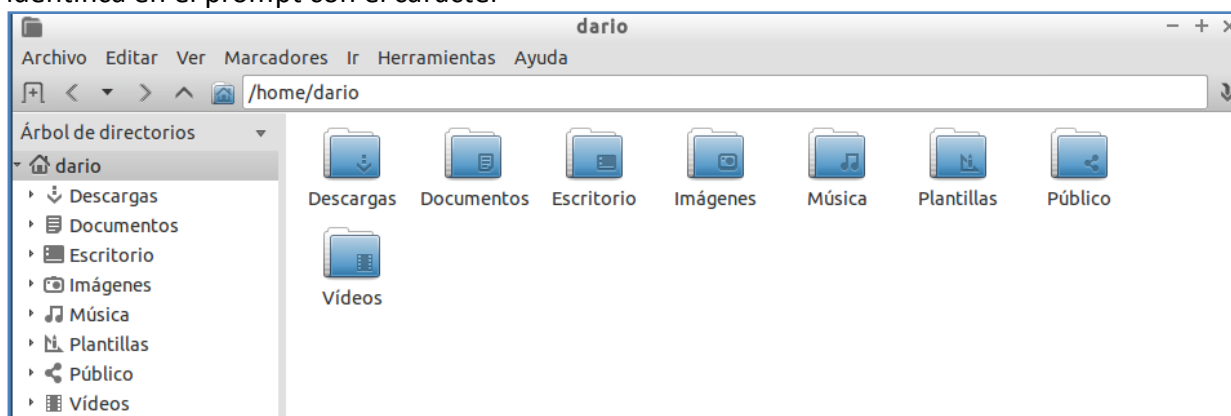
    --author
    Manual page ls(1) line 1 (press h for help or q to quit)
```

Bien, ahora que sabemos que ls lista el contenido de un directorio, hagamos la prueba:



```
darlo@darlo-vm: ~
Archivo Edición Pestañas Ayuda
darlo@darlo-vm:~$ ls
Descargas Escritorio Música Público
Documentos Imágenes Plantillas Vídeos
darlo@darlo-vm:~$
```

Nótese que se listó el contenido del home, esto es porque estamos parados en el home, esto se identifica en el prompt con el caracter ~



Cuando el comando **ls** se utiliza sin un directorio objetivo, lista el contenido del directorio actual, es decir donde estoy parado.

Un comando también puede recibir parámetros de “comportamiento” o “modificadores” que se especifican con un guión medio y una letra, por ejemplo si queremos listar el contenido de un directorio, pero ahora en formato de lista, podemos ejecutar con el modificador **-l**:

```
dario@dario-vm: ~
Archivo Edición Pestañas Ayuda
dario@dario-vm:~$ ls -l
total 32
drwxr-xr-x 3 dario dario 4096 mar 16 16:27 Descargas
drwxr-xr-x 2 dario dario 4096 nov 7 2020 Documentos
drwxr-xr-x 3 dario dario 4096 mar 16 12:35 Escritorio
drwxr-xr-x 2 dario dario 4096 oct 6 2020 Imágenes
drwxr-xr-x 2 dario dario 4096 oct 6 2020 Música
drwxr-xr-x 2 dario dario 4096 oct 6 2020 Plantillas
drwxr-xr-x 2 dario dario 4096 oct 6 2020 Público
drwxr-xr-x 2 dario dario 4096 oct 6 2020 Vídeos
dario@dario-vm:~$
```

Si quisiéramos listar el contenido del Escritorio, desde donde estamos parados:

```
dario@dario-vm: ~
Archivo Edición Pestañas Ayuda
dario@dario-vm:~$ ls -l Escritorio
total 4
drwxrwxr-x 17 dario dario 4096 mar 16 18:13 daro
dario@dario-vm:~$
```

### Comando CD

Con este comando podemos “navegar” por los distintos directorios desde la terminal, supongamos que queremos movernos al Escritorio, hacemos lo siguiente:

```
dario@dario-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
dario@dario-vm:~$ cd Escritorio
dario@dario-vm:~/Escritorio$
```

Ahora que estamos parados allí, crearemos un directorio nuevo y lo llamaremos pepe, esto lo hacemos con el comando **mkdir pepe** [ENTER].

### Comando MKDIR

Con este comando podemos crear un nuevo directorio

```
dario@dario-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
dario@dario-vm:~/Escritorio$ mkdir pepe
dario@dario-vm:~/Escritorio$ ls
daro pepe
dario@dario-vm:~/Escritorio$
```

### Subdirectorios . y ..

A nuestros ojos, el directorio pepe estará vacío, pero en realidad cuando se crea un directorio, también se crean dos subdirectorios, los subdirectorios . (punto) y ..(punto punto), el primero es para hacer referencia al propio directorio (en este caso pepe) y el segundo para referenciar al directorio padre de pepe (en este caso el Escritorio).

Estos dos subdirectorios se encuentran ocultos, en Linux los archivos y directorios ocultos llevan un punto al comienzo del nombre. Podemos visualizar los archivos y directorios ocultos con el modificador -a (all) del comando ls, hagamos la prueba:

```
dario@dario-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
dario@dario-vm:~/Escritorio$ ls -a pepe
.  .  .
dario@dario-vm:~/Escritorio$
```

Si quisiéramos volver un nivel hacia atrás, en este caso al HOME (~), recuerden que estamos parados en el Escritorio, hacemos:

```
dario@dario-vm: ~
Archivo Edición Pestañas Ayuda
dario@dario-vm:~/Escritorio$ cd ..
dario@dario-vm:~$
```

Si quisiéramos obtener ayuda del comando cd con el comando man pasa lo siguiente

```
dario@dario-vm: ~
Archivo Edición Pestañas Ayuda
dario@dario-vm:~$ man cd
No existe entrada de manual para cd
dario@dario-vm:~$
```

Esto es porque cd es un comando interno, es decir está embebido en el bash, a diferencia de los comandos externos que se encuentran en el disco. Para obtener ayuda sobre los comandos internos utilizamos el comando interno help.

### Comando HELP

Ofrece ayuda sobre comandos internos

```
dario@dario-vm: ~
Archivo Edición Pestañas Ayuda
dario@dario-vm:~$ help cd
cd: cd [-L|[-P [-e]] [-@]] [dir]
    Change the shell working directory.

    Change the current directory to DIR.  The default DIR is the value of the
    HOME shell variable.
```

### Comando TYPE

Con el comando type podemos obtener ayuda sobre el tipo de comando

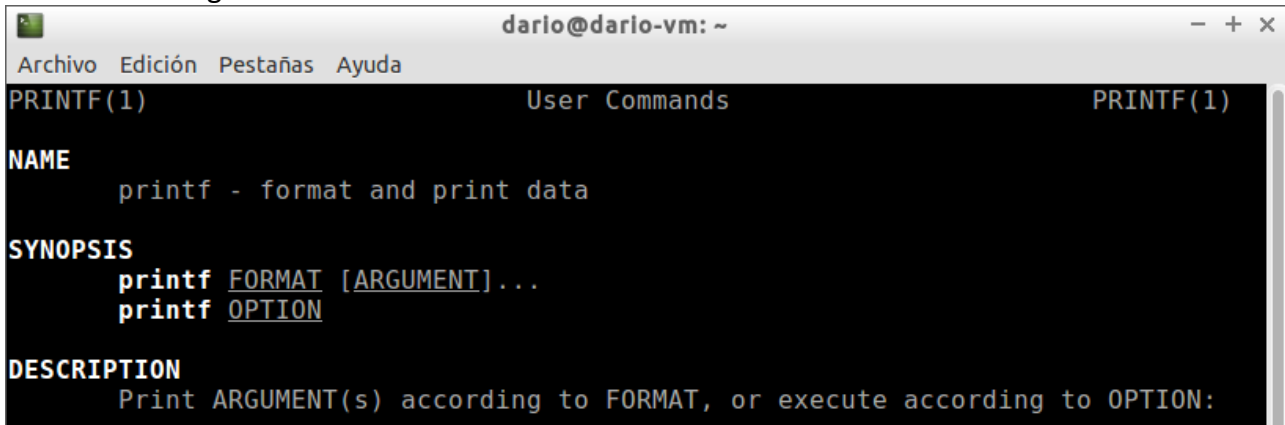
```
dario@dario-vm: ~
Archivo Edición Pestañas Ayuda
dario@dario-vm:~$ type cd
cd es una orden interna del shell
dario@dario-vm:~$ type mkdir
mkdir is /bin/mkdir
```

Una última observación sobre el comando man, al buscar ayuda por ejemplo sobre el viejo conocido **printf**

```
dario@dario-vm: ~
Archivo Edición Pestañas Ayuda
dario@dario-vm:~$ man printf
```



obtenemos lo siguiente:



```
dario@dario-vm: ~
Archivo Edición Pestañas Ayuda
PRINTF(1) User Commands PRINTF(1)

NAME
    printf - format and print data

SYNOPSIS
    printf FORMAT [ARGUMENT]...
    printf OPTION

DESCRIPTION
    Print ARGUMENT(s) according to FORMAT, or execute according to OPTION:
```

Aquí podemos observar algunas cosas, primero vemos que es el comando printf de bash, NO la función que conocemos de ANSI C. Segundo; si bien es un comando de bash ( Interno ), estamos obteniendo ayuda con man, lo que quiere decir que también es externo, en definitiva; algunos comandos tienen implementación interna y externa, por defecto al ejecutar el comando tiene prioridad la implementación interna.



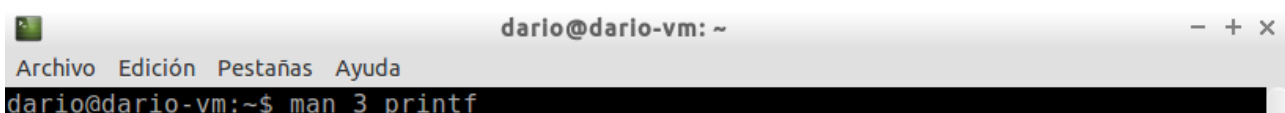
```
dario@dario-vm: ~
Archivo Edición Pestañas Ayuda
dario@dario-vm:~$ type printf
printf es una orden interna del shell
dario@dario-vm:~$ ls /usr/bin/printf
/usr/bin/printf
```

Con la ayuda del manual sobre el printf de bash, nos desplazamos hacia abajo, veremos el apartado SEE ALSO



```
dario@dario-vm: ~
Archivo Edición Pestañas Ayuda
SEE ALSO
    printf(3)
```

Vemos que informa como ayuda relacionada **printf(3)**. Lo que debemos hacer es invocar la ayuda de la siguiente manera:



```
dario@dario-vm: ~
Archivo Edición Pestañas Ayuda
dario@dario-vm:~$ man 3 printf
```

Ahora sí veremos la ayuda sobre la función printf de ANSI C:

```
darlo@darlo-vm: ~
Archivo Edición Pestañas Ayuda
PRINTF(3) Linux Programmer's Manual PRINTF(3)

NAME
    printf, fprintf, sprintf, snprintf, vprintf, vfprintf, vsprintf,
    vsnprintf - formatted output conversion

SYNOPSIS
    #include <stdio.h>

    int printf(const char *format, ...);
    int fprintf(FILE *stream, const char *format, ...);
    int sprintf(char *str, const char *format, ...);
    int snprintf(char *str, size_t size, const char *format, ...);

    #include <stdarg.h>

    int vprintf(const char *format, va_list ap);
    int vfprintf(FILE *stream, const char *format, va_list ap);
    int vsprintf(char *str, const char *format, va_list ap);
    int vsnprintf(char *str, size_t size, const char *format, va_list ap);
```

### Redirección de flujo:

Existe la posibilidad de transformar la salida de un comando en la entrada de otro redireccionando el flujo de datos con el carácter | (pipe), por ejemplo:

```
darlo@darlo-vm: ~/Escritorio/pepe
Archivo Edición Pestañas Ayuda
darlo@darlo-vm:~/Escritorio/pepe$ ls -l
total 128
-rw-rw-r-- 1 darlo darlo 127439 may 16 2021 imagen.png
darlo@darlo-vm:~/Escritorio/pepe$ ls -l | head -1
total 128
darlo@darlo-vm:~/Escritorio/pepe$
```

### Comando HEAD

Visualiza las primeras diez líneas de la entrada recibida, también se puede especificar la cantidad de líneas a visualizar. También existe el **comando TAIL**, que muestra las últimas diez líneas.

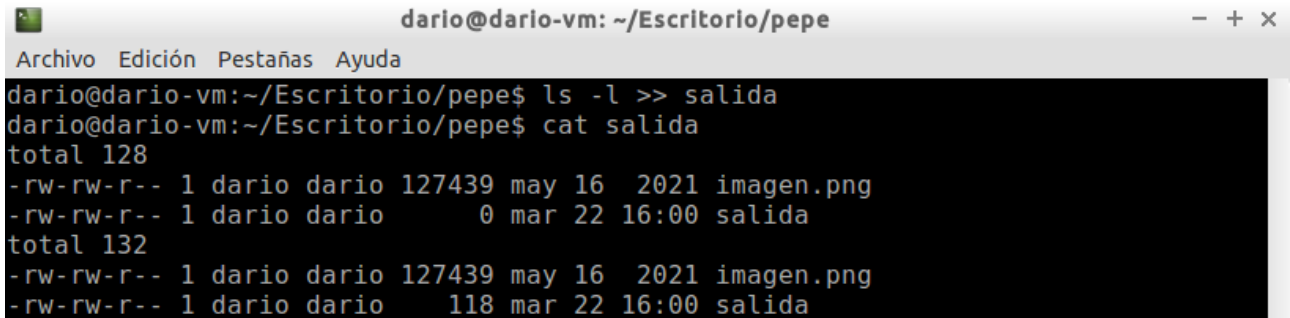
En el ejemplo anterior, la salida del comando ls es enviada a través de una tubería hacia el comando head, el cual muestra solo la primer línea de esa entrada ya que se especifica la opción -1.

También podemos redireccionar el flujo hacia un archivo con los operadores > y >>.

En el caso del operador > si el archivo no existe, lo crea y si existe lo sobrescribe. Por ejemplo:

```
darlo@darlo-vm: ~/Escritorio/pepe
Archivo Edición Pestañas Ayuda
darlo@darlo-vm:~/Escritorio/pepe$ ls -l > salida
darlo@darlo-vm:~/Escritorio/pepe$ cat salida
total 128
-rw-rw-r-- 1 darlo darlo 127439 may 16 2021 imagen.png
-rw-rw-r-- 1 darlo darlo 0 mar 22 16:00 salida
```

En el caso del operador >> si el archivo no existe lo crea y si existe agrega la información al final. Por ejemplo:



```
dario@dario-vm: ~/Escritorio/pepe
Archivo Edición Pestañas Ayuda
dario@dario-vm:~/Escritorio/pepe$ ls -l >> salida
dario@dario-vm:~/Escritorio/pepe$ cat salida
total 128
-rw-rw-r-- 1 dario dario 127439 may 16 2021 imagen.png
-rw-rw-r-- 1 dario dario 0 mar 22 16:00 salida
total 132
-rw-rw-r-- 1 dario dario 127439 may 16 2021 imagen.png
-rw-rw-r-- 1 dario dario 118 mar 22 16:00 salida
```

Nótese que el archivo salida es listado también incluso cuando no existía, esto es porque bash abre (si no existe crea) todos los archivos involucrados antes de ejecutar el comando, en este caso ls y después hace la redirección.

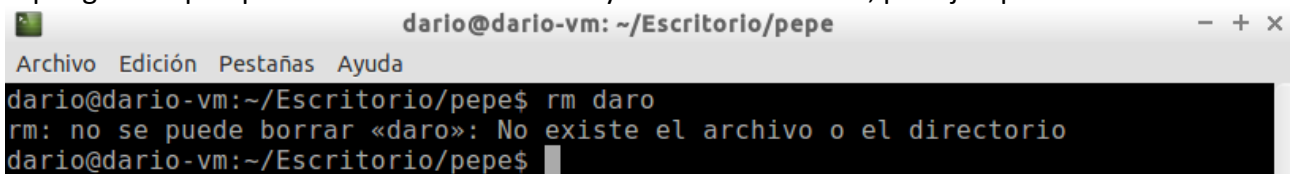
En el ejemplo se utiliza el **comando CAT** para visualizar por pantalla el contenido del archivo salida.

### Comando CAT

En el caso de suministrar varios archivos al comando, concatena el contenido de los mismos y visualiza el resultado por pantalla, en el caso de proveer un solo archivo al comando se visualiza en pantalla el contenido del mismo.

También se pueden redireccionar las corrientes stdin, stdout y stderr (entrada, salida y salida con error), estas corrientes están identificadas con los números 0, 1 y 2 respectivamente.

Supongamos que queremos borrar un archivo y el mismo no existe, por ejemplo:

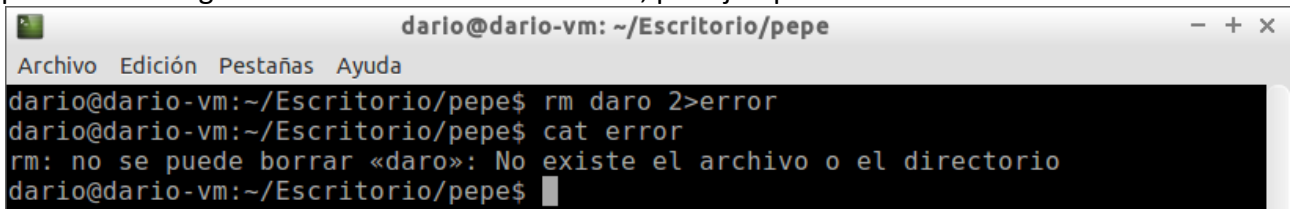


```
dario@dario-vm: ~/Escritorio/pepe
Archivo Edición Pestañas Ayuda
dario@dario-vm:~/Escritorio/pepe$ rm daro
rm: no se puede borrar «daro»: No existe el archivo o el directorio
dario@dario-vm:~/Escritorio/pepe$
```

### Comando RM

Permite eliminar un archivo o directorio, en el caso de ser un directorio debemos indicar el parámetro opcional -r.

En el ejemplo vemos que la ejecución del comando arroja un mensaje de error por pantalla, esa es la corriente stderr (2), si no quisiéramos que se visualice ese error por la razón que fuere, podemos redirigir la corriente stderr a un archivo, por ejemplo:



```
dario@dario-vm: ~/Escritorio/pepe
Archivo Edición Pestañas Ayuda
dario@dario-vm:~/Escritorio/pepe$ rm daro 2>error
dario@dario-vm:~/Escritorio/pepe$ cat error
rm: no se puede borrar «daro»: No existe el archivo o el directorio
dario@dario-vm:~/Escritorio/pepe$
```

Nótese que el mensaje de error quedó almacenado en el archivo llamado "error". Para este tipo de salidas las cuales no nos interesan, el sistema tiene un archivo especial llamado null que se encuentra en /dev, es una buena practica redireccionar este tipo de salidas a /dev/null, hablando

mal y pronto es una suerte de “agujero negro” donde lo que allí se almacena ya no se puede recuperar.

### **Comando TOUCH**

El comando touch en realidad permite cambiar la fecha de acceso y modificación de archivos y directorios, pero también lo podemos utilizar para crear archivos vacíos.

Existen un parámetro opcional `-v` (verbose) que suele estar presente en la mayoría de los comandos, que hace que el comando especifique por pantalla que acciones está llevando a cabo en cada momento. Por ejemplo para el borrado, suponiendo ahora que el archivo si existe:

```
dario@dario-vm: ~/Escritorio/pepe
Archivo Edición Pestañas Ayuda
dario@dario-vm:~/Escritorio/pepe$ touch daro
dario@dario-vm:~/Escritorio/pepe$ rm -v daro
«daro» borrado
dario@dario-vm:~/Escritorio/pepe$
```

Nótese que creamos el archivo daro con touch, lo borramos especificando verbose al comando rm y por lo tanto visualiza por pantalla el borrado realizado.

Cabe aclarar que también es posible crear un archivo vacío con el operador `>`, por ejemplo:

```
dario@dario-vm: ~/Escritorio/pepe
Archivo Edición Pestañas Ayuda
dario@dario-vm:~/Escritorio/pepe$ > daro2
```

Ahora supongamos que queremos borrar un conjunto de archivos, de los cuales no sabemos si existen o no. Por ejemplo:

```
dario@dario-vm: ~/Escritorio/pepe
Archivo Edición Pestañas Ayuda
dario@dario-vm:~/Escritorio/pepe$ touch daro
dario@dario-vm:~/Escritorio/pepe$ rm -v daro pepe
«daro» borrado
rm: no se puede borrar «pepe»: No existe el archivo o el directorio
dario@dario-vm:~/Escritorio/pepe$
```

Como vemos arroja tanto la stdout (el borrado) y la stderr (el error) a la stdout, pero si quisiéramos enviar a modo de log, la salida con los archivos borrados y los que no pudieron ser borrados (error) a un archivo, podemos hacer lo siguiente:

```
dario@dario-vm: ~/Escritorio/pepe
Archivo Edición Pestañas Ayuda
dario@dario-vm:~/Escritorio/pepe$ touch daro daro2
dario@dario-vm:~/Escritorio/pepe$ rm -v daro pepe daro2 > resultado 2>&1
dario@dario-vm:~/Escritorio/pepe$ cat resultado
«daro» borrado
rm: no se puede borrar «pepe»: No existe el archivo o el directorio
«daro2» borrado
dario@dario-vm:~/Escritorio/pepe$
```

Creamos de nuevo daro y un nuevo archivo daro2, para que se ejemplifique mejor. En la ejecución del comando nótese que se redirige la stdout del comando al archivo resultado y por último

(2>&1) especifica redirigir la stderr (2) hacia donde esté apuntando la stdout (1), que está apuntando al archivo resultado porque lo especificamos con (> resultado).

Y si quisiéramos enviar a archivos separados, lo hacemos así:

```
dario@dario-vm: ~/Escritorio/pepe
Archivo Edición Pestañas Ayuda
dario@dario-vm:~/Escritorio/pepe$ touch daro daro2
dario@dario-vm:~/Escritorio/pepe$ rm -v daro pepe daro2 > borrados 2>con_error
dario@dario-vm:~/Escritorio/pepe$ cat borrados
«daro» borrado
«daro2» borrado
dario@dario-vm:~/Escritorio/pepe$ cat con_error
rm: no se puede borrar «pepe»: No existe el archivo o el directorio
dario@dario-vm:~/Escritorio/pepe$
```

También se puede redirigir la stdin desde un archivo hacia un comando con el operador <, siendo el contenido del archivo la entrada del comando, por ejemplo:

```
dario@dario-vm: ~/Escritorio/pepe
Archivo Edición Pestañas Ayuda
dario@dario-vm:~/Escritorio/pepe$ cat < borrados
«daro» borrado
«daro2» borrado
dario@dario-vm:~/Escritorio/pepe$
```

También podemos redireccionar datos con los operadores << y <<<.

Para ejemplificar esto vemos el **comando WC**

### **Comando WC**

El comando wc cuenta la cantidad de líneas, palabras y caracteres contenidos en la entrada proporcionada.

Por ejemplo para el archivo letras:

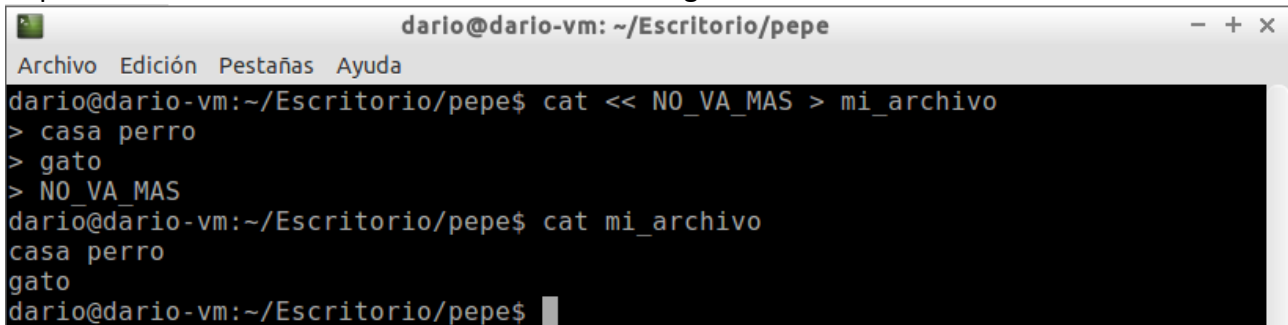
```
dario@dario-vm: ~/Escritorio/pepe
Archivo Edición Pestañas Ayuda
dario@dario-vm:~/Escritorio/pepe$ cat letras
a b c
d e
f
dario@dario-vm:~/Escritorio/pepe$ wc letras
 3  6 12 letras
dario@dario-vm:~/Escritorio/pepe$
```

Ahora, podríamos especificar la entrada con el operador << y no desde un archivo:

```
dario@dario-vm: ~/Escritorio/pepe
Archivo Edición Pestañas Ayuda
dario@dario-vm:~/Escritorio/pepe$ wc << fin
> 1 2 3
> 4 5
> 6
> fin
 3  6 12
dario@dario-vm:~/Escritorio/pepe$
```

El comando `wc` esperará el ingreso de datos desde el teclado, la palabra `fin` auspicia de delimitador de la entrada de datos (`fin` es un ejemplo, puede ser cualquier palabra), cuando se ingrese la palabra `fin` finalizará la lectura de datos y se ejecutará el comando.

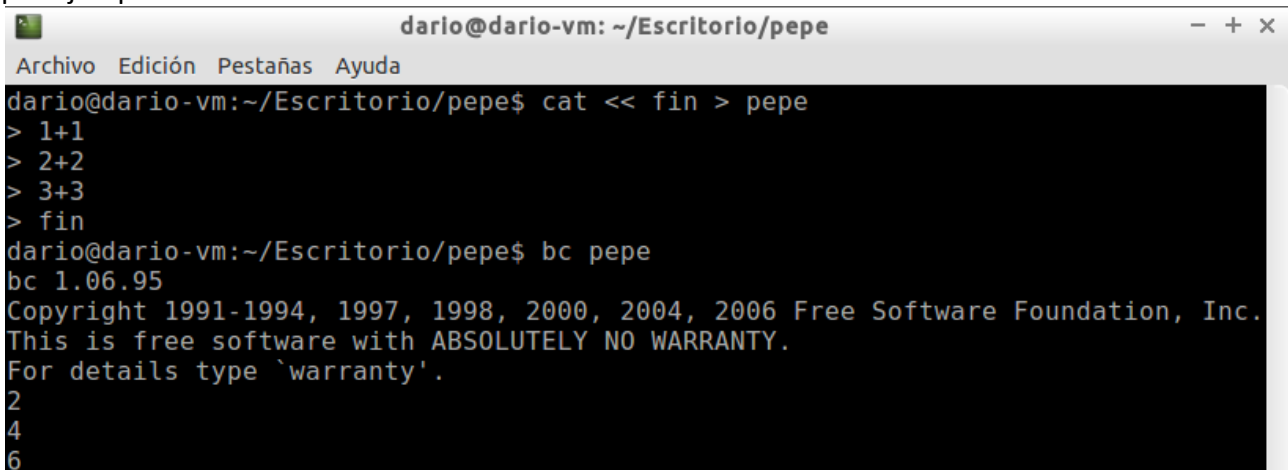
Si quisiéramos crear un nuevo archivo realizando carga de datos:



```
darío@darío-vm: ~/Escritorio/pepe
Archivo Edición Pestañas Ayuda
darío@darío-vm:~/Escritorio/pepe$ cat << NO_VA_MAS > mi_archivo
> casa perro
> gato
> NO_VA_MAS
darío@darío-vm:~/Escritorio/pepe$ cat mi_archivo
casa perro
gato
darío@darío-vm:~/Escritorio/pepe$
```

Por último el operador `<<<` es similar, solo que debemos proveer una cadena con los datos que necesita el comando.

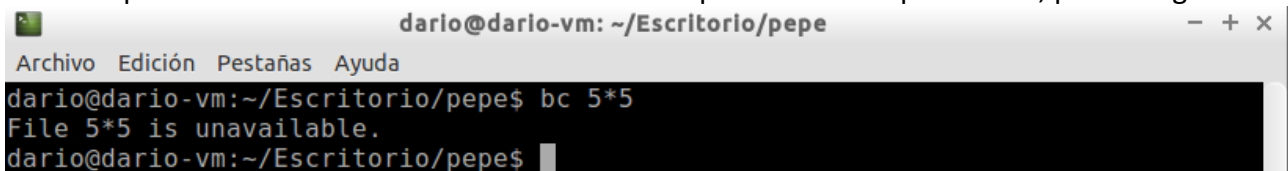
Existe un comando para realizar operaciones matemáticas y comparaciones, el **comando BC**, para el ejemplo utilizaremos dicho comando, en general se le debe proveer de un lote de operaciones, por ejemplo:



```
darío@darío-vm: ~/Escritorio/pepe
Archivo Edición Pestañas Ayuda
darío@darío-vm:~/Escritorio/pepe$ cat << fin > pepe
> 1+1
> 2+2
> 3+3
> fin
darío@darío-vm:~/Escritorio/pepe$ bc pepe
bc 1.06.95
Copyright 1991-1994, 1997, 1998, 2000, 2004, 2006 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type `warranty'.
2
4
6
```

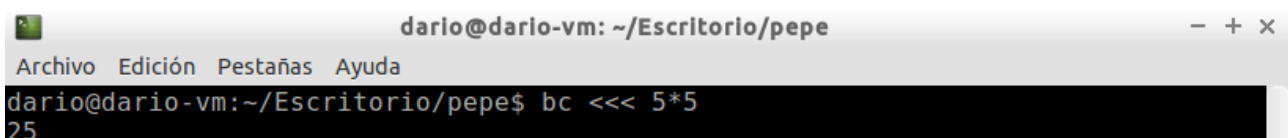
Nótese que se genera un archivo (`pepe`) con las operaciones y luego se ejecuta el comando `bc` con el archivo `pepe`, generando la salida con el resultado.

Ahora si quisiéramos utilizar el comando `bc` con una operación como parámetro, pasa lo siguiente:



```
darío@darío-vm: ~/Escritorio/pepe
Archivo Edición Pestañas Ayuda
darío@darío-vm:~/Escritorio/pepe$ bc 5*5
File 5*5 is unavailable.
darío@darío-vm:~/Escritorio/pepe$
```

Dado que espera un archivo con las operaciones, no reconoce la operación pasada como parámetro. Podemos solucionar esto pasando la operación con el carácter `<<<`



```
darío@darío-vm: ~/Escritorio/pepe
Archivo Edición Pestañas Ayuda
darío@darío-vm:~/Escritorio/pepe$ bc <<< 5*5
25
```

## Tipos de archivo:

En MS Windows estamos acostumbrados a identificar el tipo de archivo por su extensión, de hecho el sistema operativo así lo hace, pero en Linux los archivos no tienen extensión, es decir podemos como usuarios poner una extensión en el nombre e incluso la interfaz gráfica asocia un icono, pero el sistema no lo identifica por extensión, sino que utiliza lo que se conoce como **magic number**, que en este caso está representado por una serie de bytes en el encabezado del archivo. Supongamos un archivo png, con el comando hexdump podemos visualizar los bytes del archivo a modo de visor hexadecimal:

```
dario@dario-vm:~/Escritorio/pepe$ hexdump -C imagen.png | more
00000000  89 50 4e 47 0d 0a 1a 0a 00 00 00 0d 49 48 44 52  .PNG .....IHDR|
00000010  00 00 05 b2 00 00 02 ba 08 02 00 00 00 63 73 3d  .....CS=|
00000020  39 00 00 00 03 73 42 49 54 08 08 08 db e1 4f e0  9....sBIT....0.|
00000030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  . . . . .
```

En éste caso los cuatro primeros bytes del archivo lo identifican como un archivo de tipo png.

## Comando FILE

Una forma mas cómoda de identificar el tipo de archivo es con el comando **file**:

```
dario@dario-vm: ~/Escritorio/pepe
Archivo Edición Pestañas Ayuda
dario@dario-vm:~/Escritorio/pepe$ file imagen.png
imagen.png: PNG image data, 1458 x 698, 8-bit/color RGB, non-interlaced
dario@dario-vm:~/Escritorio/pepe$ file --mime-type imagen.png
imagen.png: image/png
dario@dario-vm:~/Escritorio/pepe$ file -b --mime-type imagen.png
image/png
dario@dario-vm:~/Escritorio/pepe$
```

## Permisos sobre archivos:

Cuando ejecutamos el comando **ls -l** podemos visualizar los permisos del elemento y su tipo (archivo/directorio/etc.) observando los primeros 10 caracteres. Por ejemplo:

```
dario@dario-vm: ~/Escritorio/pepe
Archivo Edición Pestañas Ayuda
dario@dario-vm:~/Escritorio/pepe$ ls -l
total 128
-rw-rw-r-- 1 dario dario 127439 may 16 2021 imagen.png
dario@dario-vm:~/Escritorio/pepe$
```

**Type**

```
-rw-rw-r-- 1 dario dario 127439 may 16 2021 imagen.png
```

Diagram illustrating the permissions **-rw-rw-r--**:

- The first character **-** indicates the file type (regular file).
- The next three characters **rw-rw-r--** represent permissions for different groups:
- rw** (User): Read and Write permissions for the User.
- rw** (Group): Read and Write permissions for the Group.
- r--** (Others): Read permission for Others.

El primer carácter indica el tipo de elemento, a continuación se presentan los tipos posibles:







En definitiva, para un directorio; r y w permiten operaciones de lectura/escritura sobre el mismo, x permite acceder al directorio y si también están presentes r y/o w se permitirán operaciones de lectura/escritura sobre el contenido del directorio.

### Comando CHMOD

Permite cambiar los permisos de un elemento.

El comando chmod soporta varias formas de manipular los permisos, en este caso utilizaremos la forma octal. Si vemos a los permisos como un conjunto de bits y a su vez a estos bits los representamos en base octal, para el ejemplo anterior tendríamos para el archivo **texto**:

User			Group			Others		
r	w	-	r	w	-	r	-	-
1	1	1	1	1	1	1	0	1
7			7			5		

Si quisiéramos por ejemplo asignar todos los permisos al user y quitar todos los permisos a los demás usuarios, ejecutamos lo siguiente:

```
darlo@darlo-vm: ~/Escritorio/pepe
Archivo Edición Pestañas Ayuda
darlo@darlo-vm:~/Escritorio/pepe$ chmod 700 imagen.png
darlo@darlo-vm:~/Escritorio/pepe$ ls -l
total 128
-rwx----- 1 darlo darlo 127439 mar 25 12:55 imagen.png
darlo@darlo-vm:~/Escritorio/pepe$
```

La otra forma funciona indicando el tipo de usuario con los caracteres u,g,o,(a para todos) seguido de +,-,= para otorgar, quitar o establecer un conjunto de permisos. Por ejemplo para otorgar permiso de ejecución al user, sería u+x, si quisiéramos quitar los permisos de ejecución para todos los usuarios, hacemos a-x.

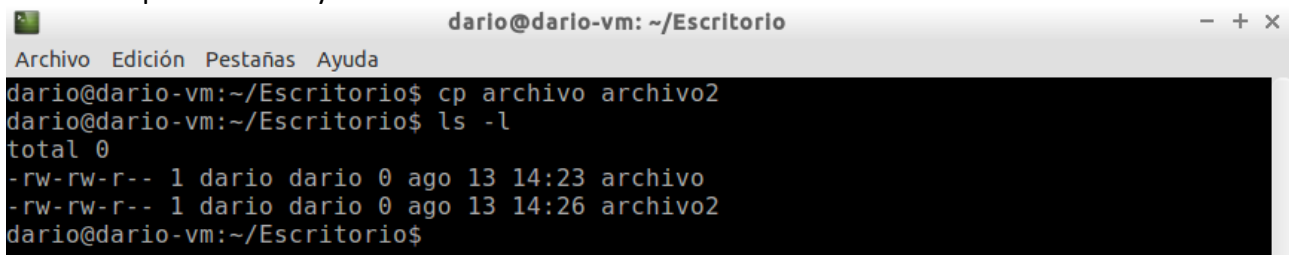
Para el ejemplo sería:

```
darlo@darlo-vm: ~/Escritorio/pepe
Archivo Edición Pestañas Ayuda
darlo@darlo-vm:~/Escritorio/pepe$ chmod u=rwx,go= imagen.png
darlo@darlo-vm:~/Escritorio/pepe$ ls -l
total 128
-rwx----- 1 darlo darlo 127439 mar 25 12:55 imagen.png
darlo@darlo-vm:~/Escritorio/pepe$
```

### Mas comandos:

#### **CP:**

Permite copiar archivos y directorios.



```
dario@dario-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
dario@dario-vm:~/Escritorio$ cp archivo archivo2
dario@dario-vm:~/Escritorio$ ls -l
total 0
-rw-rw-r-- 1 dario dario 0 ago 13 14:23 archivo
-rw-rw-r-- 1 dario dario 0 ago 13 14:26 archivo2
dario@dario-vm:~/Escritorio$
```

#### **MKDIR:**

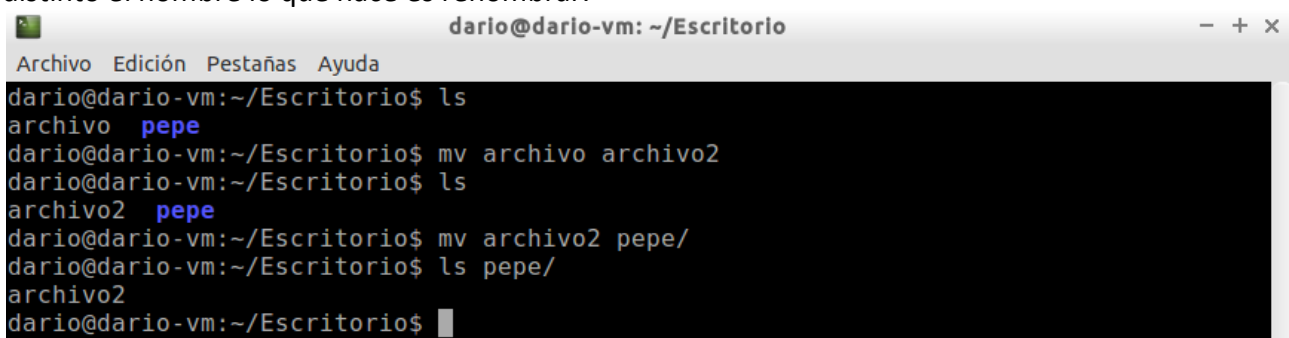
Permite crear directorios.



```
dario@dario-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
dario@dario-vm:~/Escritorio$ mkdir dir1
dario@dario-vm:~/Escritorio$ ls -l
total 4
-rw-rw-r-- 1 dario dario 0 ago 13 14:23 archivo
-rw-rw-r-- 1 dario dario 0 ago 13 14:26 archivo2
drwxrwxr-x 2 dario dario 4096 ago 13 14:29 dir1
dario@dario-vm:~/Escritorio$
```

#### **MV:**

Permite mover archivos o directorios, en el caso de mover con destino igual al origen, siendo distinto el nombre lo que hace es renombrar.

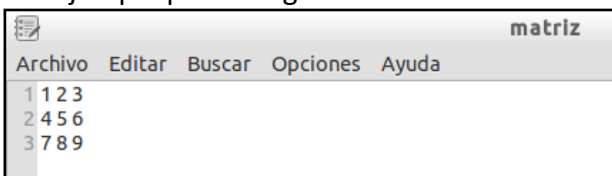


```
dario@dario-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
dario@dario-vm:~/Escritorio$ ls
archivo pepe
dario@dario-vm:~/Escritorio$ mv archivo archivo2
dario@dario-vm:~/Escritorio$ ls
archivo2 pepe
dario@dario-vm:~/Escritorio$ mv archivo2 pepe/
dario@dario-vm:~/Escritorio$ ls pepe/
archivo2
dario@dario-vm:~/Escritorio$
```

#### **CUT:**

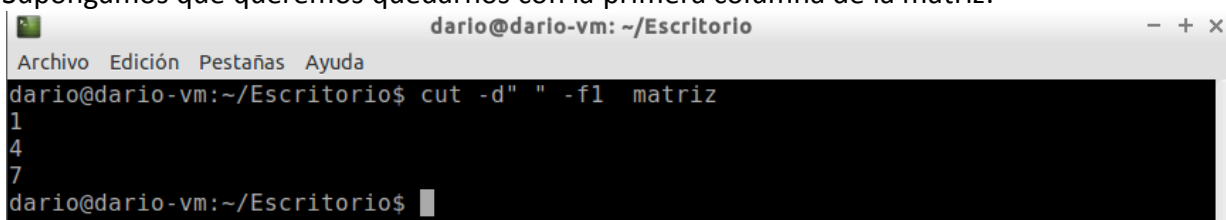
Permite cortar partes de cada línea de un archivo o una entrada de texto.

Por ejemplo para el siguiente archivo



```
matriz
Archivo Editar Buscar Opciones Ayuda
1 1 2 3
2 4 5 6
3 7 8 9
```

Supongamos que queremos quedarnos con la primera columna de la matriz:



```
dario@dario-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
dario@dario-vm:~/Escritorio$ cut -d" " -f1 matriz
1
4
7
dario@dario-vm:~/Escritorio$
```

Observamos en la ejecución del comando lo siguiente:

-d" " (Especifica que el delimitador de campos será el espacio)

-f1 (Especifica que queremos cortar el primer (1) campo (field) de cada línea)

Matriz (Especifica el archivo de entrada con los datos)

El comando trabaja sobre todas las líneas.

Si quisiéramos quedarnos con la columna 1 y 3 de la matriz:

```
dario@dario-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
dario@dario-vm:~/Escritorio$ cut -d" " -f1,3 matriz
1 3
4 6
7 9
dario@dario-vm:~/Escritorio$
```

También se puede especificar un rango de campos:

```
dario@dario-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
dario@dario-vm:~/Escritorio$ cut -d" " -f2-3 matriz
2 3
5 6
8 9
dario@dario-vm:~/Escritorio$
```

```
dario@dario-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
dario@dario-vm:~/Escritorio$ cut -d" " -f1,2-3 matriz
1 2 3
4 5 6
7 8 9
dario@dario-vm:~/Escritorio$
```

El rango puede ir hasta el final si no se especifica el limite de fin, por ejemplo:

```
dario@dario-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
dario@dario-vm:~/Escritorio$ cut -d" " -f1- matriz
1 2 3
4 5 6
7 8 9
dario@dario-vm:~/Escritorio$
```

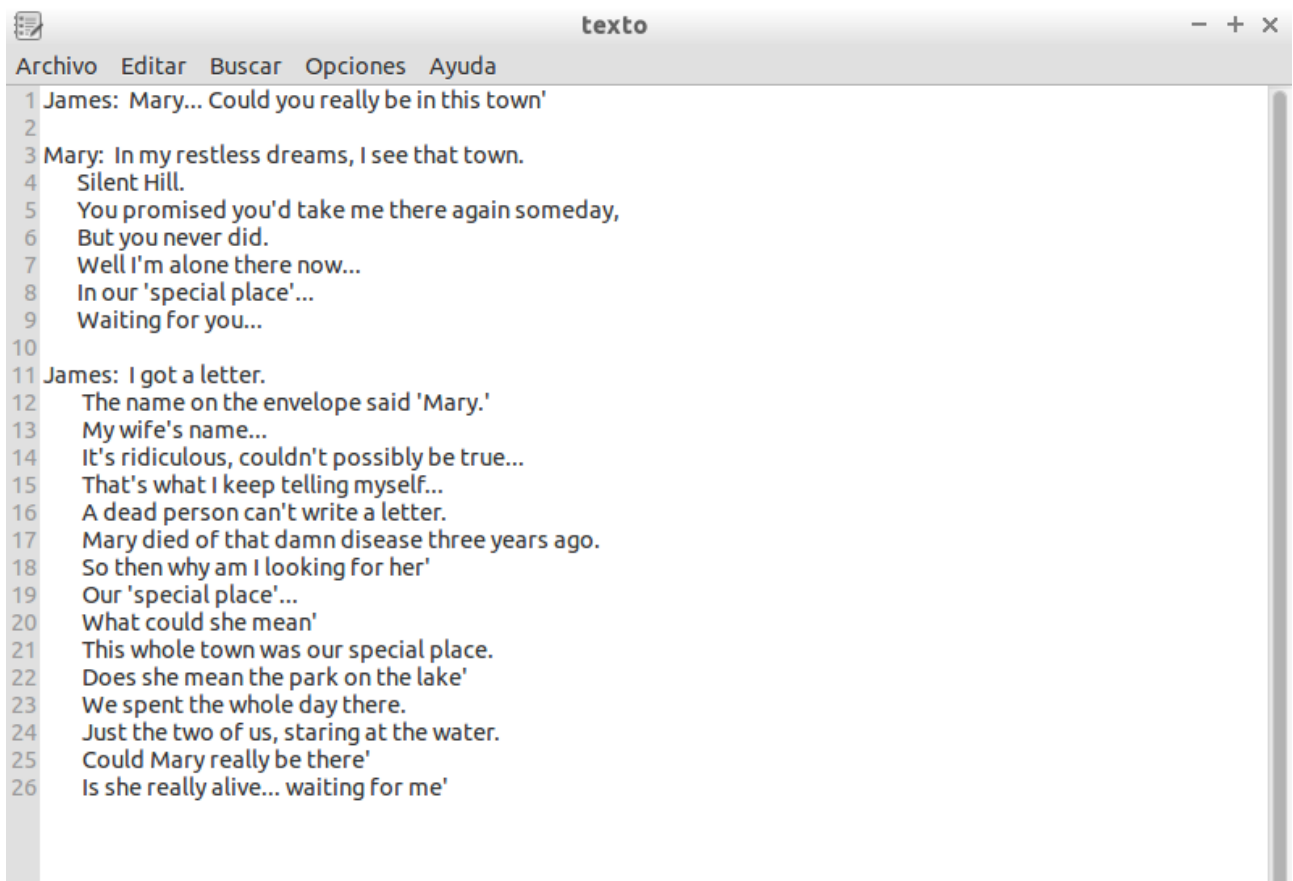
Lo mismo aplica si se especifica el limite final y no el inicial:

```
dario@dario-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
dario@dario-vm:~/Escritorio$ cut -d" " -f-2 matriz
1 2
4 5
7 8
dario@dario-vm:~/Escritorio$
```

## GREP:

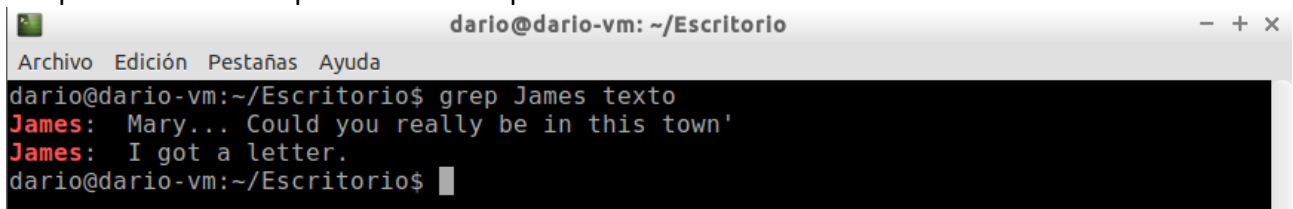
Muestra las líneas que cumplen con un patrón dado.

Trabajemos con el siguiente texto:



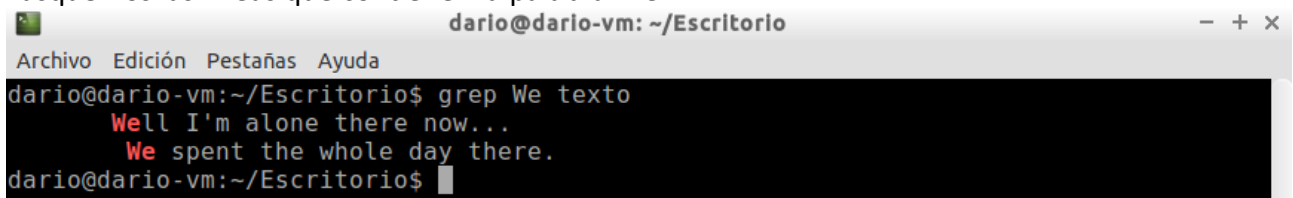
```
1 James: Mary... Could you really be in this town'
2
3 Mary: In my restless dreams, I see that town.
4   Silent Hill.
5   You promised you'd take me there again someday,
6   But you never did.
7   Well I'm alone there now...
8   In our 'special place'...
9   Waiting for you...
10
11 James: I got a letter.
12   The name on the envelope said 'Mary.'
13   My wife's name...
14   It's ridiculous, couldn't possibly be true...
15   That's what I keep telling myself...
16   A dead person can't write a letter.
17   Mary died of that damn disease three years ago.
18   So then why am I looking for her'
19   Our 'special place'...
20   What could she mean'
21   This whole town was our special place.
22   Does she mean the park on the lake'
23   We spent the whole day there.
24   Just the two of us, staring at the water.
25   Could Mary really be there'
26   Is she really alive... waiting for me'
```

Busquemos las líneas que contienen la palabra James



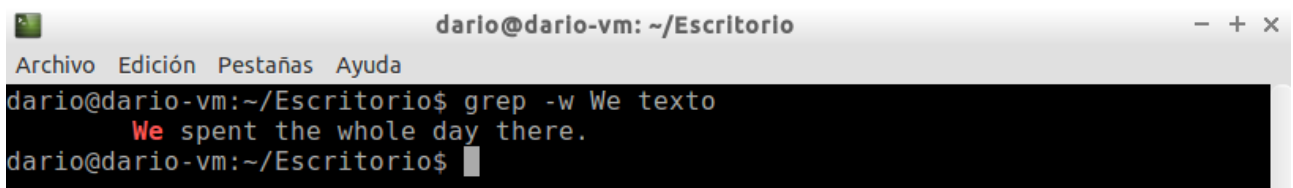
```
dario@dario-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
dario@dario-vm:~/Escritorio$ grep James texto
James: Mary... Could you really be in this town'
James: I got a letter.
dario@dario-vm:~/Escritorio$
```

Busquemos las líneas que contienen la palabra We



```
dario@dario-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
dario@dario-vm:~/Escritorio$ grep We texto
Well I'm alone there now...
We spent the whole day there.
dario@dario-vm:~/Escritorio$
```

Vemos que también trajo la palabra Well, por contener We, tendremos que ser mas específicos. Con el modificador `-w` le decimos a grep que busque la cadena exacta.



```
dario@dario-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
dario@dario-vm:~/Escritorio$ grep -w We texto
We spent the whole day there.
dario@dario-vm:~/Escritorio$
```

También lo podemos hacer con una expresión regular:

```
darío@darío-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
darío@darío-vm:~/Escritorio$ grep "\<We\>" texto
    We spent the whole day there.
darío@darío-vm:~/Escritorio$
```

Ahora busquemos las líneas que comienzan con Mary:

```
darío@darío-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
darío@darío-vm:~/Escritorio$ grep Mary texto
James:  Mary... Could you really be in this town'
Mary:  In my restless dreams, I see that town.
       The name on the envelope said 'Mary.'
       Mary died of that damn disease three years ago.
       Could Mary really be there'
darío@darío-vm:~/Escritorio$
```

Esto no nos sirve ya que trae cada línea donde aparece Mary, entonces podemos especificar mediante una expresión regular a través del carácter ^ que solo traiga la/s línea/s donde Mary sea la primera palabra:

```
darío@darío-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
darío@darío-vm:~/Escritorio$ grep "^Mary" texto
Mary:  In my restless dreams, I see that town.
darío@darío-vm:~/Escritorio$
```

Si quisiéramos además que muestre las líneas donde Mary es la primera palabra pero está antecedita por espacios como por ejemplo la línea 17 del archivo, la expresión sería así:

```
darío@darío-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
darío@darío-vm:~/Escritorio$ grep "[[:space:]]\{3,\}Mary" texto
    Mary died of that damn disease three years ago.
darío@darío-vm:~/Escritorio$
```

[[:space:]]      Buscar espacio en blanco

{3,}            Especifica que al menos haya 3 ocurrencias de espacios antes de Mary

Ahora juntamos ambas expresiones:

```
darío@darío-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
darío@darío-vm:~/Escritorio$ grep "^Mary\| [[:space:]]\{3,\}Mary" texto
Mary:  In my restless dreams, I see that town.
    Mary died of that damn disease three years ago.
darío@darío-vm:~/Escritorio$
```

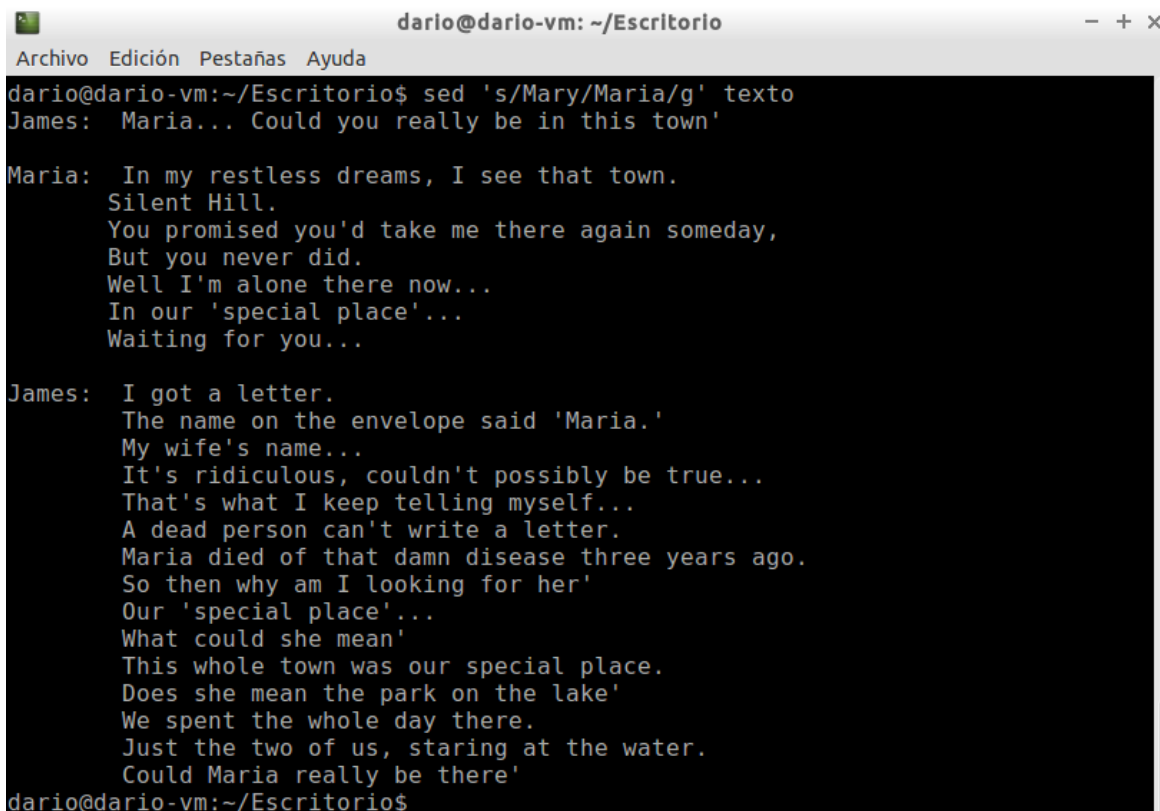
Nótese que unimos ambas expresiones con | , esto funciona como una Or

Si queremos contar las líneas donde a parece una palabra y donde no:

```
darío@darío-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
darío@darío-vm:~/Escritorio$ grep -c Mary texto
5
darío@darío-vm:~/Escritorio$ grep -vc Mary texto
21
darío@darío-vm:~/Escritorio$
```

## SED:

Editor de flujo de texto, en general utilizado para realizar un reemplazo en un texto dado. Por ejemplo, siguiendo con el archivo texto, si quisiéramos reemplazar Mary por Maria:



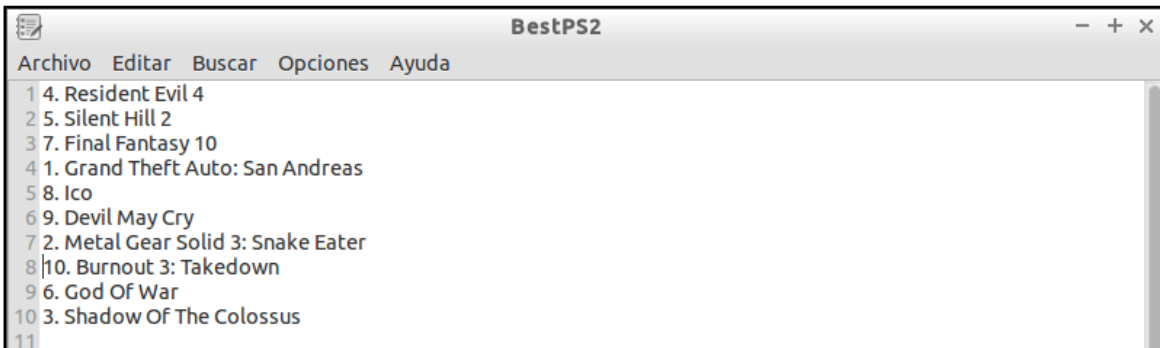
```
dario@dario-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
dario@dario-vm:~/Escritorio$ sed 's/Mary/Maria/g' texto
James: Maria... Could you really be in this town'

Maria: In my restless dreams, I see that town.
      Silent Hill.
      You promised you'd take me there again someday,
      But you never did.
      Well I'm alone there now...
      In our 'special place'...
      Waiting for you...

James: I got a letter.
      The name on the envelope said 'Maria.'
      My wife's name...
      It's ridiculous, couldn't possibly be true...
      That's what I keep telling myself...
      A dead person can't write a letter.
      Maria died of that damn disease three years ago.
      So then why am I looking for her'
      Our 'special place'...
      What could she mean'
      This whole town was our special place.
      Does she mean the park on the lake'
      We spent the whole day there.
      Just the two of us, staring at the water.
      Could Maria really be there'
dario@dario-vm:~/Escritorio$
```

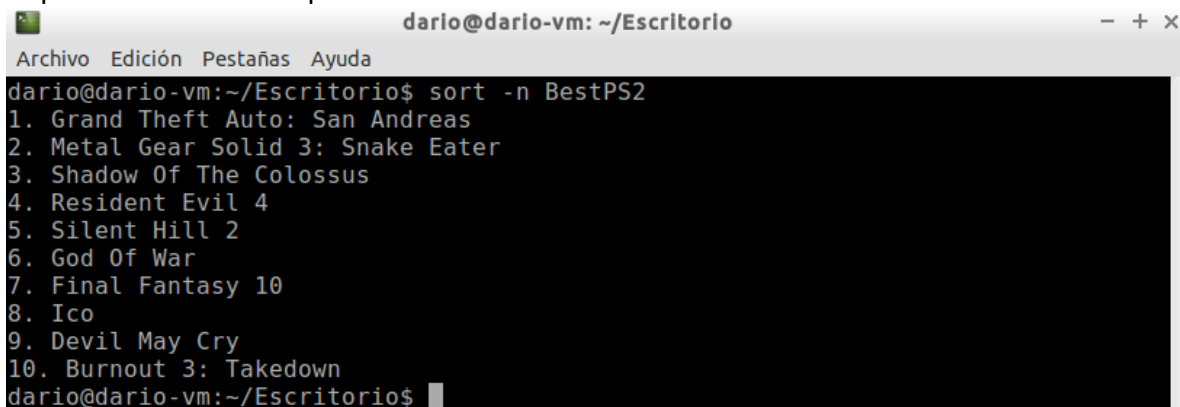
## SORT:

Ordena las líneas de un archivo.



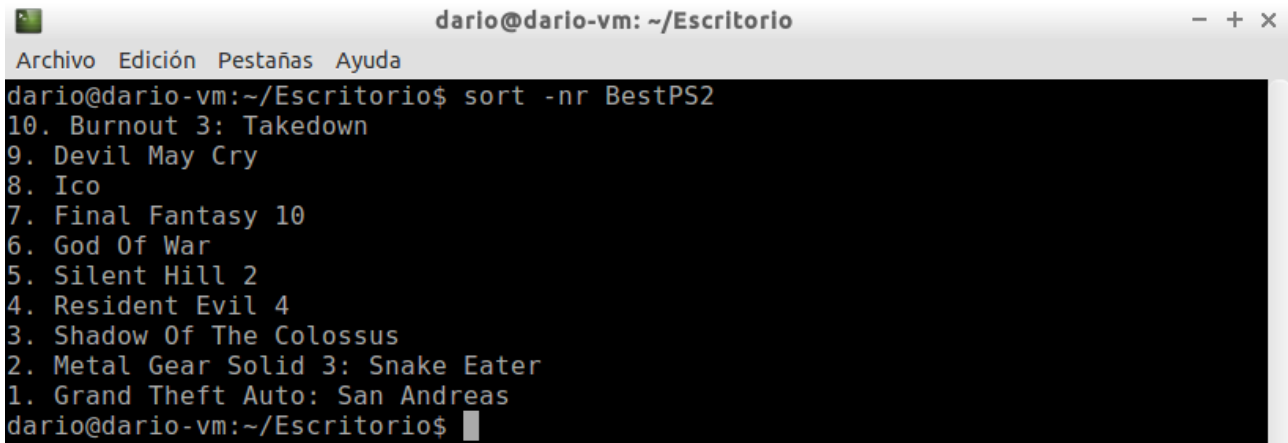
```
BestPS2
Archivo Editar Buscar Opciones Ayuda
1 4. Resident Evil 4
2 5. Silent Hill 2
3 7. Final Fantasy 10
4 1. Grand Theft Auto: San Andreas
5 8. Ico
6 9. Devil May Cry
7 2. Metal Gear Solid 3: Snake Eater
8 10. Burnout 3: Takedown
9 6. God Of War
10 3. Shadow Of The Colossus
11
```

Si quisiéramos ordenar por número:



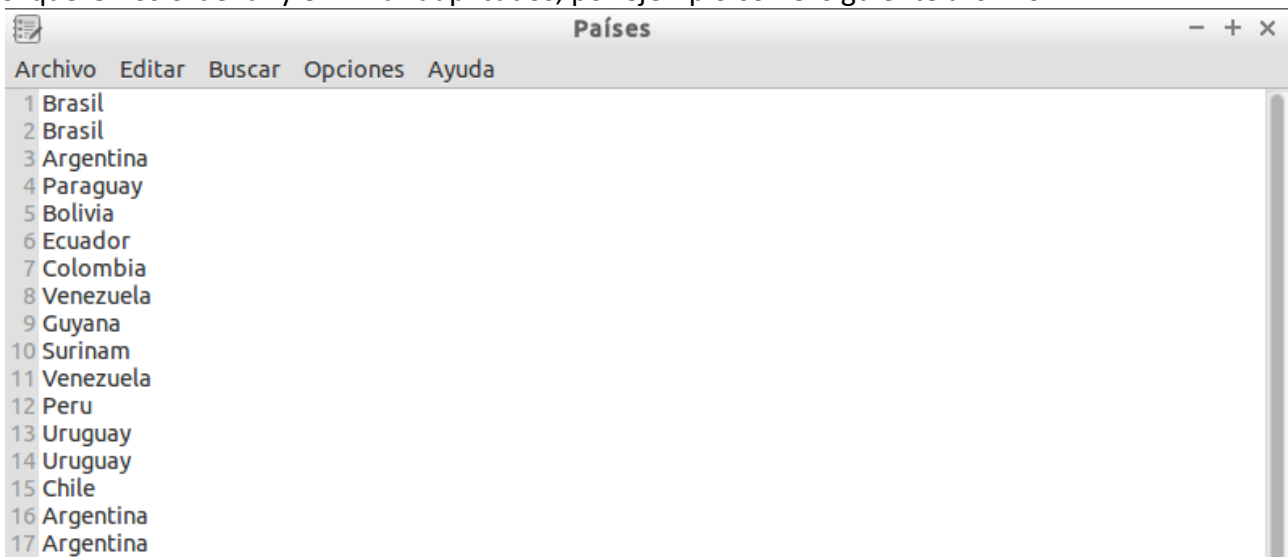
```
dario@dario-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
dario@dario-vm:~/Escritorio$ sort -n BestPS2
1. Grand Theft Auto: San Andreas
2. Metal Gear Solid 3: Snake Eater
3. Shadow Of The Colossus
4. Resident Evil 4
5. Silent Hill 2
6. God Of War
7. Final Fantasy 10
8. Ico
9. Devil May Cry
10. Burnout 3: Takedown
dario@dario-vm:~/Escritorio$
```

Si quisiéramos ordenar por número en orden descendente:



```
dario@dario-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
dario@dario-vm:~/Escritorio$ sort -nr BestPS2
10. Burnout 3: Takedown
9. Devil May Cry
8. Ico
7. Final Fantasy 10
6. God Of War
5. Silent Hill 2
4. Resident Evil 4
3. Shadow Of The Colossus
2. Metal Gear Solid 3: Snake Eater
1. Grand Theft Auto: San Andreas
dario@dario-vm:~/Escritorio$
```

Si queremos ordenar y eliminar duplicados, por ejemplo con el siguiente archivo:



```
Países
Archivo Editar Buscar Opciones Ayuda
1 Brasil
2 Brasil
3 Argentina
4 Paraguay
5 Bolivia
6 Ecuador
7 Colombia
8 Venezuela
9 Guyana
10 Surinam
11 Venezuela
12 Peru
13 Uruguay
14 Uruguay
15 Chile
16 Argentina
17 Argentina
```

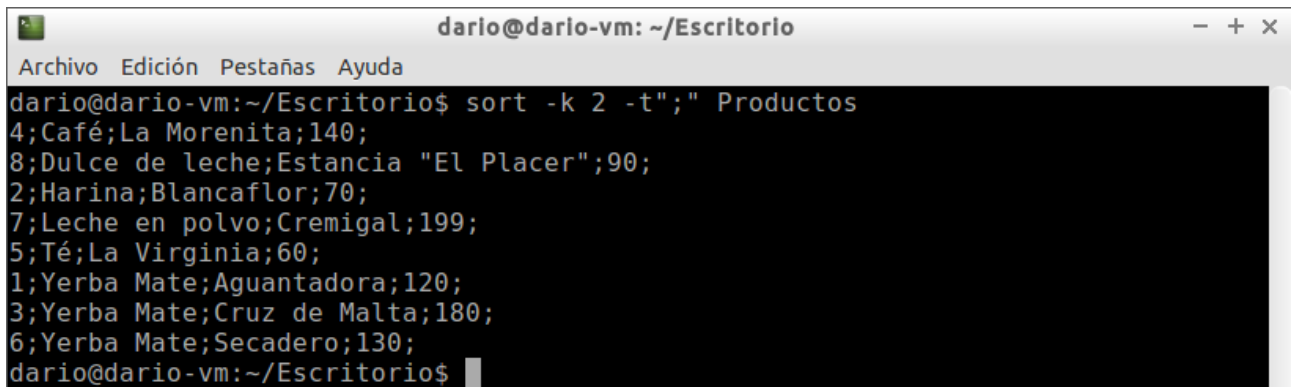


```
dario@dario-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
dario@dario-vm:~/Escritorio$ sort -u Países
Argentina
Bolivia
Brasil
Chile
Colombia
Ecuador
Guyana
Paraguay
Peru
Surinam
Uruguay
Venezuela
dario@dario-vm:~/Escritorio$
```

Si queremos ordenar el siguiente archivo .csv por nombre de producto:



```
1 1;Yerba Mate;Aguantadora;120;
2 2;Harina;Blancaflor;70;
3 3;Yerba Mate;Cruz de Malta;180;
4 4;Café;La Morenita;140;
5 5;Té;La Virginia;60;
6 6;Yerba Mate;Secadero;130;
7 7;Leche en polvo;Cremigal;199;
8 8;Dulce de leche;Estancia "El Placer";90;
```



```
darío@darío-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
darío@darío-vm:~/Escritorio$ sort -k 2 -t";" Productos
4;Café;La Morenita;140;
8;Dulce de leche;Estancia "El Placer";90;
2;Harina;Blancaflor;70;
7;Leche en polvo;Cremigal;199;
5;Té;La Virginia;60;
1;Yerba Mate;Aguantadora;120;
3;Yerba Mate;Cruz de Malta;180;
6;Yerba Mate;Secadero;130;
darío@darío-vm:~/Escritorio$
```

Si queremos ordenar por precio de mayor a menor:



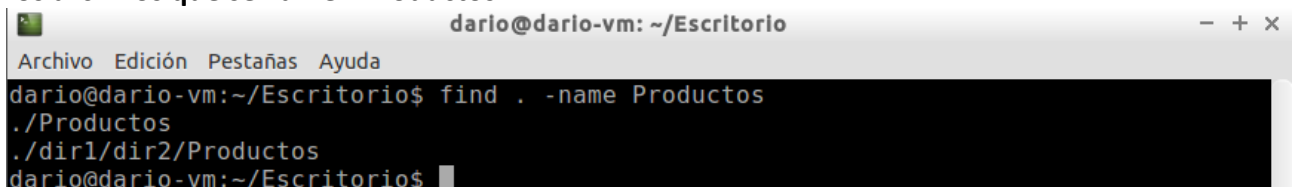
```
darío@darío-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
darío@darío-vm:~/Escritorio$ sort -k 4 -t";" -nr Productos
7;Leche en polvo;Cremigal;199;
3;Yerba Mate;Cruz de Malta;180;
4;Café;La Morenita;140;
6;Yerba Mate;Secadero;130;
1;Yerba Mate;Aguantadora;120;
8;Dulce de leche;Estancia "El Placer";90;
2;Harina;Blancaflor;70;
5;Té;La Virginia;60;
darío@darío-vm:~/Escritorio$
```

## FIND:

Busca archivos en una jerarquía de directorios.

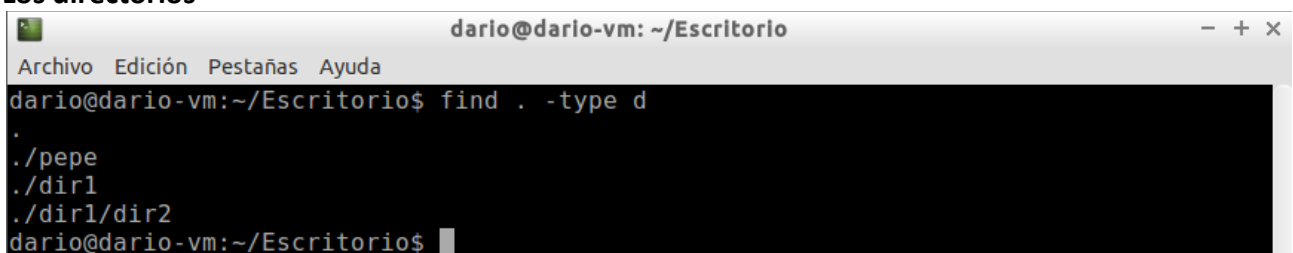
Si quisiéramos buscar desde donde estamos parados:

### Los archivos que se llamen Productos



```
darío@darío-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
darío@darío-vm:~/Escritorio$ find . -name Productos
./Productos
./dir1/dir2/Productos
darío@darío-vm:~/Escritorio$
```

### Los directorios



```
darío@darío-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
darío@darío-vm:~/Escritorio$ find . -type d
.
./pepe
./dir1
./dir1/dir2
darío@darío-vm:~/Escritorio$
```



## Los archivos/directorios vacíos

```
dario@dario-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
dario@dario-vm:~/Escritorio$ find . -empty
./pepe/archivo2
dario@dario-vm:~/Escritorio$
```

## Borrar los archivos/directorios vacíos

```
dario@dario-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
dario@dario-vm:~/Escritorio$ find . -empty -delete
dario@dario-vm:~/Escritorio$
```

## Listar los archivos/directorios con un tamaño mayor a 512 bytes

```
dario@dario-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
dario@dario-vm:~/Escritorio$ find . -size +512c
.
./dir1
./dir1/dir2
./texto2
./texto
dario@dario-vm:~/Escritorio$
```

## Listar los archivos/directorios con que fueron modificados hace mas de un día.

## Listar los archivos/directorios con que fueron modificados hace mas de un minuto.

```
dario@dario-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
dario@dario-vm:~/Escritorio$ find . -mtime +1
dario@dario-vm:~/Escritorio$ find . -mmin +1
.
./Productos
./dir1
./dir1/dir2
./dir1/dir2/Productos
./cuentas
./texto2
./Países
./res
./BestPS2
./texto
dario@dario-vm:~/Escritorio$
```

## PS:

Lista los procesos que se están ejecutando actualmente en el sistema (snapshot).

```
dario@dario-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
dario@dario-vm:~/Escritorio$ ps -aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.6  4456  3396 ?        Ss   06:53   0:21 /sbin/init
root         2  0.0  0.0      0     0 ?        S    06:53   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    06:53   0:00 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S<   06:53   0:00 [kworker/0:0H]
root         7  0.0  0.0      0     0 ?        S    06:53   0:11 [rcu_sched]
root         8  0.0  0.0      0     0 ?        S    06:53   0:00 [rcu_bh]
root         9  0.0  0.0      0     0 ?        S    06:53   0:00 [migration/0]
root        10  0.0  0.0      0     0 ?        S    06:53   0:00 [watchdog/0]
root        11  0.0  0.0      0     0 ?        S    06:53   0:00 [kdevtmpfs]
```

## TOP:

Lista los procesos que se están ejecutando actualmente en el sistema (se actualiza). Una especie de administrador de tareas de Windows.

```
dario@dario-vm: ~/Escritorio
top - 15:03:36 up 8:09, 3 users, load average: 0,12, 0,15, 0,17
Tareas: 146 total, 2 ejecutar, 144 hibernar, 0 detener, 0 zombie
%Cpu(s): 3,0 usuario, 3,3 sist, 0,0 adecuado, 93,8 inact, 0,0 en espera, 0,
KiB Mem: 505412 total, 486892 used, 18520 free, 136744 buffers
KiB Swap: 448508 total, 7424 used, 441084 free. 147432 cached Mem

  PID  USUARIO  PR  NI  VIRT  RES  SHR  S  %CPU  %MEM  HORA+  ORDEN
 1194  root      20   0 119620 38852 12968 R   3,3   7,7   3:04.19  Xorg
 2129  dario     20   0 114740 23628 18376 S   1,0   4,7   0:34.47  lxterminal
23242  root      20   0 862676 16712 10476 S   0,7   3,3   0:00.22  snapd
22469  root      20   0      0      0      0 S   0,3   0,0   0:00.04  kworker/0:0
23239  dario     20   0   6852  2872  2468 R   0,3   0,6   0:00.13  top
```

## KILL:

Envía una señal a un proceso.

Lista de señales:

```
dario@dario-vm: ~/Escritorio
dario@dario-vm:~/Escritorio$ kill -l
1) SIGHUP      2) SIGINT      3) SIGQUIT     4) SIGILL      5) SIGTRAP
6) SIGABRT     7) SIGBUS      8) SIGFPE      9) SIGKILL     10) SIGUSR1
11) SIGSEGV    12) SIGUSR2    13) SIGPIPE    14) SIGALRM    15) SIGTERM
16) SIGSTKFLT  17) SIGCHLD    18) SIGCONT    19) SIGSTOP    20) SIGTSTP
21) SIGTTIN    22) SIGTTOU    23) SIGURG     24) SIGXCPU    25) SIGXFSZ
26) SIGVTALRM  27) SIGPROF    28) SIGWINCH   29) SIGIO       30) SIGPWR
31) SIGSYS     34) SIGRTMIN   35) SIGRTMIN+1 36) SIGRTMIN+2 37) SIGRTMIN+3
38) SIGRTMIN+4 39) SIGRTMIN+5 40) SIGRTMIN+6 41) SIGRTMIN+7 42) SIGRTMIN+8
43) SIGRTMIN+9 44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9 56) SIGRTMAX-8 57) SIGRTMAX-7
58) SIGRTMAX-6 59) SIGRTMAX-5 60) SIGRTMAX-4 61) SIGRTMAX-3 62) SIGRTMAX-2
63) SIGRTMAX-1 64) SIGRTMAX
```

Si ejecutamos el comando PS

```
dario@dario-vm: ~/Escritorio
ntp      1565  0.0  0.7  5732  3788 ?        Ss   06:54   0:04 /usr/sbin/ntpd -p /var/run/ntpd.pid -g -u 10
root     1583  0.1  1.0  42832  5240 ?        Sl   06:54   0:40 /usr/bin/vmtoolsd
root     1631  0.0  0.3   4660  1880 tty1     Ss+  06:54   0:00 /sbin/getty -8 38400 tty1
dario    2129  0.1  4.7 114804 23852 ?        Sl   06:57   0:37 lxterminal
dario    2130  0.0  0.3   2428  1812 ?        S    06:57   0:00 gnome-pty-helper
dario    2131  0.0  0.9   8288  4996 pts/0    Ss   06:57   0:01 /bin/bash
root     4739  0.0  0.9  21768  4632 ?        Sl   07:11   0:00 /usr/sbin/console-kit-daemon --no-daemon
dario    8601  0.0  4.3 128228 22120 ?        Sl   13:40   0:03 leafpad /home/dario/Escritorio/res
dario    11557 0.0  4.4 128512 22736 ?        Sl   13:57   0:04 leafpad /home/dario/Escritorio/res
dario    20208 0.0  0.9   8212  4868 pts/4    Ss   11:42   0:00 /bin/bash
root     21914 0.0  0.0      0      0 ?        S    14:56   0:00 [kworker/u2:2]
root     22469 0.0  0.0      0      0 ?        S    14:59   0:00 [kworker/0:0]
root     23133 0.0  0.0      0      0 ?        S    15:02   0:00 [kworker/u2:1]
dario    23239 0.3  0.5   6852  2872 pts/4    S+   15:03   0:01 top
root     23278 0.0  0.0      0      0 ?        S    15:04   0:00 [kworker/0:1]
dario    24013 0.0  4.4 120200 22484 ?        Sl   08:59   0:10 leafpad /home/dario/Escritorio/matriz
root     24331 0.0  0.0      0      0 ?        S    15:09   0:00 [kworker/0:2]
root     24730 0.0  0.0      0      0 ?        S    15:11   0:00 [kworker/u2:0]
root     25006 7.3  3.2 862676 16676 ?        Ssl  15:12   0:00 /usr/lib/snapd/snapd
root     25016 0.0  0.4  13148  2488 ?        S    15:12   0:00 /lib/systemd/systemd-udevd --daemon
dario    25020 0.0  0.4   6516  2488 pts/0    R+   15:12   0:00 ps -aux
```

La segunda columna es el PID (Process ID) del proceso, vamos a enviar una señal al proceso leafpad (que tiene abierto el archivo de texto matriz), para ello necesitamos conocer su PID, en este caso es 24013.

```
dario@dario-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
dario@dario-vm:~/Escritorio$ kill 24013
```

Vemos que el proceso ya no está, esto es porque al ejecutar kill sin ningún tipo de señal, por defecto se envía una señal de finalización al proceso.

```
dario@dario-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
dario 1551 0.0 0.9 10524 4756 ? S 06:54 0:00 /usr/lib/i386-linux-gnu/gconf/gconfd-2
dario 1556 0.0 1.2 63024 6420 ? Sl 06:54 0:10 /usr/lib/gvfs/gvfsd-trash --spawner :1.7 /or
ntp 1565 0.0 0.7 5732 3788 ? Ss 06:54 0:04 /usr/sbin/ntpd -p /var/run/ntpd.pid -g -u 10
root 1583 0.1 1.0 42832 5240 ? Sl 06:54 0:40 /usr/bin/vmtoolsd
root 1631 0.0 0.3 4660 1880 tty1 Ss+ 06:54 0:00 /sbin/getty -8 38400 tty1
dario 2129 0.1 4.7 114868 23864 ? Sl 06:57 0:38 lxterminal
dario 2130 0.0 0.3 2428 1812 ? S 06:57 0:00 gnome-pty-helper
dario 2131 0.0 0.9 8288 4996 pts/0 Ss 06:57 0:01 /bin/bash
root 4739 0.0 0.9 21768 4632 ? Sl 07:11 0:00 /usr/sbin/console-kit-daemon --no-daemon
dario 8601 0.0 4.3 128228 22120 ? Sl 13:40 0:03 leafpad /home/dario/Escritorio/res
dario 11557 0.0 4.4 128512 22736 ? Sl 13:57 0:04 leafpad /home/dario/Escritorio/res
dario 20208 0.0 0.9 8212 4868 pts/4 Ss 11:42 0:00 /bin/bash
root 23133 0.0 0.0 0 0 ? S 15:02 0:00 [kworker/u2:1]
dario 23239 0.3 0.5 6852 2872 pts/4 S+ 15:03 0:02 top
root 23378 0.0 0.0 0 0 ? S 15:04 0:00 [kworker/0:1]
root 24331 0.0 0.0 0 0 ? S 15:09 0:00 [kworker/0:2]
root 24730 0.0 0.0 0 0 ? S 15:11 0:00 [kworker/u2:0]
root 25323 0.0 0.0 0 0 ? S 15:14 0:00 [kworker/0:0]
root 25760 10.5 3.3 862676 16996 ? Ssl 15:16 0:00 /usr/lib/snapd/snapd
root 25770 0.0 0.4 13148 2488 ? S 15:16 0:00 /lib/systemd/systemd-udevd --daemon
dario 25774 0.0 0.4 6516 2488 pts/0 R+ 15:16 0:00 ps -aux
dario@dario-vm:~/Escritorio$
```

Si quisiéramos enviar una señal específica, podemos hacerlo así:

```
dario@dario-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
dario@dario-vm:~/Escritorio$ kill -2 24013
```

o así

```
dario@dario-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
dario@dario-vm:~/Escritorio$ kill -s SIGINT 24013
```

En ambos casos se envía la misma señal, que es la que se envía cuando hacemos CTRL-C desde el teclado para finalizar un proceso que haya tomado la terminal.

## TRAP:

Permite capturar una señal, por parte de un proceso. La aplicaremos en la parte de scripting a nuestros scripts para que puedan capturar señales y llevar a cabo alguna tarea específica o solo ignorarlas.

## Variables:

**De shell:** Su alcance se limita a la sesión de shell actual.

**De entorno:** Es una variable de shell exportada, con lo cual su alcance se extenderá a cualquier subshell o proceso que se cree desde la sesión que creo la variable.

Por ejemplo, creemos una variable de shell:

```
dario@dario-vm: ~
Archivo Edición Pestañas Ayuda
dario@dario-vm:~$ mi_variable="Hola Mundo"
dario@dario-vm:~$ echo $mi_variable
Hola Mundo
dario@dario-vm:~$
```

Con el comando echo podemos imprimir un mensaje en pantalla, en este caso mostramos el contenido de la variable.

Pero ahora si creamos una nueva sesión desde la sesión actual y mostramos la variable:

```
dario@dario-vm: ~
Archivo Edición Pestañas Ayuda
dario@dario-vm:~$ bash
dario@dario-vm:~$ echo $mi_variable

dario@dario-vm:~$
```

Si creamos una variable de entorno, es básicamente una variable que exportamos con el comando export

```
dario@dario-vm: ~
Archivo Edición Pestañas Ayuda
dario@dario-vm:~$ export VAR="Variable de entorno"
dario@dario-vm:~$ bash
dario@dario-vm:~$ echo $VAR
Variable de entorno
dario@dario-vm:~$ exit
exit
dario@dario-vm:~$ echo $VAR
Variable de entorno
dario@dario-vm:~$
```

Vemos que la nueva sesión puede visualizar la variable.

Si intentamos modificar su valor desde una nueva sesión, vemos que los cambios solo surten efecto en la sesión “hija”, no se puede modificar el entorno hacia arriba en la jerarquía.

```
dario@dario-vm: ~
Archivo Edición Pestañas Ayuda
dario@dario-vm:~$ bash
dario@dario-vm:~$ VAR=pepe
dario@dario-vm:~$ echo $VAR
pepe
dario@dario-vm:~$ exit
exit
dario@dario-vm:~$ echo $VAR
Variable de entorno
dario@dario-vm:~$
```

Podemos visualizar las variables de shell y de entorno con el comando set

```
dario@dario-vm: ~
Archivo Edición Pestañas Ayuda
dario@dario-vm:~$ set | grep VAR
VAR='Variable de entorno'
dario@dario-vm:~$
```

Una de las variables mas comunes es PATH:

```
dario@dario-vm: ~
Archivo Edición Pestañas Ayuda
dario@dario-vm:~$ set | grep PATH
DEFAULTS_PATH=/usr/share/gconf/Lubuntu.default.path
MANDATORY_PATH=/usr/share/gconf/Lubuntu.mandatory.path
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
```

Contiene una lista de directorios (paths) separados por : al ejecutar un comando por ejemplo, si dicho comando no se encuentra en el directorio actual, se lo busca en los directorios que figuran en la variable PATH.

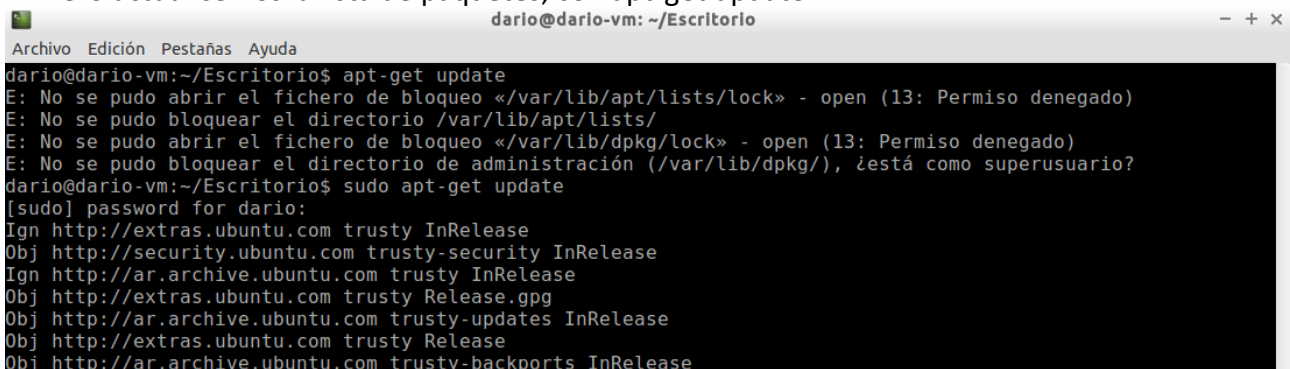
### Superusuario (root):

La cuenta de superusuario es la cuenta con mayor permiso en el sistema. Tiene permisos de lectura, escritura y ejecución de cualquier aplicación del sistema.

Por defecto el usuario que creamos durante la instalación del sistema será un usuario administrador con privilegios de superusuario (root). Puede haber varios usuarios administradores pero solo hay una cuenta de superusuario.

Utilicemos como ejemplo el comando **APT-GET** que es utilizado para administrar e instalar paquetes a nuestra distribución de Linux de Ubuntu. Es muy probable que al instalar Ubuntu por ejemplo, debamos instalar por separado otras herramientas como por ejemplo el compilador gcc, para esto debemos utilizar el comando **apt-get**.

Primero actualicemos la lista de paquetes, con apt-get update



```
dario@dario-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
dario@dario-vm:~/Escritorio$ apt-get update
E: No se pudo abrir el fichero de bloqueo «/var/lib/apt/lists/lock» - open (13: Permiso denegado)
E: No se pudo bloquear el directorio /var/lib/apt/lists/
E: No se pudo abrir el fichero de bloqueo «/var/lib/dpkg/lock» - open (13: Permiso denegado)
E: No se pudo bloquear el directorio de administración (/var/lib/dpkg/), ¿está como superusuario?
dario@dario-vm:~/Escritorio$ sudo apt-get update
[sudo] password for dario:
Ign http://extras.ubuntu.com trusty InRelease
Obj http://security.ubuntu.com trusty-security InRelease
Ign http://ar.archive.ubuntu.com trusty InRelease
Obj http://extras.ubuntu.com trusty Release.gpg
Obj http://ar.archive.ubuntu.com trusty-updates InRelease
Obj http://extras.ubuntu.com trusty Release
Obj http://ar.archive.ubuntu.com trusty-backports InRelease
```

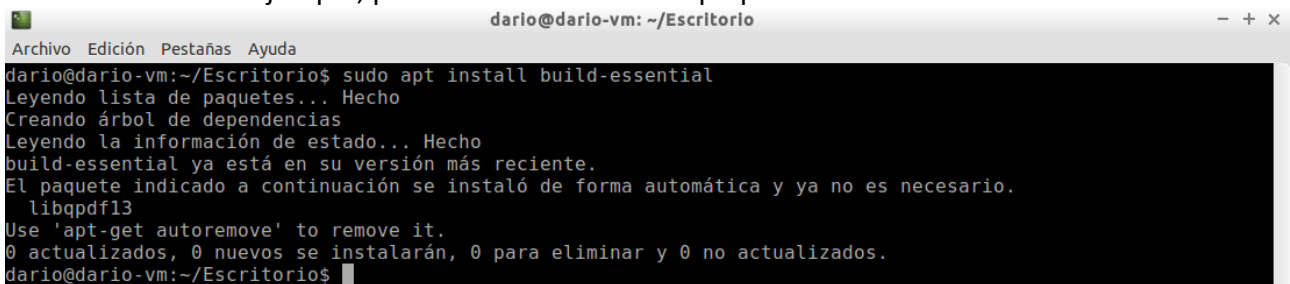
En la primera ejecución nótese que arrojó errores de permiso, incluso pregunta ¿está como superusuario?, porque apt-get es una herramienta que debe ser ejecutada con privilegios de superusuario (root).

Entonces procedemos a ejecutar el comando, anteponiendo el comando **SUDO**, véase que solicitará nuestra contraseña, para ver si efectivamente podemos ejecutar como superusuario.

### SUDO:

Permite ejecutar un comando como otro usuario, si no se especifica usuario asume que se quiere ejecutar como superusuario.

Terminando con el ejemplo, procedemos a instalar el paquete build-essential



```
dario@dario-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
dario@dario-vm:~/Escritorio$ sudo apt install build-essential
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
build-essential ya está en su versión más reciente.
El paquete indicado a continuación se instalará de forma automática y ya no es necesario.
libqpdf13
Use 'apt-get autoremove' to remove it.
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
dario@dario-vm:~/Escritorio$
```

Y las paginas del manual



```
darío@darío-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
darío@darío-vm:~/Escritorio$ sudo apt-get install manpages-dev
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
manpages-dev ya está en su versión más reciente.
fijado manpages-dev como instalado manualmente.
El paquete indicado a continuación se instaló de forma automática y ya no es necesario.
 libqpdf13
Use 'apt-get autoremove' to remove it.
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
darío@darío-vm:~/Escritorio$
```

En mi caso ya tenía la última versión actualizada. Para validar la versión del compilador de GCC:

```
darío@darío-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
darío@darío-vm:~/Escritorio$ gcc --version
gcc (Ubuntu 4.8.4-2ubuntu1~14.04.4) 4.8.4
Copyright (C) 2013 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

En el caso de querer utilizar la sesión como si fuéramos el root, podemos hacerlo con la ejecución de **sudo su** ó **sudo -i**:

```
root@darío-vm: /home/darío/Escritorio
Archivo Edición Pestañas Ayuda
darío@darío-vm:~/Escritorio$ sudo su
[sudo] password for darío:
root@darío-vm:/home/darío/Escritorio#
```

Nótese que el usuario en el prompt cambió a root, ahora si quisiéramos ejecutar el **apt-get update**, ya no es necesario anteponer el comando **sudo**:

```
root@darío-vm: /home/darío/Escritorio
Archivo Edición Pestañas Ayuda
root@darío-vm:/home/darío/Escritorio# apt-get install build-essentials
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
```

Con el comando **exit**, podemos volver a utilizar la sesión con nuestro usuario:

```
darío@darío-vm: ~/Escritorio
Archivo Edición Pestañas Ayuda
root@darío-vm:/home/darío/Escritorio# exit
exit
darío@darío-vm:~/Escritorio$
```

De todas formas al cerrar la terminal, finalizará la sesión y al abrir otra terminal estaremos de nuevo con nuestro usuario, por más que no hayamos deslogueado al root.

Para llevar a cabo este mecanismo de política de seguridad, el sistema se apoya en un archivo, **/etc/sudoers**, si queremos visualizar su contenido:

```
darío@darío-vm: ~
Archivo Edición Pestañas Ayuda
darío@darío-vm:~$ cat /etc/sudoers
cat: /etc/sudoers: Permiso denegado
```

hay que ejecutarlo el comando como superusuario:

```
darío@darío-vm: ~
Archivo Edición Pestañas Ayuda
cat: /etc/sudoers: Permiso denegado
darío@darío-vm:~$ sudo cat /etc/sudoers
#
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults      env_reset
Defaults      mail_badpass
Defaults      secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin"

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL

# Members of the admin group may gain root privileges
%admin   ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "#include" directives:

#includedir /etc/sudoers.d
```

### **root ALL=(ALL: ALL) ALL**

El usuario root tiene privilegios ilimitados.

### **%admin ALL=(ALL) ALL**

Cualquier miembro del grupo de administradores tiene los mismos privilegios que el root.

### **%sudo ALL = (ALL: ALL) ALL**

Todos los miembros del grupo sudo tienen los privilegios para ejecutar cualquier comando.

Analizando la primera línea:

### **root ALL=(ALL: ALL) ALL**

**usuario host=(como que usuario: como que grupo) para que comandos**

usuario a quien se aplica la regla, si se antepone % representa un nombre grupo

host a que se aplica la regla

como que usuario puede ejecutar

como que grupo puede ejecutar

que comandos puede ejecutar

Para mayor información referirse a la ayuda con el comando **man 5 sudoers**

Si quisiéramos ejecutar con privilegios de root en un usuario que no es administrador, ni tampoco está configurado en sudoers.

```
pepe@darío-vm: ~
Archivo Edición Pestañas Ayuda
pepe@darío-vm:~$ ls
Descargas  Escritorio  Música     Público
Documentos Imágenes    Plantillas Vídeos
pepe@darío-vm:~$ sudo -i
[sudo] password for pepe:
pepe no está en el archivo sudoers. Se informará de este incidente.
pepe@darío-vm:~$
```