GUIÓN 1
INTRODUCCIÓN A LINUX

Bibliografía

Sarwar, S. M.; Koretsky, R.; Sarwar, S. A. El libro de LINUX. Addison Wesley, 2005.
Sobell, Mark G. Manual práctico de Linux. Comandos, editores y programación Shell. Anaya Multimedia, 2008
Sánchez Prieto, S. UNIX y LINUX. Guía práctica (Tercera edición). Ra-Ma, D.L. 2004.
Pons, N. Linux. Principios básicos de uso del sistema. Ediciones ENI, 2011.

El Sistema Operativo Linux es indudablemente uno de los más importantes de nuestro tiempo. Se ha extendido de forma generalizada sobre todo para los computadores pequeños, situados por encima de los computadores personales. Se trata de un sistema con desarrollo jerárquico y que gestiona adecuadamente los recursos de un sistema computacional, considerando que va a dar soporte a varios programas (multiprogramación), a varios usuarios (multiusuario) y a máquinas remotas (redes), sin olvidar nunca todas las cuestiones referentes a la protección y seguridad, tan importantes en este tipo de sistemas.

1. CONEXIÓN CON EL SISTEMA (ESTABLECER UNA SESIÓN)

Antes de trabajar con la máquina es necesario que ésta nos reconozca como usuario; por tanto es necesario identificarnos. Esta operación se denomina **login**. Cuando ejecutamos el software de conexión al "host", quedamos conectados físicamente al sistema, y la primera información que aparece sobre la pantalla es el denominado "saludo de recepción", junto con el requerimiento de que se introduzca el **login** del usuario, es decir, cuál es el nombre con el que el sistema conoce al usuario que intenta conectarse.
Se supone que el usuario ya tiene asignado un nombre de entrada (**login**). Una vez introducido éste, el sistema pedirá posteriormente la contraseña (**password**).

> ***Nota:*** *El sistema distingue entre mayúsculas y minúsculas por lo tanto, es conveniente tenerlo en cuenta.*

El sistema operativo y los equipos con los que trabajamos nos proporcionan un entorno multiusuario. Por este motivo es preciso respetar algunas reglas de protección y de acceso a los datos particulares de cada uno. Por ello, cada usuario tiene asignada una contraseña (*password*), que él sólo conoce, y que es necesario proporcionar al sistema para conectarse a él. Por supuesto, la contraseña no se visualiza en la pantalla.
Al realizar la conexión es posible que se prescinda de la contraseña (puede ocurrir la primera vez que un usuario nuevo se conecta). Sin embargo, todo el mundo debería tener una contraseña y cambiarla a intervalos de tiempo razonablemente frecuentes por razones de seguridad. De hecho, en algunas

versiones de Linux, el sistema obliga a cambiar la contraseña a los usuarios cuando exceden un tiempo ya predeterminado sin actualizarla. Una vez proporcionada la contraseña, el sistema visualizará un mensaje de bienvenida. Desde este momento, se pone a la espera de cualquier mandato a través del ***prompt***, que está representado por el carácter dólar ($).

login as: sbperez
password:
Last login: Thu Sep 10 10:44:39 2017 from *nombre (o IP) de la máquina desde la que se conectó*
sbperez@EIXE:~$

Nota: *Observar que aparece la fecha y hora de la última conexión al sistema del usuario, esto es muy útil para detectar si alguien ha logrado entrar en el sistema con nuestro nombre de usuario.*

Cuando nos conectamos al sistema, el diálogo se inicia mediante un programa especial conocido por **shell** (Intérprete de comandos). Por cada usuario que se conecta se crea un ejemplar particular de dicho programa. El **shell** interpreta los comandos introducidos en un terminal por el usuario y los transmite al sistema en un lenguaje y una sintaxis comprensibles para éste. El *prompt* que se muestra (sbperez@EIXE:~$), es el que utiliza el intérprete de comandos escogido (**bash**). El sistema permite elegir a los usuarios entre varios **shell** de los que dispone. Cada uno de ellos suele tener un *prompt* distinto por defecto.
La sintaxis general o estructura de un comando, cuando éste se escribe en la línea de comandos, es como sigue:

$ comando [[-]opción(es)] [argumento(s) de opción(es)] [argumentos de orden(es)] [-- help]

dónde:

$	indicador del Shell, que puede ir precedido por otros caracteres (en nuestro caso el signo de dólar($) va precedido por el nombre del usuario@EIXE:~)
comando	nombre de un comando de Linux válido para ese shell, escrito en letras minúsculas
[ ]	cualquier cosa que esté encerrada entre [ ] será opcional
[[-]opción(es)]	uno o más modificadores que cambian el comportamiento de un comando
[argumento(s) de opción(es)]	uno o más modificadores que cambian el comportamiento de las opción(es)
[argumentos de orden(es)]	uno o más objetos que serán afectados por el comando
[-- help]	proporciona ayuda sobre las opciones del comando (no todas las órdenes utilizan --help para mostrar la información de ayuda)

El carácter (;) sirve para separar más de un mandato dentro de la misma línea de órdenes.

```
Ejemplo:
       cp *  dir1; clear; ls  dir1
```

2. COMANDOS BÁSICOS

exit. Fin de sesión

Utilizando este comando destruimos el **shell** en el que nos encontramos. Se trata de la **forma correcta** de cerrar una sesión y salir del sistema.

Sintaxis: exit

passwd. Cambio de contraseña

Se trata del comando que permite cambiar la contraseña (o crear una si todavía no tiene). La contraseña deberá incluir un mínimo de seis caracteres y estar compuesta de letras (minúsculas o mayúsculas) y cifras. Este comando es interactivo en el sentido de que va indicando lo que quiere que introduzca. Primero pide la antigua contraseña y posteriormente la nueva contraseña debe ser escrita dos veces. Esto es debido a que el sistema no hace eco de las contraseñas mientras que se escriben, y por tanto es necesario pedir confirmación. Existe un fichero que contiene los datos de los usuarios y, usualmente, todos los usuarios tienen acceso, por lo que para preservar la seguridad de las contraseñas, éstas están encriptadas. El encriptamiento de la contraseña significa que nadie, ni siquiera el *superusuario*, puede averiguar cuál es a partir de la versión encriptada, así que no se debe olvidar la contraseña.

Sintaxis: passwd

> ***Nota:*** *Por supuesto, el superusuario puede asignar una nueva "password" (contraseña) a cualquier usuario, lo que no puede hacer es averiguar cuál es la contraseña.*

man. Muestra el manual de sistema

Siguiendo la tendencia actual de la mayoría de los programas, las versiones modernas de Linux no suelen venir provistas de "manuales" en formato de libros, sino que casi toda la ayuda, manuales e información sobre el sistema se encuentran en soporte electrónico, accesible si se ejecutan los mandatos adecuados.

El comando **man** sirve para consultar el manual de usuario de forma interactiva. Contiene la información referente a cada comando Linux. Para detener el comando (salir) se debe pulsar la tecla *q*.

Sintaxis: man [-k] nombre_comando

Opciones:

1. La opción **–k** (*keyword*, palabra clave), busca por palabra clave.

El comando **apropos** tiene la misma función que el comando man -k

Otro comando que proporciona información de ayuda sobre los comandos es **info**.

```
Ejemplos:
       man date
       man -k date
       apropos date
       info date
```

echo. Copia sus argumentos en la pantalla

Se encarga de enviar a la pantalla los caracteres que siguen a dicho comando y terminados en una nueva línea.

Sintaxis: echo [-ne] *texto a visualizar*

El texto puede ir entre comillas o sin comillas.

Opciones:

1. La opción **–n**, no introduce el salto de línea.
2. La opción **– e**, activa la interpretación de caracteres de barra invertida. En este caso, el texto tiene que ir entre comillas.

Código	Significado
\b	Retroceso de espacio
\f	Salto de página
\n	Salto de línea
\r	Retorno de carro
\t	Tabulador horizontal
\v	Tabulador vertical
\a	Alerta (pitido)
\"	Comillas dobles

```
Ejemplos:
        echo esto es una prueba
        echo "esto es una prueba"
        echo –n "esto es una prueba"
        echo -e "esto \n es una \t prueba"
```

date. Muestra la hora y la fecha

Permite obtener la fecha y hora del sistema. El formato de la salida se puede especificar precedido por el carácter + (ver la sintaxis con el comando man).

Sintaxis: date

```
Ejemplos:
        date
        date +%x
        date +"Hoy es %x"
```

cal. Visualiza un calendario

Esta orden sirve para manejar un calendario muy completo (desde el año 1 hasta el año 9999). Los años deben darse completos, (96 lo interpretaría como el año 96 de nuestra era).

Sintaxis: cal [[mes] año]

Si se ejecuta sin argumentos, nos muestra el mes en que estamos.

```
Ejemplos:
        cal
        cal 2013
        cal 9 2013
```

clear. Limpia la pantalla

Este comando sirve para limpiar la pantalla.

Sintaxis: clear

who. Listado de quien se encuentra en el sistema

Con el mandato **who** obtenemos la lista de los nombres de los usuarios conectados al sistema actualmente, el terminal en que se encuentran, además de la fecha y la hora en que se conectaron. La información que muestra who es útil cuando se quiere comunicar algo a otro usuario.

Sintaxis: who [am i] [-q]

Opciones:

La opción **–q**, muestra el número de usuarios conectados al sistema y sus nombres.

Si se introduce con los parámetros *am i* se obtiene el nombre del usuario que está conectado en el terminal. Puede ser útil cuando una persona tiene varios nombres de entrada (login). En un entorno multiusuario siempre se tiene la impresión de estar solo con la máquina, ya que la presencia de otros usuarios es totalmente transparente. Si en un momento dado hay una gran concentración de usuarios, se puede notar una pérdida de eficiencia en algunas operaciones.

```
Ejemplos:
        who
        who am i
        whoami
        who –q
```

finger. Lista usuarios del sistema

Se puede utilizar el comando **finger** para ver información más detallada sobre los usuarios que se encuentran en el sistema. Además de los nombres de inicio de sesión (login), finger proporciona el nombre completo junto con información del dispositivo que cada uno utilizó para conectarse, tiempo de inactividad,…

Sintaxis: finger [usuario]

Opciones:

La opción **usuario**, muestra un listado de todos aquellos usuarios en cuyo nombre completo o login está la cadena **usuario**. El comando finger no distingue entre mayúsculas y minúsculas.

```
Ejemplos:
        finger
        finger  Perez
```

id. Muestra datos del usuario

Este comando también nos da información del usuario conectado al terminal. Nos devuelve su UID (Identificador de usuario) y su GID (Identificador de grupo).

Sintaxis: id

3. TECLAS PARA EDITAR LA LÍNEA DE COMANDOS

Combinación de teclas	Descripción
ctrl. + l	Es equivalente al comando clear ya que limpia la pantalla.
ctrl. + a	Coloca el cursor al principio de la línea.
ctrl. + e	Coloca el cursor al final de la línea.
ctrl. + k	Elimina desde el cursor al final de la línea.
tab	Autocompleta (comando o nombre de fichero/directorio)
ESC	Autocompleta (comando o nombre de fichero/directorio). Requiere que se presione dos veces.
↑↓	Recupera los comandos introducidos anteriormente.

4. SISTEMA DE FICHEROS

El sistema de ficheros consta, entre otros elementos, de ficheros y directorios. Un fichero contiene información relacionada entre si por algún criterio, mientras que un directorio es una estructura organizativa sin información que puede contener ficheros y otros directorios (subdirectorios).

El Sistema Operativo asocia a cada usuario dos directorios:

- *directorio de inicio,* también denominado directorio de recepción o personal, que es un subdirectorio de la propiedad del usuario en el que se sitúa automáticamente cuando se conecta al sistema. Este directorio no varía durante toda la sesión de trabajo y cuelga normalmente del directorio *home*. Además, su existencia se debe a que se trata de un entorno multiusuario, por lo que se reserva para cada usuario ese espacio de trabajo con el fin de que nadie llegue a interferir

los dominios de otro.

- *directorio de trabajo actual* también denotado por directorio activo, es el subdirectorio, dentro del sistema de ficheros, donde el usuario se encuentra en cada momento y sobre el que realiza sus acciones siempre que no especifique lo contrario. Como se verá posteriormente existen comandos que permiten variar este citado directorio.

Cuando el usuario se conecta al sistema su directorio de trabajo actual coincide con su directorio de inicio.

Teniendo en cuenta estos dos directorios, un fichero o un directorio pueden identificarse de dos formas distintas:

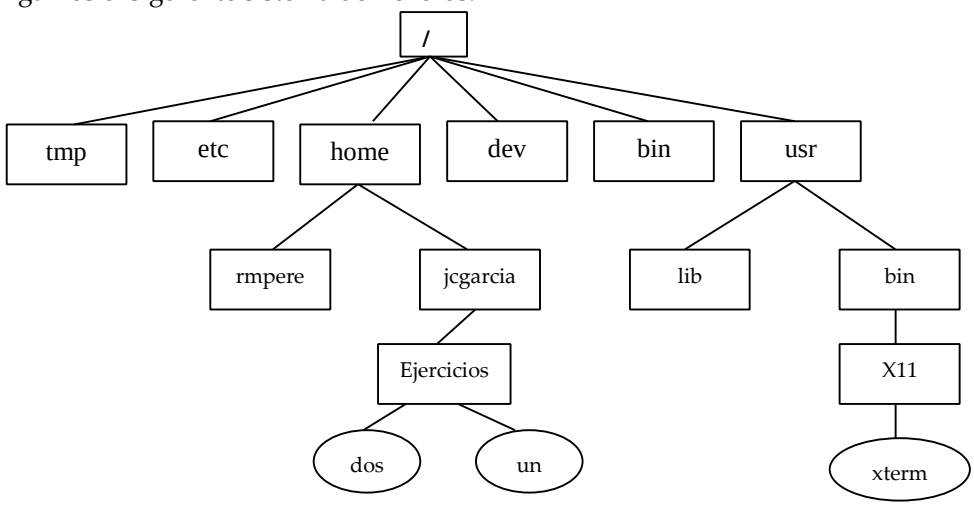
- mediante su camino absoluto o dirección completa que parte del directorio raíz (*pathname* o ruta de acceso absoluta),
- mediante su camino relativo o dirección relativa que parte del directorio de trabajo actual (ruta de acceso relativa).

Los nombres de los ficheros deben ser únicos dentro del directorio pero no dentro del sistema de ficheros.

Nota: *Los directorios que componen los caminos en Linux se separan usando la barra inclinada a la derecha (ej.:* ***/home/jcgarcia/Ejercicios****). Esta barra inclinada también se usa para representar al directorio raíz (/).*

- / o raíz: El directorio raíz siempre se denota por una / (barra) y se indica con este carácter.
- Un punto (.) es sinónimo del nombre de la ruta del directorio de trabajo actual.
- Un punto doble (..) es sinónimo del nombre de la ruta del padre del directorio de trabajo actual.
- Signo de equivalencia, tilde o virgulilla de la ñ (~) es sinónimo de la ruta absoluta al directorio de inicio. Por lo que su uso permite abreviar parte de la ruta absoluta, en concreto la que corresponde al directorio de inicio del usuario. (*Alt+126* ó *AltGr+4*)

Supongamos el siguiente sistema de ficheros:



Para el usuario cuyo directorio de inicio es jcgarcia, los posibles caminos para identificar al fichero dos, siendo el directorio /usr/lib el directorio de trabajo actual, son:

- /home/jcgarcia/Ejercicios/dos (ruta absoluta)
- ~/Ejercicios/dos (ruta absoluta)
- ../../home/jcgarcia/Ejercicios/dos (ruta relativa)

5. ORDENES RELACIONADAS CON DIRECTORIOS

pwd. Muestra la ruta del directorio de trabajo actual

En muchos sistemas Linux, el *prompt* no visualiza el directorio en el que estamos trabajando (las versiones modernas si suelen incluirlo). El comando **pwd** (Print Working Directory) nos devuelve por pantalla el pathname completo del directorio de trabajo actual.

Sintaxis: pwd

mkdir. Crea un directorio

Este comando permite crear los directorios especificados en nombre(s)_directorio(s).

Sintaxis: mkdir [-pv] nombre(s)_directorio(s)

Opciones:

1. La opción **–p** crea los directorios padres que no existan en las rutas especificadas.
2. La opción **–v** visualiza el nombre de los directorios a medida que los crea.

```
Ejemplos:
       mkdir primero
       mkdir –p primero/segundo/tercero
       mkdir cuarto quinto
```

cd. Cambia de un directorio a otro

Sirve para cambiar de directorio. Se puede indicar el "pathname" completo o uno relativo al directorio actual.

Sintaxis: cd [camino_directorio]

Si no ponemos ningún argumento, este comando nos devuelve al directorio de inicio. Existen dentro de cada directorio dos directorios "especiales" que representan al padre (**..**) y al propio directorio (**.**)

```
Ejemplos:
       cd primero
       cd
       cd primero/segundo
       cd ..
       cd /home/pcperez/cuarto
```

rmdir. Elimina un directorio

Se utiliza para borrar los directorios vacíos que se especifiquen en nombre(s)_directorio(s).

Sintaxis: rmdir nombre(s)_directorio(s)

```
Ejemplos:
        rmdir /home/pcperez/primero
        rmdir quinto
```

Nota: *Para eliminar un directorio que no está vacío se utiliza el comando* ***rm*** *con la opción* ***–r.***

6. ÓRDENES RELACIONADAS CON FICHEROS

tee. Crea un fichero de texto

Este comando copia los datos que recibe de la entrada estándar a la salida estándar y/o a los archivos indicados en lista_ficheros.

Sintaxis: tee [-a] [lista_ficheros]

Nota: *Los nombres de ficheros en Linux pueden ser cualquier cadena de caracteres. Su longitud puede variar según la versión. En muchos casos, llega hasta 255 caracteres.*

Opciones:

1. La opción **-a** inserta los datos al final de los ficheros.

```
        tee fich1.txt
        .... Aquí se escribirían las líneas que quisiéramos...
        para acabar y volver al shell se debe teclear ctrl.+ d en una nueva línea

        tee fich2.txt
        tee -a fich1.txt fich2.txt
```

cat. Muestra el contenido de un fichero de texto

Este comando nos sirve para visualizar y concatenar en vertical los ficheros contenidos en lista_ficheros.

Sintaxis: cat [lista_ficheros]

Si ponemos un nombre de fichero existente, el resultado de este comando es la visualización por pantalla del contenido del fichero. Como en otros sistemas operativos, en algunos ficheros puede haber problemas en su visualización (p.e. si son ficheros ejecutables). Este comando sólo es útil para visualizar ficheros pequeños (deben caber en una pantalla) ya que no permite ir deteniendo la visualización.

```
Ejemplos:
        cat fich1
        cat fich1 fich2
```

touch. Crea un fichero vacío

Este comando nos sirve para crear uno o varios ficheros sin contenido. Si ponemos como nombre de fichero uno existente, este comando no elimina el contenido de dicho fichero pero si actualiza la fecha

de la última modificación.

Sintaxis: touch [lista_ficheros]

more. Visualiza un fichero de texto

Este comando nos permite concatenar en vertical y visualizar de una forma cómoda los ficheros contenidos en lista_ficheros, ya que visualiza página a página su contenido. La visualización se detiene después de cada página y se deberá pulsar **<Return>**, si sólo quiere avanzar una línea del fichero, o **<Space>** para visualizar la siguiente página.

Sintaxis: more [lista_ficheros]

less. Visualiza un fichero de texto

Este comando nos permite concatenar en vertical y visualizar de una forma cómoda los ficheros contenidos en lista_ficheros, permitiendo una completa navegación por el contenido de los ficheros, con avance/retroceso de página (o línea a línea) y movimiento lateral.

Sintaxis: less [lista_ficheros]

Para la navegación se utilizan las teclas de Flechas, AvPág, RePág, Inicio y Fin.

ls. Lista los nombres de los ficheros

Este comando ofrece un listado en pantalla de los nombres de los ficheros y directorios que contiene un directorio. Si no se especifica ningún directorio en la línea de la orden **ls**, se obtiene la lista de ficheros del directorio actual, que es aquél en el que se está trabajando.

Sintaxis: ls [-laRrt] [nombre_directorio]|[nombre_fichero]

> ***Nota:*** *Este comando, como la mayoría de comandos Linux, contiene numerosas opciones. Estas opciones se deben de colocar seguidas después del nombre del comando y precedidas por un guión. Se pueden colocar todas las opciones que se quieran, y tanto el guión como la última opción, deben ir separados por al menos un espacio en blanco, del nombre del comando y del resto de parámetros respectivamente.*

Opciones:

1. La opción **-l** (llamada "listado largo") permite obtener información sobre las características de un fichero.

   Al trabajar en un entorno multiusuario, cada usuario puede, pues, crear ficheros y directorios dentro de su espacio de trabajo, del que resulta ser propietario y gestor. Por otro lado, los usuarios pueden pertenecer a un grupo y compartir ficheros con otros miembros del grupo del que forman parte. Un usuario puede, si lo desea, dar autorización a otros usuarios, cualquiera que sea el grupo al que pertenece, para acceder a algunos de sus ficheros y directorios. Todos estos permisos de acceso son gestionados mediante un sistema de autorizaciones de acceso a ficheros y directorios.

   Con el mandato **ls -l** se visualiza en pantalla el propietario y el grupo de cada fichero del directorio al cual se aplica, así como los derechos de acceso asignados a los distintos elementos.

La primera columna de información que visualiza es la que indica el tipo de fichero del que se trata y los derechos asociados a él. El primer carácter indica el tipo de fichero:

- si es **-** entonces es un fichero ordinario.
- si es **d** se trata de un directorio.
- si es **c** significa que es un fichero especial en modo carácter.
- si es **b** entonces es un fichero especial en modo bloque.
- si es **l** se trata de un vínculo (o enlace) simbólico

Los nueve caracteres siguientes indican los derechos de acceso al fichero para las tres categorías de usuarios que maneja Linux (propietario del fichero, miembros del grupo al que pertenece el propietario, resto de usuarios). A continuación muestra un número que corresponde al número de enlaces que contiene el fichero, el propietario del fichero, el grupo al que pertenece, el tamaño del mismo en bytes, la fecha y hora de la última modificación, y por último el nombre del fichero.

Ejemplo:

```
$  ls -l
total  3
-rw-r--r--            2 recaredo     SO      161 2008-01-27  11:52  clases
drwxr-xr-x            2 recaredo     SO      512 2008- 01-27 12:27  esei
-rw-r--r--            1 recaredo     SO      14  2008-01-27 11:35   horarios
```

2. La opción **-a** permite visualizar todos los ficheros incluidos en el directorio, incluyendo los que podemos considerar "ocultos", que son los ficheros que comienzan con un punto. (ej.: el fichero .nuevo no se visualizaría con el comando ls, para verlo habría que poner ls -a).

3. La opción **-R** realiza un listado recursivo del directorio especificado (el actual si no se especifica ninguno) y de todos los subdirectorios que contiene.

4. La opción **–r** realiza un listado de los nombres de los ficheros y directorios en orden alfabético inverso.

5. La opción **–t** ordena el listado de los nombres de los ficheros y directorios por fecha de modificación.

6. La opción **–S** ordena el listado de los nombres de los ficheros y directorios por tamaño.

Nota: *Las distintas opciones de los comando Linux se pueden incluir en una misma línea de comando. p.e., el mandato* ***ls -la*** *visualizando el listado largo de todos los ficheros del directorio (incluyendo los "ocultos").*

Ejemplos:

```
ls  -lt  /etc
ls  -lr  /home
```

El nombre de los ficheros y de los directorios a listar se pueden identificar mediante expresiones de

búsqueda. El comando ls no admite ni comillas dobles ni simples con las que acotar la expresión de búsqueda, pero sí que se pueden usar los siguientes caracteres comodín para crearlas:

a) * El asterisco sustituye a cualquier cadena de caracteres incluyendo la cadena vacía.
b) ? La interrogación sustituye a un sólo carácter en la posición donde aparece. Por ejemplo, ls *.?? listaría todos los ficheros/directorios que tienen dos caracteres como extensión del nombre.
c) [ ] Los corchetes permite definir un conjunto de caracteres, correspondiendo a un sólo carácter en la expresión de búsqueda. Dentro de los corchetes se pueden usar los siguientes caracteres:
    a. El guión "-" para especificar rangos de caracteres. Si se necesita usar el guión como literal en el grupo de caracteres a localizar, se debe poner al principio o al final de dicho grupo.
    b. El circunflejo "^" al principio del conjunto de caracteres para localizar cualquier carácter que no coincida con los especificados por dicho grupo.
    c. Las clases de caracteres predefinidas. La siguiente tabla presenta estas clases de caracteres:

Valor	Significado
[:alnum:]	Cualquier carácter alfanumérico (letras y dígitos): 0-9, A-Z, a-z
[:alpha:]	Cualquier carácter alfabético (letras): A-Z, a-z
[:digit:]	Cualquier carácter numérico (dígitos): 0-9
[:blank:]	Espacios en blanco y tabulador
[:xdigit:]	Cualquier dígito hexadecimal: 0-9, A-F, a-f
[:punct:]	Signos de puntuación: . , " ' ¿ ¡ ; : # $ % & ( ) * + - / < > = @ [ ] \ ^ _ { } | ~
[:print:]	Cualquier carácter imprimible
[:space:]	Cualquier carácter de espacio en blanco: espacios en blanco, tabulador, NL, FF, VT, CR
[:graph:]	Cualquier carácter gráfico excluyendo el espacio en blanco y el tabulador
[:lower:]	Cualquier carácter alfabético (letras) en minúscula: a-z
[:upper:]	Cualquier carácter alfabético (letras) en mayúscula: A-Z
[:cntrl:]	Caracteres de control: NL CR LF TAB VT FF NUL SOH STX EXT EOT ENQ ACK SO SI DEL DC1 DC2 DC3 DC4 NAK SYN ETB CAN EM SUB ESC IS1 IS2 IS3 IS4 DEL.

Estas clases de caracteres deben ir encerradas entre corchetes. Por ejemplo, para localizar las cadenas que contienen dígitos, la expresión regular a usar sería [[:digit:]].

```
Ejemplos:
        ls  /etc/a*
        ls  /home/[aeiou]*
        ls  /home/a[b-d]*
        ls  /home/*[[:digit:]]
```

cp. Copia un fichero

Se utiliza para copiar ficheros. Se puede copiar fichero a fichero, directorio a directorio y lista de ficheros a directorio.

Sintaxis: cp [-irp] (fichero fichero | directorio directorio | lista_ficheros directorio)

Opciones:

1. La opción **-i** nos pide confirmación de copia para cada fichero (**y** o **n**), si el fichero existe en destino.
2. La opción **-r** permite copiar de forma recursiva, es decir si alguno de los ficheros origen es un directorio, copia recursivamente su contenido. El destino debe ser un directorio.
3. La opción -**p** copia ficheros y directorios sin modificar sus permisos ni sus fechas de modificación.

```
Ejemplos:
        Del directorio de trabajo actual cuelga el directorio d1 que contiene ficheros y
subdirectorios con ficheros:
                1. Suponiendo que el directorio d2 no existe:       cp -r   d1  d2
                2. Suponiendo que el directorio d2 existe:          cp -r   d1  d2
        Del directorio de trabajo actual cuelgan los ficheros f1, f2 y f3:
                cp  f1  f2  f3  dir
                cp   /home/ficherosdatos/usuarios  .
```

scp. Copia ficheros de forma segura entre distintos hosts (máquinas)

Se utiliza para copiar ficheros entre distintos host a través de una conexión segura. El comando scp utiliza por defecto el puerto 22 y se conecta mediante una conexión *ssh*.

Sintaxis: scp [-P] [-r] origen destino

Opciones:

1. La opción **-P** nos permite indicarle el puerto a través del cual nos queremos conectar
2. La opción **-r** nos permite copiar un directorio de forma recursiva

```
Ejemplos:
        Del directorio de trabajo actual cuelga el directorio d1 que contiene el fichero f1.txt
y queremos copiarlo al directorio user1/d2 del servidor eixe.esei.uvigo.es, a través del puerto
22:
                scp  -P22   d1/f1.txt   user1@eixe.esei.uvigo.es:/user1/d2/

        Del directorio de trabajo actual cuelga el directorio d1 que contiene los ficheros
f1.txt y f2.txt y queremos copiarlos al directorio user1/d2 del servidor eixe.esei.uvigo.es, a
través del puerto 23:
                1. Recursivamente:
                        scp -P22 -r d1 user1@eixe.esei.uvigo.es:/user1/d2/
                2. Fichero a fichero:
                       scp  -P22  d1/f1.txt    d1/f2.txt   user@eixe.esei.uvigo.es:/user1/d2/
```

rm. Elimina un fichero

Este comando suprime uno o varios ficheros.

Sintaxis: rm [-ir] nombre_fichero1 nombre_fichero2 ...

Opciones:

1. La opción **-i** nos pide confirmación de borrado para cada fichero (**y** o **n**). De todas formas, en muchos sistemas actuales, el sistema pide confirmación de borrado por defecto.
2. La opción **-r** permite borrar de forma recursiva, es decir si el argumento es un directorio, se borra su contenido y recursivamente el de todos los subdirectorios que contenga.

```
Ejemplos:
       rm  -i  f1.txt  dir1/f2.out  ~/f3.exe
        rm   ~/Ejercicios/octubre/*.txt

       Suponiendo que el directorio dir está vacío:              rm dir
                                                                 rm –r dir
       Suponiendo que el directorio dir no está vacío:           rm –r dir
```

Nota: *Dentro de la línea de órdenes Linux permite usar una serie de caracteres que tienen un significado especial en determinados comandos, por lo que se recomienda que no se usen en los nombres de ficheros. Además, se ha de tener en cuenta que el significado especial de estos caracteres pueden variar entre los distintos "shell" o intérpretes de comandos que tiene el sistema. Entre estos caracteres cabe destacar el asterisco (*) que sustituye a cualquier cadena (incluida la vacía).*

mv. Cambia el nombre de un fichero o mueve ficheros

Este comando sirve para mover a los ficheros de directorio o renombrarlos.

Sintaxis: mv [-i] nombre_fichero nuevo_nombre_fichero
mv [-i] lista_ficheros directorio

Opciones:

1. La opción **-i** nos pide confirmación para sobrescribir cualquier fichero o subdirectorio existente.

```
Ejemplos:
       mv  -i  f1  f1
       mv  f1.txt  dir1/f2.out  ~/f3.exe  directrio
```

head. Muestra el comienzo de un fichero

Este comando nos visualiza las primeras líneas o caracteres de un fichero. Si no le especificamos un número de líneas, la opción por defecto son 10.

Sintaxis: head [-número_líneas] [-c número_caracteres] nombre_fichero1 nombre_fichero2 …

Opciones:

1. La opción **–número_líneas** devuelve las primeras **número_líneas** del fichero. Si no le especificamos un número de líneas, la opción por defecto son 10.

2. La opción **–c** visualiza los primeros **número_caracteres** de los ficheros especificados.

```
Ejemplos:
       head   ~/Ejercicios/f*
       head  -5  f1.txt
       head  -c5  f1.txt  dir1/f2.dat  ~/f3.txt
```

tail. Muestra el final de un fichero

Este comando funciona exactamente igual que el anterior, pero con las últimas líneas o caracteres de un fichero.

Sintaxis: tail [-número_líneas][+número_línea] [-c número_caracteres][-f] nombre_fich1 nombre_fich2 …

Opciones:

1. La opción **–número_líneas** devuelve las últimas **número_líneas** del fichero. Si no le especificamos un número de líneas, la opción por defecto son 10.
2. La opción **+número_línea** devuelve desde la línea especificada por **número_línea** hasta el final del fichero.
3. La opción **–c** visualiza los últimos **número_caracteres** de los ficheros especificados.
4. La opción **–f** visualiza los cambios de un fichero en tiempo real.

```
Ejemplos:
       tail   ./f*
       tail  -5  f1.txt
       tail   +2  f1.txt
       tail  -c15  f1.txt  dir1/f2.dat  ~/f3.txt
       tail  –f   f1.txt
```

wc. Visualiza el número de líneas, palabras y bytes de un fichero

El comando wc devuelve el número de líneas, de palabras y de bytes que tiene un fichero (o la entrada dada desde teclado). Las palabras son cadenas de caracteres rodeadas por espacios en blanco.

Sintaxis: wc [-cwlL] nombre_fichero1 nombre_fichero2 …

Opciones:

1. La opción **–c** devuelve el número de bytes que tiene cada fichero.
2. La opción **–w** devuelve el número de palabras que tiene cada fichero.
3. La opción **–l** devuelve el número de líneas que tiene cada fichero.
4. La opción **–L** devuelve el tamaño de la línea más larga contenida en cada fichero.

```
Ejemplos:
       wc   ./f*
       wc  -c  f1.txt
       wc  -w  f1.txt
       wc  -l  f1.txt  dir1/f2.dat  ~/f3.txt
       wc  -L  f1.txt
```