Índice

¿Linux? 2
Historial 2
GNU/Linux 2
Distribuciones de Linux 2
Kernel 3
Shell 3
Bash Shell 4
Sistemas de Archivos. Filesystems 4
Comandos 5
Sintaxis básica de comandos 6
Opciones 6
Comando exec y comando source 8
Información de un comando 8
Obtención de ayuda 10
Secciones del manual 10
Ficheros de inicialización 10
Anexo-1 Solución de los ejercicios. 13
Ejercicio 1 35
Solución 35
Ejercicio- 2 35
Solución 35
Ejercicio- 3 35
Solución 35
Ejercicio- 4 36
Solución 36
Ejercicio- 5 36
Solución 36

Ejercicio- 6 36
Solución 36

¿Linux?

La forma correcta de nombrarlo sería GNU-Linux

El nombre de Linux proviene del nombre del creador de una sección del SO llamada Kernel "**Linu**s Torvalds" y del sistema operativo antecesor llamado Mini**x** que no es más que una versión reducida de UNIX.

Historial

UNIX fue creado por AT&T Bell Labs en la década de los 1970

El desarrollo del Lenguaje C fue entre 1969 y 1973 y se convirtió en estándar ANSI en el 1990

En el 1991 se lanza la primera versión de Linux

Linux, es decir el Kernel de GNU-Linux está básicamente programado en C, este lenguaje también fue usado para programar UNIX

Las primeras versiones se emitieron como código cerrado, pero en 1991 gracias a Richard Matthew Stallman se convierte en código abierto bajo licencia GPL.

GNU/Linux

El kernel por sí solo no es utilizable. Para que un SO funcione necesita mucho más, Shell, compiladores, bibliotecas, etc. . Todas estas otras partes eran las implementadas en GNU

Sistema Operativo

- Nucleo (Kernel)
- Shell
- Sistema de archivos
- Aplicaciones

Distribuciones de Linux

Tome el Kernel de Linux y las herramientas de GNU, agrega aplicaciones adicionales orientadas a usuarios, como: cliente de correo electrónico, procesadores de texto, etc. y tendrás un sistema GNU-Linux completo (una distribución).

Las distribuciones se encargan de configurar el almacenamiento, instalar el Kernel e instalar el resto del software y un administrador de paquetes.

Las principales distribuciones Linux son las basadas en "Red Hat" y las basadas en "Debian". La diferencia más visible entre ellas entre otras cosas es el gestor de paquetes.

Red Hat dispone de un gestor de paquete denominado Red Hat Package Manager (RPM) basado en archivos ".rpm". Es una distribución dirigida a aplicaciones de servidor. Esta

distribución incluye un asesoramiento de pago. Y de ella derivan las distribuciones CentOS y Fedora.

CentOS es una distribución clon de Red Hat pero sin soporte y por tanto gratis.

Fedora es una distribución de la familia de Red Hat de propósito general orientada a equipos de uso personal.

Aunque existen más distribuciones, estas son las que LPIC reconoce como las dominantes del mundo Linux. No obstante, existe una distribución más antigua que a un están en eso es la basada en la familia Slackware, con variantes tan famosas como SUSE o OpenSUSE

Kernel

Es el software encargado de gestionar el hardware e interactuar con las aplicaciones, dando acceso a la memoria, gestionando el uso de las CPU, cerrando programas cuando estos estén huérfanos, etc.

Shell

En ocasiones llamado interprete de comando que permite al usuario comunicarse con el sistema operativo. Hay dos tipos de comunes de Shell son la interface gráfica (GUI) y la interface de línea de comandos (CLI)

Ventajas del CLI

- Repetición de comandos: La GUI no tiene una forma fácil para repetir un comando o para modificar un comando anterior o anteriores
- Flexibilidad de comandos: el Shell GUI tiene una forma limitada de utilización de los comandos
- Recursos: La GUI suele utilizar más recursos como (RAM, CPU, etc.)
- Secuencias de comandos: La GUI no tiene una forma fácil de enlazar comandos secuenciales.

- Acceso remoto: Aun que se pueden ejecutar comandos de forma remota en la GUI esta función no está configurada de forma predeterminadas.
- Desarrollo: El tiempo de implementar un programa basado en GUI es mayor al tiempo necesario para realizarlo en CLI.

Los sistemas Linux tiene una manera fácil de cambiar entre CLI y GUI (también llamado entorno de escritorio). Dos de estas GUI más comunes son GNOME y KDE.

De la misma forma que hay varios GUI también existe la posibilidad de disponer de varios shell CLI como bash, sh, zsh, …

Bash Shell

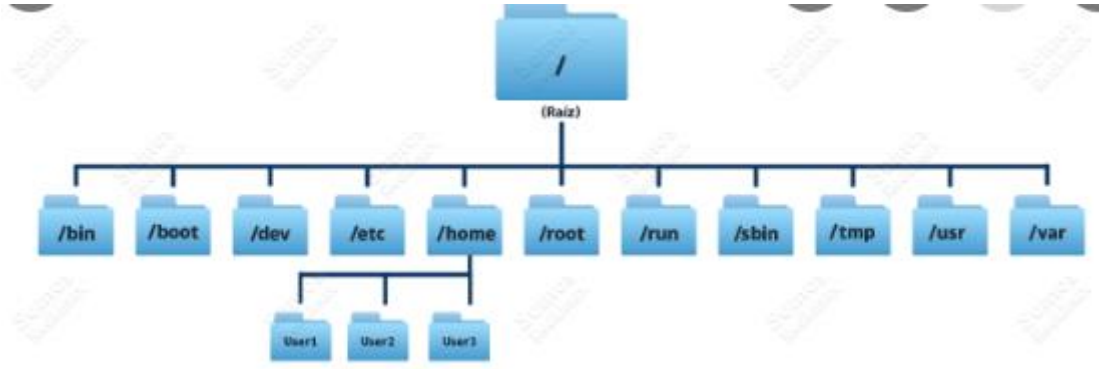
```
command [options] [arguments]
```

El shell Bash o también conocido como GNU Bash es el Shell más extendido y utilizado de todos. Dispone de numerosos comandos y funciones incorporados con innumerables opciones, entre las funcionalidades podemos indicar:

- **Alias**: asigne a un comando un nombre diferente o más corto para que el trabajo con el shell sea más eficiente.
- **Volver a ejecutar comandos**: para evitar tener que volver a escribir líneas de comando largas.
- **Coincidencia de comodines**: utiliza caracteres especiales como: ?, * Y [] para seleccionar uno o más archivos como un grupo para su procesamiento.
- **Input/Output Redirection**: Usa caracteres especiales para redirigir la entrada, < o <<, y saluda, >.
- **Tuberías**: se utilizan para conectar uno o más comandos simples para realizar operaciones más complejas.
- **Procesamiento en segundo plano**: permite que los programas y comandos se ejecuten en segundo plano mientras el usuario continúa interactuando con el shell para completar otras tareas.

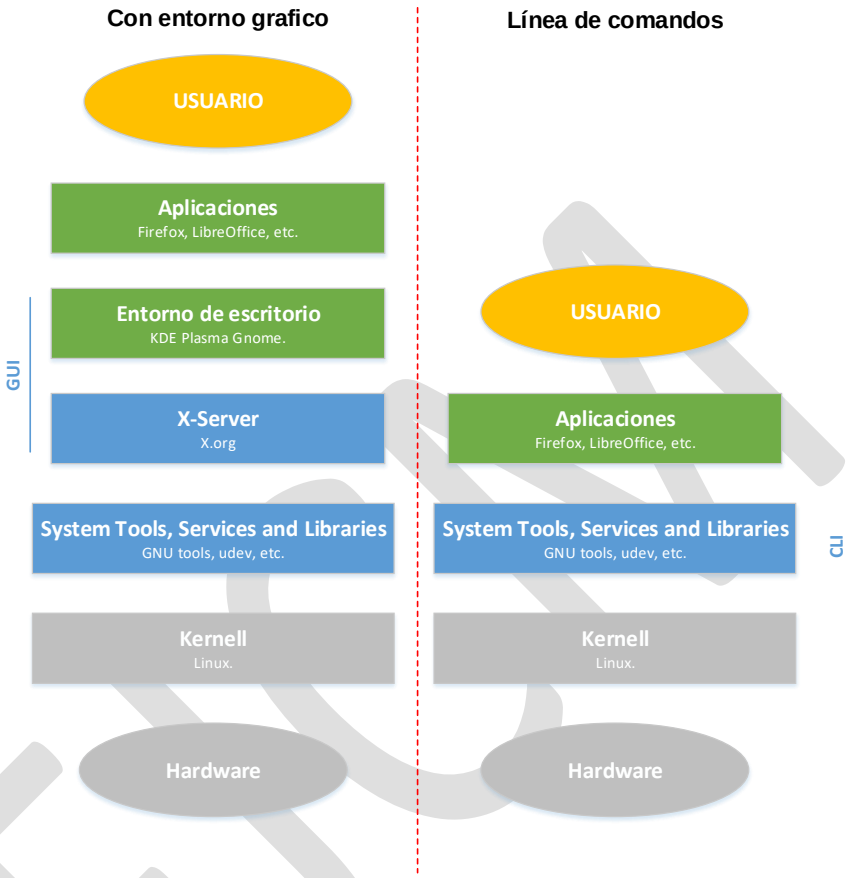
Sistemas de Archivos. Filesystems

A parte del Kernel y del Shell el otro componente clave del sistema operativo es el sistema de archivos.



Desde el punto de vista de usuario es una jerarquía de directorios

Desde el punto de vista del sistema operativo es la estructura en una tabla de partición de disco y definir la ubicación de directorios y archivos.



Ejercicio 1

Viendo el diagrama de bloques anterior. Cuando un user entre en entorno gráfico y abre la aplicación terminal ¿está abriendo un CLI?

Comandos

Hay varias formas de clasificar los comandos y una de ellas es en función de donde proceden.

- **Comandos internos:** también llamados comandos integrados y estos comandos están integrados dentro del propio Shell. Cuando un usuario escribe estos comandos eje el Shell bash ya conoce dicho comando, al ser un comando propio y ya sabe cómo interpretar ese comando y no necesita programas adicionales. Ej el comando *cd*

- **Comandos externos:** estos comandos se almacenan en archivos que son buscados por el Shell en una lista predeterminada de directorios. Estos comandos también se pueden ejecutar escribiendo la ruta completa del comando. Ej el comando *ls*

Almacenados en los directorios predeterminado para ellos la mayoría están escritos en lenguaje C

- **Alias:** es una asociación de un nombre a un comando que se ejecutara con preferencia al comando y se utiliza para simplificar opciones de los comandos. Ej. alias ll = 'ls –l' Este alias genera un nuevo comando llamado ll en el cual usa un comando ya existente junto con la opción –l (de listado).

- **Funciones**: se podría ver como la creación de nuevos comandos que utilizan comandos existentes. O como la creación de un script que realiza una función deseada y enlazada con el PATH del usuario, simulando así un nuevo comando.

Sintaxis básica de comandos

La sintaxis básica de un comando es:

```
command [options…] [arguments…]
```

Muchos comandos se pueden usar sin opciones y sin argumentos y otros necesitan algunas de ellas o las dos. También hay argumentos que se pueden usar con o sin.

Si algún argumento contiene caracteres no alfanuméricos se debe citar los argumentos, con comillas simple o dobles si deseas literalidad o evaluar respectivamente.

Es decir las comillas dobles interpretara los metacaracteres como: !!, \ , $,`

Las comillas simples, lo tratara todo como un literal.

Puede darse el caso que en algunas ocasiones tengamos argumentos que contengan caracteres inusuales en estos casos la mejor opción es citar (añadir comillas) dichos argumentos

Para tener presente el tema de los metacaracteres propongo la siguiente práctica.

Practica

Si lanzamos el comando *ls –l*, ¿que saldrá por pantalla? Probarlo

Si a continuación lanzásemos el comando: *echo holaaaa mundo !!*

¿Qué pensáis que salda por la consola? Probarlo

Por qué creéis que ha salido, lo que ha salido

Hacer la misma secuencia cambiando el comando *echo holaaaa mundo !!* por *echo "holaaaa mundo !!"*

Volver a repetir la secuencia ahora cambiando el segundo comando por *echo 'holaaaa mundo !!'*

Opciones

La mayoría de comandos disponen de un conjunto de opciones que expande y/o modifican la forma de operar el comando.

Hablaremos de esto más adelante, cuando conozcamos más comandos, pero como idea tomar que nos podemos encontrar que con que *<comando> -a* no dé el mismo resultado que *<comando> a* O que *<comando> -la* tenga un comportamiento totalmente diferente a *<comando> -al*

Existen tres formatos en las que podemos dar estas opciones a los comandos (BSD, UNIX, GNU). Algunos comandos solo permitirán trabaja en uno de estos formatos y otros comandos tendrán la capacidad de trabajar con varios y/o todos los formatos. Ahora enumeramos como distinguirlos.

- BSD: no usa guiones y cada opción es un solo carácter

  Ej ps a

- UNIX: usa un solo guion y cada opción es un solo carácter

  Ej. ls -a

- GNU: usa dos guiones seguidos y las opciones son palabras en lugar de caracteres.

  Ej. ls --all

Por antigüedad el orden cronológico de más antiguo a más moderno de estos formatos es BSD → UNIX → GNU

Los comandos que dispongan de la posibilidad de usar varios de estos formatos puede suceder que en versiones futuras desaparezcan los formatos más antiguos.

Hay que tener mucho cuidado con el tema de las opciones y también con sus formatos porque a un que los formatos más modernos tienen en cuenta la funcionalidad de los más antiguos puede suceder que la opción no dé el mismo resultado. También depende de la distribución de Linux y del orden en el que se pongan dichas opciones.

En distribuciones derivadas de Debian, **normalmente** el orden de las opciones no suele afectar, en cambio en derivados de Red Hat el orden de las opciones en algunos casos son muy importantes y pueden cambiar totalmente la funcionalidad del comando.

Las opciones en formatos BSD y UNIX se pueden apilar y o unificar

En este caso *ls –a –l* realizara la misma función que *ls –l –a* o que *ls –la* o *ls –al*

Las opciones en GNU no se puede apilar, tiene que ser individuales, por separado.

ls --all --directory

Propuesta de ejercicios.

Para la generación de estos ejercicios hay muchas formas de realizarlos y todas ellas son correctas, se valorará más eficientes y simples.

Ejercicio- 2

Conociendo que el comando “ls” lista el contenido
Verificar en vuestra consola que los comando realizados anteriormente se cumplen.
Usar también las opciones –t, -r, -d para que sucede.

Ejercicio- 3

Listar del directorio actual todos los ficheros “visibles” ordenados por fecha de creación en orden ascendente

Ejercicio-4

Listar el contenido de los archivos y carpetas “ocultos” de tu carpeta actual. (solo el nivel actual)

Ejercicio-5

Lista solo las carpetas no ocultas contenidas en tu carpeta actual.

Ejercicio-6

Lista solo las carpetas ocultas contenidas en tu carpeta actual.

Comando exec y comando source y el comando “.”

Una excepción a la sintaxis de comandos básica es la utilización del comando *exec* y el comando *source* que sus argumentos son comandos

El comando *source* normalmente se suele utilizar para cargar configuraciones en el shell actual. El comando *source* es sinónimo del comando **.** (comando punto) y es capaz de ejecutar un comando en el shell actual

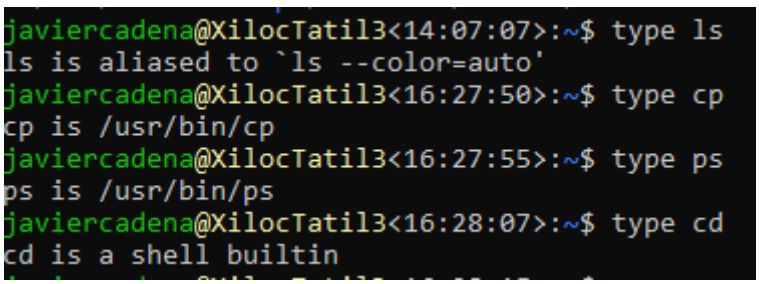
El comando *exec* también se usa para ejecutar script y/o programas pero a diferencia de *source* el comando *exec* cierra el shell actual tras finalizar la ejecución del comando que lanza, se usa habitualmente en el arranque del SO para saltar de fases.

Otra peculiaridad es que estos dos comandos son capaces de lanzar scripts directamente

Información de un comando

Existen comandos que nos permiten identificar donde está un comando o qué tipo de comando es.

El comando *type* identifica la ubicación de otros archivos y/o su origen



En este ejemplo identifica el comando *ls* como un alias del comando *ls --color=auto* los comandos *cp* y *ps* como comandos GNU ubicados en la carpeta */usr/bin/*
El comando *cd* según nos indica el ejemplo es un compilado del shell.

Si añadimos la opción –a nos dará todas las ubicaciones

> Un comando puede estar ubicado en varias localizaciones distintas y tener funcionalidades distintas. En estos casos la ruta que el comando *which* mostrará será la primera ubicación en la que el comando sea localizado. La forma de buscar será siguiendo el orden de carpetas contenidas en la variable de sistema PATH de izquierda a derecha.
>
> El comando *whereis* a diferencia del comando *which* por defecto nos dará todas las ubicaciones del comando y no solo del binario sino también del manual
>
> ```
> javiercadena@XilocTatil3<17:40:51>:~$ whereis grep
> grep: /usr/bin/grep /usr/share/man/man1/grep.1.gz /usr/share/info/grep.info.gz
> ```

También existe el comando *which* que nos permite localizar un comando.

Información del sistema

Para obtener información del sistema existe el comando *uname* que nos data información sobre el SO que se está usando.

```
javiercadena@test1:~ Thu Nov 03 <16:56:32>
$ uname
Linux
```

Sin entrar en detalles tenemos la opción –*a* que nos dará toda la información.

```
javiercadena@test1:~ Thu Nov 03 <17:38:44>
$ uname -a
Linux test1 5.4.0-128-generic #144-Ubuntu SMP Tue Sep 20 11:00:04 UTC 2022 x86_64 x86_64 x86_64 GNU/Linux
```

En este caso nos dice que el Kernel es Linux
Que el sistema operativo es GNU/Linux
Que la maquina se llama test1
Que la versión del Kernel es la 5.4.0-128

Si quisiéramos datos de la distribución que se usa es mejor usar el comando *lsb_release* con la opción –*a*

```
javiercadena@test1:~ Thu Nov 03 <17:46:58>
$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 20.04.5 LTS
Release:        20.04
Codename:       focal
```

En este caso nos está diciendo que la distribución es Ubuntu versión 20.04 que es esta versión se llamada focal, también podemos apreciar que es una versión LTS
Esto significa que es una versión de largo soporte (5 años) el resto de versiones suelen tener n soporte de 9 meses.
En el caso de Canonica, en particular de Ubuntu los números de las versiones corresponde al año y mes de lanzamiento de la versión. Así que la versión 20.04 es la versión que salió el mes

de abril del 2020. Las distribuciones de Ubuntu salen una cada 6 meses, así que tendremos todos los xx.04 y xx.10 correspondiente a abril y ha octubre e cada año xx

Autocompletar comandos

Bash dispone de funcionalidades que facilita el uso de los terminales.

La forma de utilizar el autocompletado se basa en comenzar a escribir el comando y posteriormente presionar la tecla tab para que en el caso que solo exista una combinación posible el sistema autocompletará el comando, en el caso que exista más de una posibilidad tendremos que presionar dos veces a la tecla tab y nos mostrará todas las combinaciones posibles.

Obtención de ayuda

Existe varias formas de obtención de ayuda, muchos de los comandos contienen la opción de *--help* que genera una pequeña descripción de las peculiaridades del comando.
No, obstante la forma principal de obtener ayuda es mediante el comando *man* (manual), con el comando *man* no solo obtendremos información de otros comandos, también puede mostrar información de algunos ficheros del sistema.

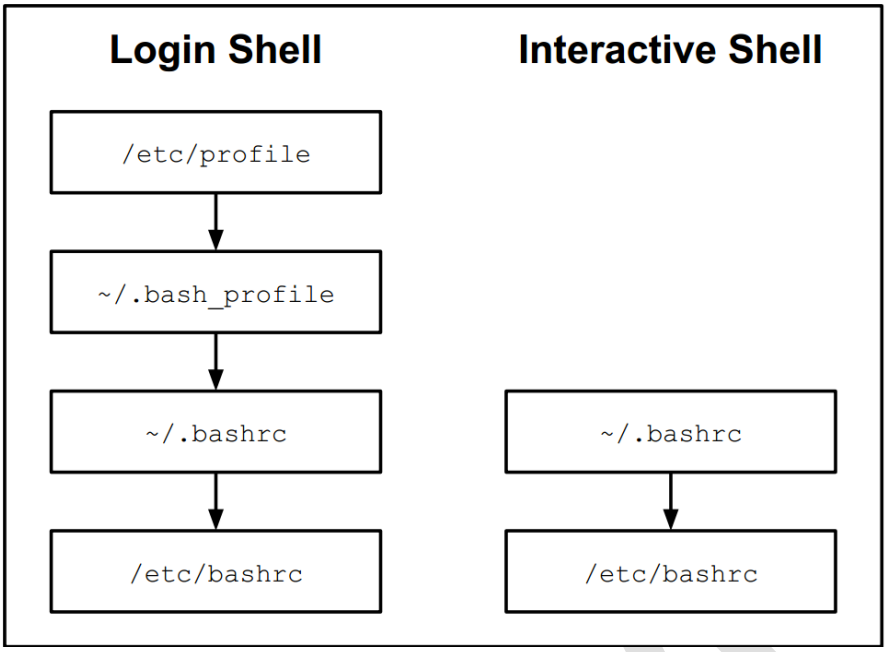
Secciones del manual

1- Programas ejecutables o comandos de Shell
2- Llamadas al sistema (funciones proporcionadas por el kernel)
3- Llamadas a la biblioteca (funciones dentro de las bibliotecas de programas)
4- Archivos especiales (generalmente se encuentran en */dev*)
5- Convenciones y formatos de archivo, p. Ej. */etc/passwd*
6- Juegos
7- Varios (incluidos paquetes de macros y convenciones), p. Ej. hombre (7), groff (7)
8- Comandos de administración del sistema (generalmente solo para root)
9- Rutinas de kernel [no estándar]

Ficheros de inicialización

Los ficheros de inicialización o también conocidos como ficheros de configuración son los encargados de cargar las configuraciones y las personalizaciones de la Shell en cuestión.

Los ficheros de inicialización de bash son:



Así como estos ficheros son los lanzados al cargar el bash también existe la posibilidad de lanzar otros scripts al salir del bash. Esto normalmente se suele usar para tácticas de limpieza y los archivos usados normalmente suelen ser: *~/.bash_logout* y */etc/bash_logout*

Historial de comandos

Bash guarda un historial de últimos comandos no repetidos al inmediatamente anterior y esto nos permite acceder a ellos con posterioridad de forma fácil.

Hay varias formas de acceder a ellos.

- Puede usar las teclas de flecha arriba ↑ y abajo ↓ para revisar su historial y seleccionar un comando anterior para ejecutarlo nuevamente.
- Puede seleccionar un comando anterior y modificarlo antes de ejecutarlo.
- Puede hacer una búsqueda inversa a través del historial para encontrar un comando anterior para seleccionarlo, editarlo y ejecutarlo. Para iniciar la búsqueda, presione Ctrl + R y luego comience a escribir una parte de un comando anterior.
- Ejecuta un comando nuevamente, basado en un número que está asociado con el comando.

Acción	Tecla	Alternativa
Ítem anterior del historial	↑	Ctrl+P
Próximo ítem del historial	↓	Ctrl+N
Búsqueda de historial inversa		Ctrl +R
Principio de línea	Home	Ctrl + A
Fin de línea	End	Ctrl + E
Borra un carácter	Del	Ctrl + D
Borrar carácter a la izquierda	Backspace	Ctrl + X
Mover cursos a la izquierda	←	Ctrl + B
Mover cursos a la derecha	→	Ctrl+ F

Para ver el listado del historial use el comando *history*

```
sysadmin@localhost:~$ history
        1  ls
        2  cd test
        3  cat alpha.txt
        4  ls -l
        5  cd ..
        6  ls
        7  history
```

Ejercicio – 7

Verifica que sucede si usas dos veces seguidas el mismo comando.

¿Se queda registrado en el historial?

Ejercicio – 8

Ejecuta el último comando de tres formas diferentes.

Ejercicio -9

Ejecuta el comando que se ejecutó hace 3 comandos

Globbing (Comodines)

Carácter asterisco * → ninguno o más caracteres

Carácter interrogación *?* → un carácter

Carácter corchete *[ ]* → un solo carácter de la lista

Nota: GNU-Linux permite añadir comodines a los nombres de los archivos y por tanto a los nombres de las carpetas, pero se recomienda efusivamente no usarlos.

Una cosa curiosa es que podemos usar el comando *echo ** como un *ls*

Cuando un carácter comodín se coloca dentro de los corchetes el carácter pierde su significado comodín y se convierte en carácter literal.

Si el primer carácter dentro del corchete empieza por la ! o por la ^ entonces ese primer carácter tiene un significado especial, en este caso el patrón [!a-f]* Aquí el sistema "buscara" todas las palabras cuyo nombre NO empiece con ninguno de los carácter de la *'a'*, a la *'f'*.

Ejercicio – 10

Cuál será el resultado de los siguientes comandos

*echo **

ls [*]?

*ls [*]**

*ls w[1-3]**

ls w?

*ls w[!a,e,i,o,u]**

*ls w[!a,e,i,o,u]**

Manipulación de ficheros

En GNU-Linux todo se considera archivos, por lo que esta sección es muy importante.

No solo los documentos se consideran archivos si no que las carpetas son archivos, los dispositivos hardware como discos duros, hardware de comunicación.

Ya conocemos unos cuantos comandos como:

echo
uname
lsb_release
Pero a continuación nombraremos unos cuantos más.

Unos primeros comandos

pwd

Este comando se usa para mostrar mi ubicación dentro del árbol de carpetas.

ls

Es unos de los comandos más usados y con más opciones. Se usa para lista contenido de una carpeta. Y podemos identificar mucha información de los archivos en función de las diferentes opciones que utilicemos. Entre ellas podemos ver.

- Quien es el propietario del fichero
- A que grupo pertenece.
- Cuáles son los permisos del archivo como escritura, lectura, ejecución
- La cantidad de enlaces duros asociado a este archivo
- Qué tipo de archivo es: si es un archivo de bloque, archivo de carácter, una carpeta o un fichero “estándar”
- Podemos ver el tamaño que ocupa el fichero en el disco
- Podemos ver el tamaño real de la información del fichero
- Podemos ver a que inodo pertenece
- Etc, etc..

Hablaremos más adelante sobre los permisos de los archivos

Existe muchísimas opciones de este comando no los veremos todos, pero a continuación mostramos los más relevantes:

- l lista con detalles
- a lista todos los archivos incluidos los “ocultos”
- t lista ordenados por marca de tiempo
- r reverse, invierte el orden del listado
- i lista los inodos
- d lista las carpetas
- s lista el tamaño que ocupa en disco
- h l la compactación de k, M, G

NOTA: *Hago notar la diferencia entre tamaño de un fichero y espacio que ocupa el fichero en el disco. El tamaño mínimo de un fichero puede ser de 1byte, no obstante, el espacio mínimo que ocupara en el disco es el espacio del bloque, del inodo, es decir el realizar el formateo se definirá el tamaño de los inodos y esto definirá la porción mínima de espacio por cada bloqueo, por cada inodo. Que en la mayoría de los casos será 4K*

Otros conceptos y convenios de GNU/Linux

Explicaremos unos conceptos que ayudaran al dia a dia del uso de la consola. No pertenecen a ninguna sección en concreto pero …

La ubicación ./

Utilizaremos el punto barra cuando hagamos referencia a elementos en nuestra ubicación actual.

Por ejemplo cuando queramos ejecutar un fichero A que este en nuestro directorio actual sea cual sea usaremos ./A

La ~

Este símbolo alt+126 o al AlGr+4 hare referencia a la carpeta del usuario también conocida como la home del usuario…
Este símbolo puede aparece tanto en el Promt para indicarnos donde estamos como en los comando vistos anteriormente.

```
sysadmin@localhost:~$
```

Estamos en nuestra carpeta home.

```
sysadmin@localhost:~$ cp Documents/n* ~
```

En este caso está copiando dentro de la carpeta Documents todos los elementos que empiezan con la letra n y los está copiando en ~ es decir los está copiando en la carpeta del usuario.

La #

Cuando en el promt aparece # en lugar de ~ quiere decir que estamos en la home del user root o del super usuario.

En este caso no se recomienda usarlo para los comandos.

El punto .

Hace referencia a la carpeta actual.

Cuando queremos copiar o mover archivos a la carpeta en la que nos encontramos actualmente se usa el punto "." Por ejemplo si queremos copiar el fichero de configuración de ssh a nuestra carpeta actual haremos:

cp /etc/ssh/sshd.conf .

cp */etc/ssh/sshd.conf* . → copia

cp /etc/ssh/sshd.conf . → el fichero sshd.conf

cp /etc/ssh/sshd.conf . → en mi ubicación

Punto punto

Hace referencia a la carpeta anterior en el árbol de directorios.

cd

Este comando es usando para cambiar de ubicación, cambiar de carpeta.

Existen varios atajos a tener en cuenta.

Practica

Ejecuta el siguiente comando.

ls –l

```
$ ls -la
total 84
drwxr-xr-x 10 javiercadena javiercadena 4096 nov 15 11:28 .
drwxr-xr-x  4 root         root         4096 nov  2 15:51 ..
-rw-------  1 javiercadena javiercadena 4346 nov 15 06:31 .bash_history
-rw-r--r--  1 javiercadena javiercadena  220 feb 25  2020 .bash_logout
```

¿A que hace referencia las dos primeras líneas?

Entonces, ¿cómo iremos a la posición anterior?

Visto esta práctica tenemos dos formas por desplazamos por las carpetas del SO en rutas absolutas o en rutas relativas.

Las rutas absolutas se definen como las ruta que parten la carpeta raíz o también conocida como carpeta root, que se identifica con el carácter /

Las rutas relativas se desplazas desde nuestra carpeta actual

cp

Este comando lo podemos usar para copiar archivo/s su formato es:

cp [opciones] <origen> <destino>

Unas de las opciones más comunes son *–v* : verbose y *–R* de recursivo

mv

Este comando lo podemos usar para mover ficheros de una ubicación a otra.

Este comando también se suele usar para renombrar un archivo para ello hay que poner la misma ubicación destino que origen. Su formato de uso es:

mv <origen> <destino>

mkdir

Se usa para crear carpetas. Su formato es:

mkdir <nombre de la carpeta>

rm

Se usa para borrar archivos y depende de la opción también puede borrar carpetas.

Practica

Usando el fichero existente llamado practica que está en tu home copiarlo a tu carpeta de trabajo
¿Qué instrucción utilizaras?
Una vez copiado verifica que el fichero existe en las dos ubicaciones.
Crea una carpeta dentro de tu carpeta de trabajo con el nombre p1
¿Qué instrucción utilizaras?
Lista con detalle para ver los permisos de la carpeta
Mueve el fichero a la nueva carpeta creada
Renombre al fichero a un nombre a tu elección
Borra el fichero

touch

Este es un comando muy potente y su objetivo principal es cambiar la marca de tiempo de un fichero. Un fichero dispone de varias marcas de tiempo entre ellas:

- Fecha de la última modificación del contenido del archivo (es la marca de tiempo que se muestra al listar con *ls –l*)
- Fecha de la última vez que se accedió al archivo

- La última vez que se modificaron los atributos del archivo

Y se usan las opciones *–t*, *-a*, *–c* respectivamente

En el comando anterior, la sintaxis de fecha / hora para el comando *touch* es CCYYMMDDHHMM (opcionalmente agregue SS para los segundos)

Si quisiéramos ver las diferentes marcas de tiempo usaremos el comando *stat*

El comando touch también se suele usar para crear ficheros vacíos, si usamos el comando touch sin opciones y ponemos el nombre de un fichero que no exista creará un fichero sin contenido.

Si el fichero si existe cambiara todas las marcas de tiempo con el tiempo actual.

Si quieres ver más ver el anexo 2

Practica

Crear un fichero que se llame --file, si lo haces sin más el comando touch lo interpretara como una opción. Pruébalo

Prueba forzando una opción en formato GNU y dejándola en blanco

touch -- --file

Ahora veremos porque no es bueno usar determinados caracteres especiales en los nombres de los ficheros
Muy bien, ahora intenta bórralo con el comando rm, con lo aprendido hasta ahora deberías de poder lograrlo.

pwd

El comando pwd se usa para mostrar el directorio actual, en la mayoría de Shell actuales esta información se suele añadir por defecto en el promt pero en los Shell más básico no y para saber dónde se esta era necesario consultarlo…

find

Es un comando para buscar archivos.

Ejemplo	explicación
-iname LOSTFILE	Búsqueda que no distingue entre mayúsculas y minúsculas
-mtime -3	Ficheros modificados los últimos 3 días
-mmin -10	Ficheros modificados los últimos 10 minutos
-size +1M	Fichero de más de 1MByte
-user joe	Fichero que el user propietario sea joe
-nouser	Archivos no propiedad de ningún user
-empty	Archivos vacíos
-type d	Archivos con el atributo de directorio
-maxdepth 1	Recursividad máxima de 1 nivel (actual)

find <ubicacion> <opciones>

Para generar una salida que muestre detalles adicionales del archivo, similar al comando *ls -l*, puede especificar la opción *–ls*

```
sysadmin@localhost:~$ find . -ls -iname 'desk*' -o \( -name Downloads -a -user sysadmin \)
209950600      4 drwxr-xr-x   1 sysadmin sysadmin     4096 Mar 10 04:08 .
210019331      4 drwxr-xr-x   2 sysadmin sysadmin     4096 Mar  8 19:10 ./Templates
102108986      4 -rw-r--r--   1 sysadmin sysadmin      220 Apr  4  2018 ./.bash_logout
102108988      4 -rw-r--r--   1 sysadmin sysadmin      807 Apr  4  2018 ./.profile
210019332      4 drwxr-xr-x   2 sysadmin sysadmin     4096 Mar  8 19:10 ./Downloads
 ...
```

find /etc –size -1k

busca en la carpeta /etc ficheros de menos de 1k

find / -size +1M

busca en la carpeta raíz ficheros de 1M o más

whereis

Este comando se usa para localizar la ubicación de otros comandos y para buscar el file de ayuda

Su formato es:

whereis [opción] nombre

```
sysadmin@localhost:~$ whereis ls
ls: /bin/ls /usr/share/man/man1/ls.1.gz
```

which

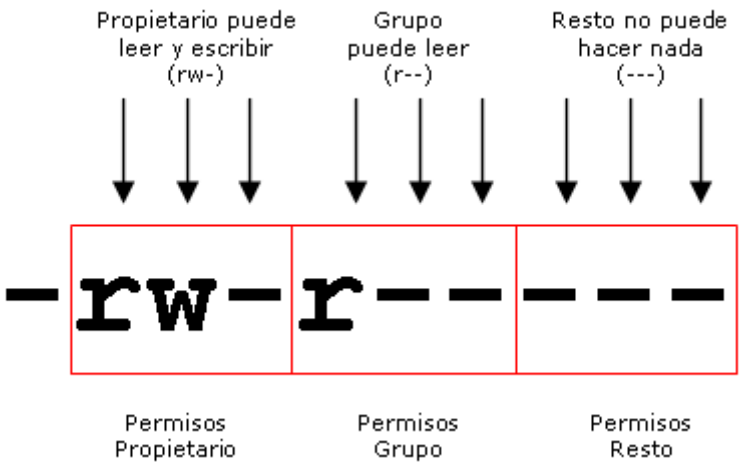
Este comando al igual que el anterior se usa para localizar comandos

which [-a] nombre

```
sysadmin@localhost:~$ which bash
/bin/bash
```

Permisos básicos de archivos

```
javiercadena@test1:~ Mon Oct 31 <16:20:55>
$ ll
total 76
drwx------ 6 javiercadena javiercadena  4096 Aug  5 16:48 ./
drwxr-xr-x 4 root         root          4096 May 26 07:12 ../
-rw------- 1 javiercadena javiercadena 13223 Oct 28 18:00 .bash_history
-rw-r--r-- 1 javiercadena javiercadena   220 Feb 25  2020 .bash_logout
```



De izquierda a derecha, el primer carácter, que no está enmarcado en rojo corresponde a los permisos especiales. Dentro de estos permisos especiales entre otras cosas podemos identificar si el archivo es:

Archivo normal: se identifican porque con tiene un ralla -

Archivo carpeta: se identifica por que se pone una d

Archivo carácter: se identifica por que se pone una c

Archivo bloque: se identifica por que se pone unb a

Archivo enlace: se identifica con la letra l

```
crw-------  1 root root
lrwxrwxrwx  1 root root
drwxr-xr-x  3 root root
crw-r--r--  1 root root
lrwxrwxrwx  1 root root
crw-rw----  1 root disk
brw-rw----  1 root disk
```

A continuación, en bloques de 3 posiciones, tendremos los permisos de propietarios, grupos y otros.

Donde:

r = permiso de lectura
w = permiso de escritura
x = permiso de ejecución.
El orden de estos permisos es muy importante como veremos más adelante

Los permisos definen como es un archivo.
Para hacer que un archivo sea ejecutable basta con darle permisos de ejecución, quien lo pueda ejecutar definirá en que bloque de los 3 hay que agregar dicho permiso (propietario, grupo u otros), es decir si deseamos que el grupo puedo escribir daremos el permiso de escritura al grupo. Para asignar estos permisos tenemos varias formas todas usando el comando chmod

chmod

chmod es un comando que nos permite modificar los permisos. Y sos opciones se dividen en tres zonas

La primera tenemos que indicar a que colectivo se aplican los cambios.

Símbolo	Grupo
u	Usuarios
g	Grupos
o	Otros
a	Todos

A continuación que modificación se realizara.

Símbolo	Operación
+	Añade el permiso, si es necesario
=	Especifica los permisos exactos
-	Borra un permiso, si es necesario

Y por último que a que permiso o permisos ya que se puede apilar

Símbolo	Permisos
r	read
w	write
x	ejecutar

A continuación, hay que añadir el nombre del fichero al que hay que realizar dicha modificación

Por ej.

chmod g+w <nombre_file>

Añadirá los permisos de escritura *w* al grupo *g* en el fichero "*nombre_file*"

Dentro de este bloque vemos que tenemos los operadores +, -, = tal como indica su cuadro

Cuando usamos el comando + estamos añadiendo la propiedad de permiso que escojamos, tenga ya dicho permiso o no. En el ejemplo añadimos el permiso de escritura al bloque de permisos de grupo, en el caso que ya lo tenga, lo sobre escribe y si no lo tiene lo añade. Similarmente pasaría con el –, le quitaría el permiso si lo tiene o lo dejaría sin permiso en el caso que no lo tenga. Pero si usamos el = la cosa cambia. En este caso estamos forzando a que todo el bloque sea con dicho permiso. Es decir, *chmod g=r <nombre_file>* estaríamos forzando a que el permiso de grupo *g* solo sea de lectura, sería equivalente a decir*: chmod g-wx, g+r <nombre_file>* como ves se puede agrupar las opciones para ejecutarlo todo en un solo comando.

Existe otro método para asignar los permisos es el método octal.

Octal Value	Permission
4	read
2	write
1	execute
0	none

Este método se asigna de forma un valor numérico a cada permiso según su posición, de aquí surge la importancia de recordar la posición de los permisos “rwx”, los pesos de estos permisos están definidos en la tabla anterior, en el caso de desear modificar más de un permiso a la vez, tendremos que sumar los pesos de dichos permisos. Esto será válido para los tres bloques de permisos propietario, grupo y otro.

Octal Value	Permission Set
7	`rwx`
6	`rw-`
5	`r-x`
4	`r--`
3	`-wx`
2	`-w-`
1	`--x`
0	`---`

Nota: Si un fichero tuviese los permisos 000 (ningún permiso), el propietario del fichero y el root podrían usar *chmod* para modificarlos.

Si un directorio tiene los permisos de *rw* se podrá borrar dicha carpeta con el comando *rm* tenga permisos en los ficheros internos o no.

Escenarios de permisos

Permisos	Efectos en el fichero	Efectos en el directorio
Read r	Permite que los procesos lean el contenido del archivo, lo que significa que el contenido se puede ver y copiar.	Se pueden enumerar los nombres de archivo en el directorio, pero no hay otros detalles disponibles.
Write w	Permite que el proceso escriba en el contenido, por lo que se pueden guardar los cambios en un archivo. Tenga en cuenta que el permiso w realmente requiere el permiso r en el archivo para funcionar correctamente.	Los archivos se pueden agregar o eliminar del directorio. Tenga en cuenta que el permiso w requiere un permiso x en el directorio para funcionar correctamente.
Execute x	Permite que un archivo se ejecute o se ejecute como un proceso.	Permite al usuario usar el comando cd para "ingresar" al directorio y usar el directorio en un nombre de ruta para acceder a los archivos y,

		potencialmente, a los subdirectorios de este directorio.

Ejercicio – 11

Dado el siguiente escenario ¿qué puede hacer el usuario bob sobre el archivo /home/bob/test?

```
drwxr-xr-x 1 root root  4096 Apr 23 21:08 /
drwxr-xr-x 1 root root  4096 Apr 17 22:31 /home
drwxr-xr-x 1 bob  staff 4096 Apr 23 21:09 /home/bob
-r--r--r-- 1 bob  staff 390  Apr 17 22:31 /home/bob/test
```

Ejercicio – 12

Dado el siguiente escenario ¿Puedo el usuario bob eliminar el archivo /home/bob/test?

```
drwxr-xr-x 1 root root  4096 Apr 23 21:08 /
drwxr-xr-x 1 root root  4096 Apr 17 22:31 /home
drwxr-xr-x 1 bob  staff 4096 Apr 23 21:09 /home/bob
---------- 1 bob  staff 390  Apr 17 22:31 /home/bob/test
```

Ejercicio – 13

Dada la siguiente información, puede el usuario *sue* puede ver el contenido del fichero /home/bob/test?

```
drwxr-xr-x 1 root root  4096 Apr 23 21:08 /
drwxr-xr-x 1 root root  4096 Apr 17 22:31 /home
drwxr--r-- 1 bob  staff 4096 Apr 23 21:09 /home/bob
-r--r--r-- 1 bob  staff 390  Apr 17 22:31 /home/bob/test
```

chown y chgrp

Se utilizan para modificar el usuario y el grupo respectivamente, al que pertenece un fichero *chown pep file1* modificara el usuario propietario del fichero file1 a pep siempre y cuando el fichero exista, el usuario exista y quien lo cambie tenga permisos para hacer la modificación.

Utilidades de texto

Contenido de un archivo

cat

Es uno de los comandos más usados para leer el contenido de un o varios archivos.

```
cat [OPTION]... [FILE]...
```

```
sysadmin@localhost:~$ cd Documents
sysadmin@localhost:~/Documents$
sysadmin@localhost:~/Documents$ cat alpha-first.txt
A is for Animal
B is for Bear
C is for Cat
D is for Dog
E is for Elephant
F is for Flower
sysadmin@localhost:~/Documents$ cat alpha-second.txt
G is for Grapes
H is for Happy
I is for Ink
J is for Juice
K is for Kangaroo
L is for Lol
M is for Monkey
```

```
sysadmin@localhost:~/Documents$ cat alpha-first.txt alpha-second.txt
A is for Animal
B is for Bear
C is for Cat
D is for Dog
E is for Elephant
F is for Flower
G is for Grapes
H is for Happy
I is for Ink
J is for Juice
K is for Kangaroo
L is for Lol
M is for Monkey
```

less o more

Si el archivo es grade y se desea recórrelo se recomienda usar los comandos less o more

Son dos comandos similares que paginan los documentos para poder visualizarlos poco a poco

nl

Numera las líneas de un fichero, por defecto solo enumera las líneas que no están en blanco para que las numere todas hay que usas la opción

```
sysadmin@localhost:~/Documents$ nl -b a newhome.txt
```

O como

```
sysadmin@localhost:~/Documents$ nl -ba newhome.txt
     1  Thanks for purchasing your new home!!
     2
     3  **Warning** it may be haunted.
     4
     5  There are three bathrooms.
     6
     7  **Beware** of the ghost in the bedroom.
     8
     9  The kitchen is open for entertaining.
    10
    11  **Caution** the spirits don't like guests.
    12
    13  Good luck!!!
```

wc

Este comando se utiliza para contar todas las líneas caracteres y palabras de un texto o un fichero.

```
sysadmin@localhost:~/Documents$ cat alpha-first.txt
A is for Apple
B is for Bear
C is for Cat
D is for Dog
E is for Elephant
F is for Flower
```

```
sysadmin@localhost:~/Documents$ wc alpha-first.txt
 6 24 90 alpha-first.txt
```

6 líneas
24 palabras
90 caracteres (incluidos espacios en blanco)

head

Este comando mostrara las primeras líneas de un fichero, por defecto enseñara las primeras 10 líneas, siempre y cuando el fichero las contenga.

Por ejemplo

```
sysadmin@localhost:~/Documents$ head alpha.txt
A is for Apple
B is for Bear
C is for Cat
D is for Dog
E is for Elephant
F is for Flower
G is for Grapes
H is for Happy
I is for Ink
J is for Juice
```

Si quisiéramos mostrar un numero diferente de líneas podemos forzarlo de una de las dos siguientes maneras.

```
sysadmin@localhost:~/Documents$ head -3 alpha.txt
A is for Apple
B is for Bear
C is for Cat
```

```
sysadmin@localhost:~/Documents$ head -n3 alpha.txt
A is for Apple
B is for Bear
C is for Cat
```

tail

Muestra las últimas líneas de un fichero. Es similar al head pero en lugar de las primeras líneas muestra las ultimas.

```
sysadmin@localhost:~/Documents$ tail alpha.txt
Q is for Quark
R is for Rat
S is for Sloth
T is for Turnip
U is for Up
V is for Velvet
W is for Walrus
X is for Xenon
Y is for Yellow
Z is for Zebra
```

```
sysadmin@localhost:~/Documents$ tail -3 alpha.txt
X is for Xenon
Y is for Yellow
Z is for Zebra
sysadmin@localhost:~/Documents$ tail -n3 alpha.txt
X is for Xenon
Y is for Yellow
Z is for Zebra
```

split

Divide ficheros, normalmente el argumento suele ser el nombre del fichero que se desea dividir.

Si no se define nada por defecto los nombre de los ficheros de salida serán x seguido de aa, ab,etc

```
sysadmin@localhost:~/Documents$ split longfile.txt
sysadmin@localhost:~/Documents$ ls
School            alpha.txt    linux.txt     profile.txt   xac  xai
Work              animals.txt  longfile.txt  red.txt       xad  xaj
adjectives.txt    food.txt     newhome.txt   spelling.txt  xae  xak
alpha-first.txt   hello.sh     numbers.txt   words         xaf
alpha-second.txt  hidden.txt   os.csv        xaa           xag
alpha-third.txt   letters.txt  people.csv    xab           xah
```

También se puede definir el nombre del fichero de salida.

```
sysadmin@localhost:~/Documents$ split longfile.txt file.
sysadmin@localhost:~/Documents$ ls
School            animals.txt  file.ag   hidden.txt    people.csv    xac
Work              file.aa      file.ah   letters.txt   profile.txt   xad
adjectives.txt    file.ab      file.ai   linux.txt     red.txt       xae
alpha-first.txt   file.ac      file.aj   longfile.txt  spelling.txt  xaf
alpha-second.txt  file.ad      file.ak   newhome.txt   words         xag
alpha-third.txt   file.ae      food.txt  numbers.txt   xaa           xah
alpha.txt         file.af      hello.sh  os.csv        xab           xai
```

Con esta opción el sufijo serán letras con el formato mencionado anteriormente.

Si queremos que en lugar de letras sean números se tiene que usar la opción *–d*

```
sysadmin@localhost:~/Documents$ split -d longfile.txt file.
sysadmin@localhost:~/Documents$ ls
School            file.00  file.08  file.af   hidden.txt    profile.txt   x
Work              file.01  file.09  file.ag   letters.txt   red.txt       x
adjectives.txt    file.02  file.10  file.ah   linux.txt     spelling.txt  x
alpha-first.txt   file.03  file.aa  file.ai   longfile.txt  words         x
alpha-second.txt  file.04  file.ab  file.aj   newhome.txt   xaa           x
alpha-third.txt   file.05  file.ac  file.ak   numbers.txt   xab           x
alpha.txt         file.06  file.ad  food.txt  os.csv        xac           x
animals.txt       file.07  file.ae  hello.sh  people.csv    xad
```

De forma predeterminada el comando dividirá el fichero en fragmentos de 1000 líneas

Si se desea modificar el número de líneas se debe utilizar la opción *–l*

Si en lugar de número de líneas se desea dividir por tamaño se puede usar la opción *–b* indicando el máximo de bytes por archivo

paste

El comando paste fusiona línea alineas uno o más archivos separado con un tabulador como separador por defecto

```
sysadmin@localhost:~/Documents$ cat numbers.txt
1
2
3
4
5
sysadmin@localhost:~/Documents$ cat letters.txt
a
b
c
d
e
```

```
sysadmin@localhost:~/Documents$ paste numbers.txt letters.txt
1       a
2       b
3       c
4       d
5       e
```

También se puede cambiar el delimitador con la opción *–d*

```
sysadmin@localhost:~/Documents$ paste -d , numbers.txt letters.txt
1,a
2,b
3,c
4,d
5,e
```

join

Concatena contenidos de ficheros con un ID común. Si el ID usar es el primer campo entonces no es necesario especificarlo.
Usaremos estos dos ficheros para mostrar el funcionamiento

```
sysadmin@localhost:~/Documents$ cat adjectives.txt
1 golden
2 honey
3 fruit
4 grey
5 bald
sysadmin@localhost:~/Documents$ cat animals.txt
1 retriever
2 badger
3 bat
4 wolf
5 eagle
```

```
sysadmin@localhost:~/Documents$ join adjectives.txt animals.txt
1 golden retriever
2 honey badger
3 fruit bat
4 grey wolf
5 bald eagle
```

Al contrario de *join*, *paste* y *cat* concatenan de forma diferente.

```
sysadmin@localhost:~/Documents$ paste adjectives.txt animals.txt
1    golden    1    retriever
2    honey     2    badger
3    fruit     3    bat
4    grey      4    wolf
5    bald      5    eagle
sysadmin@localhost:~/Documents$ cat adjectives.txt animals.txt
1    golden
2    honey
3    fruit
4    grey
5    bald
1    retriever
2    badger
3    bat
4    wolf
5    eagle
```

Con la opción *–t* utilizaremos un delimitador alternativo

```
sysadmin@localhost:~/Documents$ cat people.csv
Dennis,Richie
Andrew,Tanenbaum
Ken,Thompson
Linus,Torvalds
sysadmin@localhost:~/Documents$ cat os.csv
1970,Unix,Richie
1987,Minix,Tanenbaum
1970,Unix,Thompson
1991,Linux,Torvalds
```

```
sysadmin@localhost:~/Documents$ join -1 2 -2 3 -t',' people.csv os.csv
Richie,Dennis,1970,Unix
Tanenbaum,Andrew,1987,Minix
Thompson,Ken,1970,Unix
Torvalds,Linus,1991,Linux
```

join -1 2 -2 3 –t ‘,’ people.csv os.csv
join -1 2 -2 3 –t ‘,’ *people.csv os.csv* → del primer fichero “people.csv”
join -1 2 -2 3 –t ‘,’ *people.csv os.csv* → usa como referencia la columna 2
join -1 2 -2 3 –t ‘,’ *people.csv os.csv* → del fichero 2 “os.csv”
join -1 2 -2 3 –t ‘,’ *people.csv os.csv* → usa como referencia la columna 3
join -1 2 -2 3 –t ‘,’ people.csv os.csv → el limitador, separador de columnas será el carácter coma

sort

El comando sort se utiliza para mostrar un archivo ordenado en un campo especifico de datos.

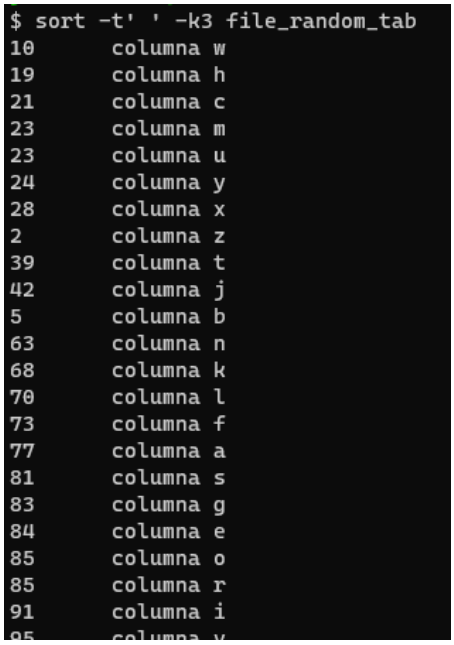
El uso por parámetros del comando sort ordena por la primera columna en orden ASCII.

Las columnas por defecto están separadas por comas, para especificar otro separador usaremos la opción –t

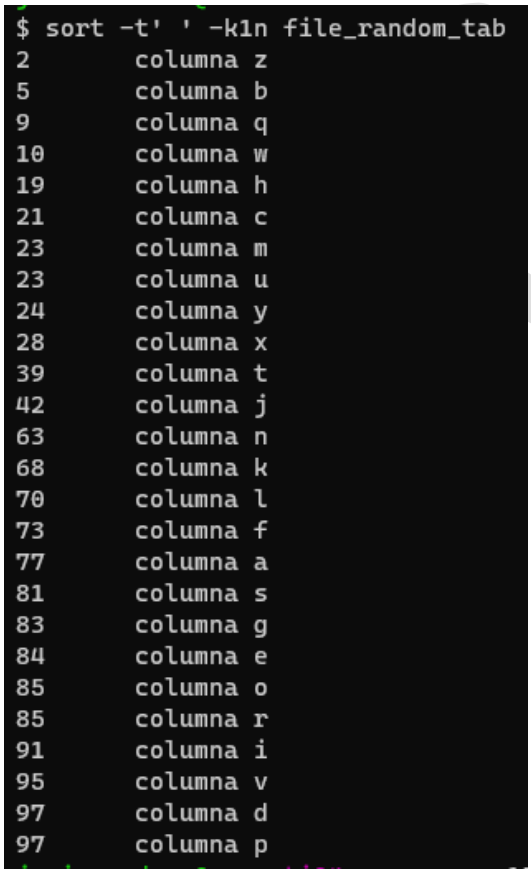
sort –t’,’ fichero_a_ordenar

Si queremos ordenar por otra columna usaremos el comando –k seguido de numero de la columna, (la primera columna es la 1, no la 0).

sort –t’,’ –k1 fichero_a_ordenar



Está ordenando en ASCII, si deseamos que ordene interpretándolo como un numero añadiremos la letra n después del número de columna



Si quisiéramos invertir el orden añadiremos la opción r

Si añadimos la opción –u eliminaremos las líneas duplicadas consecutivamente (después de ordenarlas). Es decir, primero las ordena y luego borra las duplicada.

uniq

Elimina líneas duplicadas consecutivas de un fichero

Con la opción –c somos capaces de contar las líneas duplicadas

cut

Corta el contenido de un fichero.

Opciones:
-*c*: permite seleccionar el o los caracteres que desea cortar
-*f*: permite selecciona porque bloque o bloques se desea cortar.
-*d*: permite cambiar el delimitador

tr

Este comando traslada caracteres. Si se define dos conjuntos de caracteres el comando retornara la sustitución del segundo por el primero, posición a posición.

Veamos unos ejemplos para entender mejor el funcionamiento

```
$ head -3 file_random
24, columna a
28, columna b
85, columna c
javiercadena@vmsarti04:~ mar nov 15 <17:42:34>
$ head -3 file_random | tr '1234567890' 'isekSgFBPO'
sk, columna a
sB, columna b
BS, columna c
```

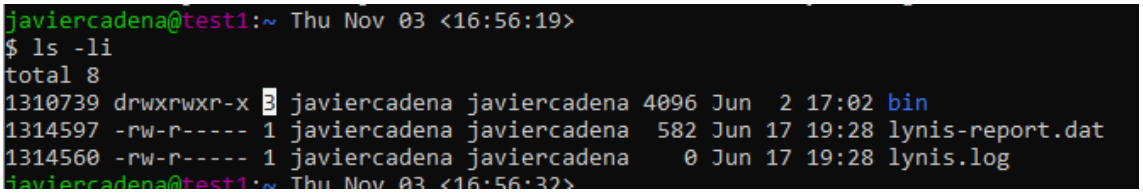
```
$ head -3 file_random | tr 'aeiou' '43107'
24, c0l7mn4 4
28, c0l7mn4 b
85, c0l7mn4 c
```

```
$ head -5 file_random | tr 'abcd' '@'
24, @olumn@ @
28, @olumn@ @
85, @olumn@ @
64, @olumn@ @
76, @olumn@ e
```

Enlaces de sistema de ficheros

Los enlaces, son conexiones entre una y otras ubicaciones, carpetas o archivos.

El número de recuentos de enlace duros sale cuando nos realizamos un listado con la opción –l



Un fichero es un bloque de información que ocupa un espacio en el disco. Esta información serán datos en función de estos datos tendremos un archivo o una carpeta, etc. .

Siempre que se pueda se recomienda utilizar enlaces suaves.

Enlace suave

Los enlaces suaves, sería equivalentes a lo que en Windows son los accesos directos, sería un fichero que apunta a la ubicación del otro fichero.

Formato: *ln –s <Fichero al que apunto> <nombre del fichero que apunta>*

Enlace duro

En enlace duro es similar al enlace suave con la diferencia de que el en lace duro no apunta el otro fichero si no que apunta directamente al inodo, es como duplicar el archivo sin realizar la copia.

De esta forma si borramos en fichero al que apunta un enlace suave este se queda sin ningún sitio a donde apuntar, por otro lado, en el enlace duro esto no pasaría.

Formato*: ln <Fichero al que apunto> <nombre del fichero que apunta>*

Practica

Crea un fichero de la siguiente forma: echo “una frase” > *fichero2*

Hace una copia de *fichero2* a *fichero3*

A continuación, crea un enlace blando a fichero2, lista con detalle la información y da especial atención a el tamaño del archivo y al tamaño del enlace. ¿Qué observas?
Saca un nuevo listado mostrando los inodos.
Borrar el archivo “*fichero2*” y saca otra vez el listado ¿Qué ha sucedido y por qué?

Haz lo mismo usando el *fichero3* pero esta vez con un enlace duro, Analizar los resultados ¿Qué conclusiones apreciáis?

Gestión de usuarios

Los sistemas GNU-Linux están muy centrado en permisos usuario, grupos, por ello la gestión de usuarios es un punto importante en los sistemas GNU-Linux.

Los usuarios en sistemas GNU/Linux se identifican por un numero de usuario llamado User ID o UID, el cual pertenecerá a un grupo principal llamado Group ID o GID. Los usuarios pueden pertenecer a tantos grupos secundarios como sea necesario.

Las definiciones de las cuentas de usuario están definidas en el fichero */etc/passwd*. Este archivo contiene una línea por cada usuario, sea de sistema o sea un user físico (“real”).

Un ejemplo de línea será:

```
sergio:x:501:500:Sergio González:/home/sergio:/bin/bash
```

Los campos están separados por el carácter ":" y son:

Campo 1	Es el nombre de usuario, el nombre que usara para identificarse, por ejemplo en el proceso de login
Campo 2	La "x" indica que hay una contraseña encriptada para este usuario, antiguamente esta casilla se usaba para poner la contraseña. En los sistema actuales la contraseña se guarda cifrada en el fichero */etc/shadow*
Campo 3	Es el UID del usuario es el número de identificación del usuario.
Campo 4	Es el GID, el número de identificación de grupo
Campo 5	Es un campo de comentarios que se puede usar para poner el nombre completo del user
Campo 6	Es la ruta a la carpeta home del Usuario.
Campo 7	Es la Shell usada por el usuario.

Crear usuario

Existen básicamente dos comandos para crear usuario, *useradd* y *adduser*.

adduser es un "script" que utiliza *useradd* que utiliza una guía para crear usuarios.

Opciones

Ya conocemos los campos que definen a los usuarios vamos ahora a rellenarlos. Solo comentar que para añadir un grupo al usuario este debe de existir, con lo que en el caso de usar *useradd* deberíamos de crear previamente el grupo.

-b, --base-dir *BASE_DIR*

define la carpeta raíz que se concatenara con el nombre de la cuenta para formar el home directori. En el caso que esta carpeta no exista se debe usar la opción *–m* para que se cree

-c, --comment *COMMENT*

Permite añadir el campo de comentario.

-d, --home *HOME_DIR*

Es el home directorio, la carpeta de inicio por defecto cuando el usuario entre en el sistema. Normalmente es la concatenación de directorio base más el nombre de login.

-g, --gid *GROUP*

Nombre del grupo principal

-G, --groups *GROUP1[,GROUP2,..[,GROUPN]]*

Lista de grupos a los que pertenece el usuario.

-s, --shell *SHELL*

Crear grupo

Para crear grupos tenemos el comando *groupadd*

groupadd <nombre del grupo>

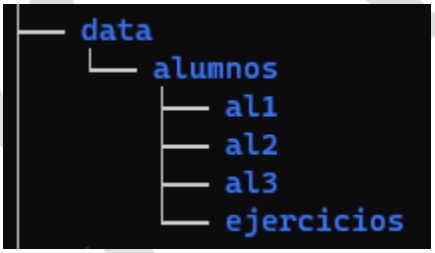
Para eliminar grupos y usuarios existen los comandos ***groupdel*** y ***userdel*** respectivamente.
También se dispone del comando ***usermod*** para modificar a un usuario.
Existen muchos otros parámetros que podemos controlar de los usuarios que no veremos aquí. Tales como, dejar a un user inactivo (para que no pueda loguearse) poner fechas de caducidad, poner fechas de expiración de contraseñas, etc.

Practica

Si quisiéramos crear 3 usuario para una clase con 3 alumnos Al1, Al2, Al3 cada uno de ellos con su directorio home personal. Los tres pertenecientes al grupo principal llamado *alumnos*, las carpetas principales de estos usuarios deben estar ubicadas en las rutas */data/alumnos/al1*, */data/alumnos/al2*, */data/alumnos/al3* respectivamente

Se tiene que crear una carpeta común a los tres usuarios llamada */data/alumnos/ejercicios* donde todos los alumnos podrán acceder, leer y escribir.

En cada una de las carpetas personales de los usuarios, "(*al1*, *al2*, *al3*)", podrán acceder todos los alumnos, pero solo podrán manipularlo el propietario



La shell por defecto de los tres usuarios creados tiene que ser "*sh*"

Se pide.

- Hacer un esquema/estudio de los pasos a seguir y necesidades.
- Listar todos los pasos y los comandos con sus opciones necesarios para crear todos estos requisitos.

Anexo-1 Solución de los ejercicios.

Ejercicio 1

Viendo el diagrama de bloques anterior. Cuando un user entre en entorno gráfico y abre la aplicación terminal ¿está abriendo un Shell?

Solución

En realidad, no se está abriendo un Shell en el sentido estricto, en realidad se está abriendo un **virtual Shell** o un **interactive Shell**, para abrir un Shell se tendría que cerrar el entorno gráfico.

Ejercicio- 2

Conociendo que el comando "ls" lista el contenido
Verificar en vuestra consola que los comando realizados anteriormente se cumplen.
Usar también las opciones –t, -r, -d para que sucede.

Solución

```
javiercadena@test1:~ Fri Oct 28 <16:01:21>
$ ls -l
total 8
drwxrwxr-x 3 javiercadena javiercadena 4096 Jun  2 17:02 bin
-rw-r----- 1 javiercadena javiercadena  582 Jun 17 19:28 lynis-report.dat
-rw-r----- 1 javiercadena javiercadena    0 Jun 17 19:28 lynis.log
```

```
javiercadena@test1:~ Fri Oct 28 <16:01:25>
$ ls -lt
total 8
-rw-r----- 1 javiercadena javiercadena  582 Jun 17 19:28 lynis-report.dat
-rw-r----- 1 javiercadena javiercadena    0 Jun 17 19:28 lynis.log
drwxrwxr-x 3 javiercadena javiercadena 4096 Jun  2 17:02 bin
javiercadena@test1:~ Fri Oct 28 <16:03:31>
$ ls -ltr
total 8
drwxrwxr-x 3 javiercadena javiercadena 4096 Jun  2 17:02 bin
-rw-r----- 1 javiercadena javiercadena    0 Jun 17 19:28 lynis.log
-rw-r----- 1 javiercadena javiercadena  582 Jun 17 19:28 lynis-report.dat
```

La opción –t ordena por tiempo y la opción –r ordena en orden inverso.

La opción –d añade directorios.

Ejercicio- 3

Listar del directorio actual, todos los ficheros "visibles" ordenados por fecha de creación en orden ascendente

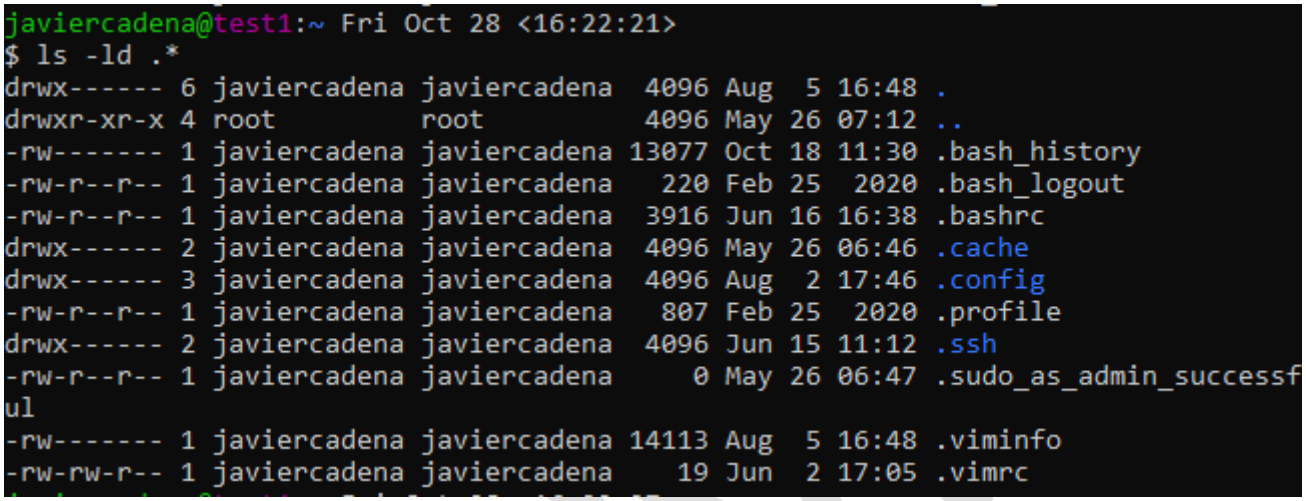
Solución

```
javiercadena@test1:~ Fri Oct 28 <16:03:33>
$ ls -ltr
total 8
drwxrwxr-x 3 javiercadena javiercadena 4096 Jun  2 17:02 bin
-rw-r----- 1 javiercadena javiercadena    0 Jun 17 19:28 lynis.log
-rw-r----- 1 javiercadena javiercadena  582 Jun 17 19:28 lynis-report.dat
```

Ejercicio- 4

Listar el contenido de los archivos y carpetas “ocultos” de tu carpeta actual. (solo el nivel actual)

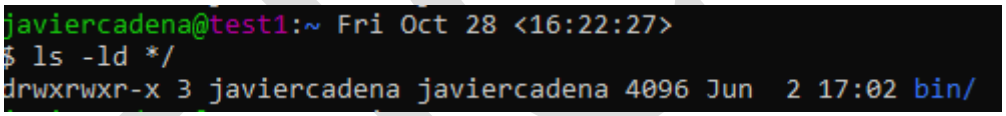
Solución



Ejercicio- 5

Lista solo las carpetas no ocultas contenidas en tu carpeta actual.

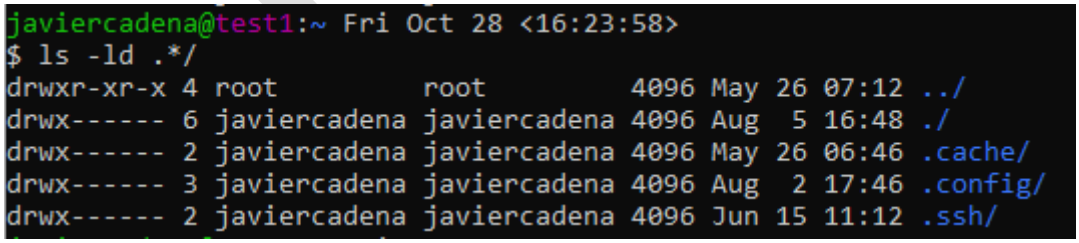
Solución



Ejercicio- 6

Lista solo las carpetas ocultas contenidas en tu carpeta actual.

Solución



Ejercicio – 7

Verifica que sucede si usas dos veces seguidas el mismo comando.

¿Se queda registrado en el historial?

Si las dos instrucciones son consecutivas y son exactamente iguales, el historial solo almacena una de ellas.

Ejercicio – 8

Ejecuta el comando anterior de tres formas diferentes.

```
  786  history
javiercadena@test1:~ Mon Oct 31 <13:53:36>
$ ls
bin  lynis-report.dat  lynis.log
javiercadena@test1:~ Mon Oct 31 <13:53:44>
$ ls
bin  lynis-report.dat  lynis.log
javiercadena@test1:~ Mon Oct 31 <13:53:50>
$ !!
ls
bin  lynis-report.dat  lynis.log
javiercadena@test1:~ Mon Oct 31 <13:53:54>
$ !787
ls
bin  lynis-report.dat  lynis.log
```

La primer que hacemos es ver cuál es el último elemento del historial, en este caso es 786 y es la instrucción de ver el historial

A continuación, lanzamos un comando

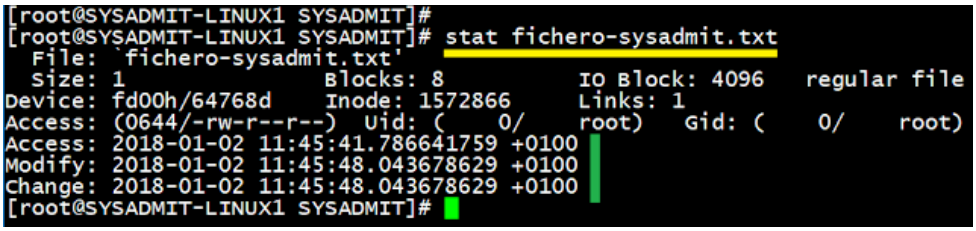
La primera forma de repetir el ultimo comando presionamos el botón fecha subir

La segunda es usando la opción de ultimo comando “!!”

La tercera es usando “!<numero posición historial>”

Anexo-2

Fijémonos en este ejemplo:



Con el comando stat, al indicar como parámetro un fichero, vemos que aparece:

Access: Se actualiza cuando abrimos el fichero, por ejemplo, cuando utilizamos comandos tipo: car, grep, tail, etc..

Modify: Se actualiza cuando cambiamos el contenido del fichero.

Change: Se actualiza cuando cambiamos permisos, propietario: usuario, grupo, etc...

También, dependiendo de la distribución de Linux que vayamos a utilizar, podremos ver que se utiliza: atime, ctime o mtime.

atime: Access time

mtime: Modify time

ctime: Change time

Cuando ejecutamos ls -l, estamos listando el: mtime

Cuando ejecutamos ls -lu, estamos listando el: atime

Cuando ejecutamos ls -lc, estamos listando el: ctime

Para cambiar cada uno de ellos, utilizaremos el comando: touch. El formato para especificar la fecha con touch, es: YYMMDD: Año, mes, día, hora, minuto. o si también queremos especificar hora:
YYMMDDhhmm: Año, mes, día, hora, minuto.

Con parámetros específicos del comando touch, podremos cambiar directamente la fecha de: Access y Modification, pero no de Change.

Para cambiar la fecha de Change, deberemos cambiar temporalmente la hora del sistema.

Especial atención con los servicios que están funcionando sobre el equipo con cambiar la hora del sistema, puede que haya algún servicio que deje de funcionar o su funcionamiento no sea correcto.

Clases:

03/1/2022 Primera clase Ikram. De 10h a 11:10 Temario de inicia a sintaxis de comandos → Opciones (pag. 6)

15/11/2022 Segunda clase: Ikram y David de 9:30 a 11h desde hoja 6 Comandos, hasta hoja 13 Globbing. Próxima clase Manipulación de ficheros.

Nombre	Apellido	Correo electrónico	Duración	Hora a la que se unió		Hora a la que salió
Ikram	Bghiel	ikram.bghiel@upc.edu	1 h 58 min	9:20		11:18
Javier	Cadena Muñoz	javier.cadena@upc.edu	2 h	9:18		11:18
David	Sarriá	david.sarria@upc.edu	1 h 32 min	9:46		11:18

21/11/2022 tercera clase: Ikram David de 10:00 a 10:50 desde la hoja 6 hasta hoja 16.

Nombre	Apellido	Correo electrónico	Duración	Hora a la que se unió	Hora a la que salió
Ikram	Bghiel	ikram.bghiel@upc.edu	54 min	9:59	10:52
Javier	Cadena	javier.cadena@upc.edu	59 min	9:53	10:52
David	Sarriá	david.sarria@upc.edu	52 min	10:01	10:52

28/11/2022 cuarta clase: Ikram, David de 09:50 a 11:00 desde la página 16 hasta la página 25 el próximo comando tiene que ser “split”.

Nombre	Apellido	Correo electrónico	Duración	Hora a la que se unió	Hora a la que salió
Ikram	Bghiel	ikram.bghiel@upc.edu	1 h 18 min	9:44	11:02
Javier	Cadena	javier.cadena@upc.edu	1 h 19 min	9:43	11:02
David	Sarriá	david.sarria@upc.edu	1 h 14 min	9:48	11:02

Anexo- 3. Consultas

15/11/2022 Ikram: que hace la opción –F del ls

Contestación:

Añade formato al listado con detalles.
Por ejemplo:

añade el carácter / al final del nombre de la carpeta

*añade el carácter * al final del nombre de los archivos ejecutables*