Tutorial BASH n°2
Trabajando con ficheros

INDICE

1. Directorio Personal....3
2. Comandos básicos....4
2.1. Listado del contenido de directorios: comando ls....4
2.2. Creación de subdirectorios. Comando mkdir....5
2.3. Borrado de subdirectorios. Comando rmdir....5
2.4. Cambio de directorio. Comando cd....5
2.5. Directorio actual. Comando pwd....5
2.6. Acceso a unidades de disco....6
2.7. Copia de ficheros. Comando cp....6
2.8. Mover y cambio de nombre de ficheros.(mv)....6
2.9. Enlaces a ficheros. Comando ln....7
2.10. Borrado de ficheros. Comando rm....7
2.11. Características de un fichero. Comando file....8
2.12. Cambio de modo: chmod/chown/chgrp....8
2.13. Búsqueda de ficheros. Comando find....11
3. Espacio ocupado en el disco. Comandos du y df....12
4. Visualizar sin formato un fichero, cat....13
5. Comando head y tail....14
6. Visualizar ficheros con formato. Comando pr....15
7. Visualización de ficheros, more y less....16
8. Comandos de impresión. Comando lpr....17
9. Compresión: Comandos tar y gzip....18
9.1. Ejemplos de uso de tar, gzip y bzip2....18

BASH - Tutorial 2 Trabajando con ficheros

1. Directorio Personal

Cuando un usuario abre la sesión en un sistema Linux, comienza a trabajar en un lugar específico dentro del sistema de ficheros. Ese lugar específico es un propio directorio privado, en el cual, el usuario, puede crear libremente ficheros y subdirectorios. Los directorios propios de otros usuarios no entran en conflicto con el suyo.

Cada usuario tiene un directorio privado. Este directorio tiene el nombre del usuario (**login**). Está dentro del directorio home y lo crea el sistema cuando se da de alta por primera vez al usuario. Es decir, cuando un usuario entra en el sistema comienza dentro de su directorio home. Los usuarios normalmente trabajan dentro de sus directorios propios o en subdirectorios que crean dentro de dichos directorios. No obstante, no importa donde se encuentre el usuario dentro del sistema de ficheros, ya que como se ha visto anteriormente, puede regresar a su directorio propio usando la orden cd sin parámetros. Con la variable del entorno **$HOME** proporcionada por el sistema, se puede ver el nombre de camino del subdirectorio propio. Para acceder al directorio personal basta con ejecutar el comando cd sin argumentos. También se referencia al directorio personal con la variable de entorno **$HOME**, y con el carácter **~**.

Scd

Scd $HOME

$cd~

2. Comandos básicos.

Existen un gran número de comandos para el shell de Linux, aunque se suele utilizar un número limitado de ellos. La ejecución de un comando en el terminal de Linux tiene la siguiente forma:

$comando [opciones] [argumentos]

Al ser un lenguaje sensible a mayúsculas, solo valen comandos escritos en minúsculas. Las opciones que se pueden añadir a un comando, son una o varias letras precedidas de guión (-). Los argumentos pueden ser un nombre de fichero y/o directorios. En muchos comandos no son necesarios ni el argumento ni el nombre del fichero.

Un comando en Linux no es más que un fichero ejecutable del sistema. Son programas que el shell encuentra y ejecuta en respuesta a las instrucciones introducidas por el usuario.

Entre los comandos básicos, podemos citar los siguientes:

cal	Muestra por pantalla el calendario. Si no recibe argumentos, el calendario será el del mes actual, si se ejecuta con argumentos, mostrará el mes y año indicado.
date	Este comando muestra o cambia la fecha y la hora actuales del sistema. Si se proporciona la fecha y la hora como argumentos, permite cambiar la fecha y la hora del sistema (siempre que seamos usuario root). #date MMDDhhmm[yy]
who	Muestra el nombre de los usuarios que se encuentran conectados al sistema en ese momento, indicando el terminal en el que están y desde que hora.
whoami	Indica mi nombre de usuario man comando Muestra el manual de ayuda para el comando pasado por parámetro.
clear	Comando para limpiar la consola

.

A continuación vamos a detallar el funcionamiento de los comandos de más utilizados, junto con sus opciones y argumentos habituales.

2.1. Listado del contenido de directorios: comando ls

Una de las acciones más habituales a la hora de trabajar es mostrar el contenido de un directorio, el shell incluye un programa con este mismo fin: ls

ls	Muestra los nombres de los ficheros y subdirectorios contenidos en el directorio en el que se está. Sólo se obtienen los nombres de los ficheros, sin ninguna otra información.
ls -a	Muestra todos los ficheros incluyendo algunos que ordinariamente están ocultos para el usuario (aquellos que comienzan por un punto).
ls -l	Esta es la opción de lista larga: muestra toda la información de cada fichero incluyendo: protecciones, tamaño y fecha de creación o del último cambio introducido.
ls -c	Muestra ordenando por día y hora de creación.
ls -t	Muestra ordenando por día y hora de modificación.
ls -r	Muestra el directorio y lo ordena en orden inverso.
ls subdir	Muestra el contenido del subdirectorio subdir.
ls -l fichero	Muestra toda la información sobre el fichero.
ls --color	Muestra el contenido del directorio coloreado.

Las opciones anteriores pueden combinarse.

Por ejemplo:

ls -cr Muestra el directorio ordenando inversamente por fechas.

El comando ls admite los caracteres de sustitución o metacarácteres (*****) y (**?**). El **carácter *** representa cualquier **conjunto o secuencia de caracteres**. El **carácter ?** representa **cualquier carácter**, pero sólo uno.

Por ejemplo:

ls *.gif Muestra todos los nombres de ficheros que acaben en .gif, por ejemplo dibl.gif, a.gif, etc.

ls file? Muestra todos los ficheros cuyos nombres empiecen por file y tengan un nombre de cinco caracteres, por ejemplo: file1, file2, filea, etc.

2.2. Creación de subdirectorios. Comando mkdir

El comando **mkdir** (**make directory**) permite a cada usuario crear un nuevo subdirectorio: **mkdir subdirl** donde subdir es el nombre del directorio que se va a crear.

2.3. Borrado de subdirectorios. Comando rmdir

Este comando borra uno o más directorios del sistema (**remove directory**), siempre que estos subdirectorios estén vacíos. Por ejemplo: rmdir subdirl donde subdír es el nombre del directorio que se va a eliminar. Cuando se desee borrar un directorio no vacío se utiliza el comando **rm -r**

2.4. Cambio de directorio. Comando cd

Este comando permite cambiar de directorio a partir del directorio actual de trabajo.

Por ejemplo:

cd /home/jgarcia En este ejemplo pasamos del directorio actual de trabajo al nuevo directorio /home/jgarcia, que será desde ahora nuestro nuevo directorio.

cd directorioA Nos traslada al subdirectorio directorioA (que deberá existir como subdirectorio en el directorio actual).

cd .. Retrocedemos un nivel en la jerarquía de directorios. Por ejemplo, si estamos en /home/jgarcia y usamos este comando, pasaremos al directorio padre, en este caso a /home.

Hay que recordar dos directorios especial, que son el directorio **.** (punto) y el directorio **..** (dos puntos). El directorio . hace referencia al **directorio de trabajo**, es decir el directorio donde nos encontramos en este momento. El directorio .. hace referencia al **directorio padre** del directorio actual, es decir, al que se encuentra en un nivel inmediatamente superior.

2.5. Directorio actual. Comando pwd

El comando **pwd** (**print working directory**) visualiza o imprime la ruta del directorio en el que nos encontramos en este momento. Este comando es uno de los pocos que no tiene opciones y se utiliza escribiendo simplemente pwd.

Tiene asociada la variable de entorno **$PWD**, que al igual que el comando, me muestra el directorio de trabajo actual.

2.6. Acceso a unidades de disco

Linux a diferencia de Windows no utiliza letras ("c:", "d:") para acceder a las distintas unidades de disco de un ordenador. En Linux para acceder al contenido de una unidad de disco o de un CD-ROM este tiene que haber sido previamente "montado". El montado se realiza mediante el comando mount, con lo que el contenido de la unidad se pone a disposición del usuario en el directorio de Linux que se elija. Para acceder al CD-ROM se teclearía el siguiente comando:

mount -t íso9660 /dev/cdrom /mnt/cdrom

donde -t iso9660 indica el **tipo de sistema** que usa la unidad de disco para guardar los ficheros Gas más usuales son: iso9660 en el caso de un CD-ROM, **vfat** en el caso de **Windows**, y **ext3** o e**xt4** en el caso de **Linux**), **/dev/cdrom** indica el dispositivo que se va a montar. Todos los dispositivos están representados por un fichero del directorio /dev, por ejemplo en el caso de un disquete será seguramente /dev/fd0, por último /mnt/cdrom es el directorio en el que se pondrá a disposición del usuario el contenido del CD-ROM.

De todas formas el usuario siempre puede crear un directorio vacío con el nombre que el elija para montar las unidades de disco que desee donde desee.

Cuando el usuario haya dejado de usar ese disco deberá "desmontarlo" mediante el comando umount antes de sacar el disquete o el CD-ROM. En este último caso debería escribir:

umount /mnt/cdrom

Para utilizar el **comando mount** de la forma anterior hace falta ser administrador o **root**. Para que un usuario común pueda utilizar disquetes, CD-ROM, etc. hay que editar el fichero **/etc/fstab**.

2.7. Copia de ficheros. Comando cp

Este comando tiene la siguiente sintaxis:

cp fichero_origen fichero_destíno

y hace una copia de *fichero_origen* y le llama *fichero_destino*. Si *fichero_destino* no existía, lo crea con los mismos atributos de *fichero_origen*. Si *fichero_destino* existía antes, su contenido queda destruido y es sustituido por el de *fichero_origen*. El fichero *fichero_destino* estará en el mismo directorio que *fichero_origen*. Tanto *fichero_origen* como fichero_ destino indican el nombre de un archivo, que puede incluir en la ruta al mismo si alguno de ellos no se encuentra en el directorio actual. Otra posibilidad es:

cp fich_origen fich_destino namedir	que hace copias de fich_origen y fich_destino en el directorio namedir.

2.8. Mover y cambio de nombre de ficheros.(mv)

Este comando tiene una forma similar al anterior.

mv fich_origen destino	El comando mv realiza la misma función que el anterior (cp) pero además destruye el fichero original. En definitiva traslada el contenido de fích_origen a destino. Si destino no es un directorio, a efectos del usuario lo que ha hecho es cambiar el nombre a fich_origen, llamándole destino.
mv fich1 fich2 dir	mueve uno o más ficheros (fích1, flch2,...) al directorio dir conservándoles el nombre.
mv dir1 dir2	cambia el nombre del subdirectorio dir1 a dir2.

2.9. Enlaces a ficheros. Comando ln

En Linux un mismo fichero puede estar repetido con más de un nombre, ya que con el comando cp se pueden realizar cuantas copias se desee del fichero. Pero, a veces, es más práctico tener un mismo fichero con varios nombres distintos, y lo que es más importante, poder acceder a él desde más de un directorio. En Linux esto recibe el nombre de enlaces múltiples a un fichero. El ahorro de espacio de disco es importante al poder compartir un fichero más de un usuario. Estos enlaces son muy prácticos a la hora de utilizar ficheros que pertenecen a directorios distintos. Gracias a los enlaces se puede acceder a muchos ficheros desde un mismo directorio, sin necesidad de copiar en ese directorio todos esos ficheros.

La sintaxis del comando es:

ln [opciones] fichero nombre_enlace

Ejemplo de uso de la orden In serían:

ln fich1 fich2 A partir de este momento el fichero fich1 tiene dos nombres: fich1 y fich2. A diferencia de los comandos **cp** y **mv**, este comando toma más precauciones, ya que advierte previamente si el nombre fich2 está ocupado, y en este caso no se ejecuta.

ln fich1 subdir/fich1 Después de este comando el fichero fich1 tendrá el mismo nombre, pero a efectos del usuario estará colocado en dos sitios distintos: en el directorio actual y en el subdirectorio subdir.

Los ficheros enlazados a otro se borran como los ficheros normales. Si se borra el fichero original permanece su contenido en los ficheros enganchados.

2.10. Borrado de ficheros. Comando rm

Este comando se utiliza para el borrado de ficheros. Tiene siguiente sintaxis:

rm [opciones] fichero...

En los siguientes ejemplos se ilustra el uso más habitual de rm:

rm fich1 fich2 Este comando elimina uno o más ficheros de un directorio en el cual tengamos permiso de escritura.

rm -i fich1 fich2 Con esta opción, Linux pedirá confirmación para borrar cada fichero de la lista, de si realmente se desea su destrucción o no. Se recomienda usar siempre este comando con esta opción para evitar el borrado de ficheros útiles. Por ejemplo, si se teclea,

rm -i fichero aparecerá en pantalla el aviso siguiente:

remove fichero? y habrá que contestar y (yes) o n (not).
En este comando se pueden utilizar los caracteres de sustitución (* y ?).

rm fích* Borra todos los ficheros del directorio actual que comiencen por fich.

rm * Borrará todos los ficheros del directorio actual.

rm -i * Borrará todos los ficheros del directorio actual pero con previa confirmación.

rm -r dir Aunque el comando **rm** se utiliza para borrar ficheros, también es útil para eliminar directorios. Con la opción -r se realiza un borrado recursivo, es decir, que si el parámetro es un directorio, lo borraría junto con su contenido.

2.11. Características de un fichero. Comando file

Este comando realiza una serie de comprobaciones en un fichero para tratar de clasificarlo.

Su sintaxis es:

file fichero Tras su ejecución este comando muestra el tipo del fichero e información al respecto del mismo,

Este comando es muy útil cuando necesitamos buscar información sobre archivos, en función de su tipo (script de lInux, ficheros de texto, código fuente, etc).

2.12. Cambio de modo: chmod/chown/chgrp

En todo sistema Linux los archivos pertenecen a quien los crea, que es entonces el único que puede borrarlos, ejecutarlos, etc. Esto es así porque Unix estaba pensado para ser manejado por muchos usuarios a la vez, y de esta forma nadie podía acceder a tus archivos sin tu consentimiento previo. Para ver los permisos de los ficheros, puedes escribir bien

vdir

ó

ls-la

y veríamos algo así:

[Atributos] [Propietario] [Grupo] [Tamaño] [Fecha creación][Nombre del fichero]

-rw-r--r-- Carlos users 1024 Dic 21 20:30 carta.txt

Que de forma resumida:

-rwxr-xr--carlos users programa.sh

El primer guión me indica que "programa.sh" es un fichero, si fuese un directorio, en vez de guión aparecería un d, y si fuera un enlace una l. Los siguientes tres dígitos **rw-**, indican que el propietario del fichero (usuario carlos), tienen permisos de lectura, escritura y ejecución sobre el archivo. Los siguientes tres dígitos r-w, indica que los usuarios del grupo de carlos pueden leer y ejecutar el archivo, pero no modificarlo. Por último, los tres últimos caracteres r--, indican que el resto de usuarios del sistema puede leer el archivo "**programa.sh**", pero no lo pueden modificar ni ejecutar.

En general los atributos nos muestran la siguiente información:

- El guion del comienzo nos indica si es fichero, directorio o enlace poniendo:
  d (directorio)
  l (enlace)
  - (fichero)

- Los atributos siguientes tomados de tres en tres significan:
  r (read): Permiso de lectura.
  w (write): Permiso de escritura, y
  x (execute): Permiso de ejecución.
  En el caso de que aparezca un **guión -** en vez del atributo, significa que se carece de ese permiso.
  Los tres primeros atributos corresponden al propietario del archivo, los tres siguientes al grupo al que ese usuario pertenezca, y los últimos al resto de usuarios.

Así, ahora utilizando por ejemplo un script que se llamara: "**miscript**" **-rwxr-xr-- carlos users miscript** tenemos que **carlos users** nos dice el dueño del fichero y el grupo del mismo, respectivamente. El usuario

carlos tiene permisos de lectura, escritura y ejecución (para poder borrarlo, por ejemplo) (-rwxr-xr—). El grupo de carlos, users, tiene permisos de lectura y ejecución, (-rwxr-xr--) y el resto de usuarios de sólo lectura (-rwxr-xr--).

El sistema operativo linux/Unix establece diferentes tipos de usuarios. Cada sistema dispone de un **superusuario** con derechos de administración, denominado **root**. Cualquier labor administrativa que se quiera realizar en el sistema, solo podrá ser realizada por este usuario. Para el uso formal" del sistema, se pueden crear un número indeterminado de usuarios. **Cada usuario** dispone de su propio directorio de trabajo **(/home/nombre_usuario**) y de privilegios sobre cada uno de los ficheros y directorios que se encuentren en su carpeta. Para simplificar la labor de administración, los usuarios se agrupan en "grupos" de usuarios. Cada grupo de usuario dispone de privilegios comunes a un conjunto de recurso.

Los permisos de cada fichero se pueden ver con el comando **Is -l**. Para cambiar los permisos de un fichero se emplea el comando **chmod**, que tiene el formato siguiente:

chmod [usuario] oper permiso files

usuario Indica a quien afecta el permiso que se desea cambiar. Es una combinación cualquiera de las letras u para el usuario, g para el grupo del usuario, o para los otros usuarios, y a para todos los anteriores. Si no se da el usuario, el sistema supone a.

oper Indica la operación que se desea hacer con el permiso. Para dar un permiso se pondrá un+, y para quitarlo se pondrá un-.

permiso Indica el permiso que se quiere dar o quitar. Será una

combinación cualquiera de las letras anteriores : r,w,x,s

files Nombres de los ficheros cuyos modos de acceso se quieren cambiar.

Por ejemplo, para quitar el permiso de lectura a los usuarios de un fichero el comando es:

chmod a -r fich Los permisos de lectura, escritura y ejecución tienen un significado diferente cuando se aplican a directorios y no a ficheros normales. En el caso de los directorios el permiso r significa la posibilidad de ver el contenido del directorio con el comando ls; el permiso w da la posibilidad de crear y borrar ficheros en ese directorio, y el permiso x autoriza a buscar y utilizar un fichero concreto.

Cada fichero es propiedad de un determinado usuario, el comando **chown** se emplea para cambiar de propietario ("**ch**ange **own**er") a un determinado conjunto de ficheros. Este comando sólo lo puede emplear el actual propietario de los mismos, o por el **root**. Los nombres de propietario que admite Linux son los nombres de usuario, que están almacenados en el fichero **/etc/passwd**. La forma general del comando **chown** es la siguiente:

chown nuevoDueñofícherol fíchero2...

Análogamente, el grupo al que pertenece un fichero puede ser cambiado con el comando **chgrp**, que tiene una forma general similar a la de **chown**,

chgrp nuevoGrupo fícherol fíchero2...

Los grupos de usuarios están almacenados en el fichero **/etc/group**.

Como hemos indicado, todo fichero en Linux tiene asociado un conjunto de atributos, como son los permisos de acceso, propietario del archivo y grupo del propietario del archivo. Estos atributos se establecen durante la creación del archivo, pero pueden ser modificados. Para cambiar los atributos a un fichero disponemos de tres comandos:

chown	Cambia de dueño al fichero.
chgrp	Cambia de grupo a un fichero.
chmod	Cambia los atributos a un fichero.

Al crear un fichero, por defecto, será del dueño que lo crea, si estoy como javier, y escribo:

gedit carta.txt, la carta creada será de javier, con los atributos propios del mismo, pero si estoy como root, y creo la misma carta, la carta ahora será de root.

Antes de empezar a explicar cómo cambiar los atributos aun fichero, debemos conocer lo siguiente:

1. Sólo el dueño de un fichero, podrá cambiarles sus propiedades, e incluso de dueño.
2. Ningún usuario podrá cambiarle las propiedades a ningún fichero, ni su dueño root, podrá cambiarle las propiedades a TODOS los ficheros, e incluso cambiarles de dueño.
3. Lo de arriba mencionado, no servirá de nada, si al grupo al que pertenece también tiene la propiedad de escribir sobre el fichero, por lo cual, podrá cualquier usuario cambiar el dueño, grupo o propiedades aun fichero.

Ejemplos de modificación de un fichero, y su significado.

chmod u+rwx carta.txt	Donde "**u**", se refiere al **usuario** el cual creó el archivo. chmod g+rx-w carta.txt Donde "**g**", se refiere al **grupo** del usuario. chmod o+r-wx carta.txt Donde "**o**", se refiere al **resto** de usuarios.
chmod a+x miscript	Usando "**a**", modificará todos los atributos. Le pondrá todos los atributos de ejecución al script "**miscript**", para lo ejecuten, el dueño, el grupo y otros.

Para cambiar de dueño a un fichero, se deberá usar el comando chown, o bien utilizar mcf que es más fácil, siempre y cuando tenga los atributos que permitan ser cambiados.

chown -c Javier EsteFichero	Ahora "EsteFichero" pertenecerá a Javier.
chgrp -c users EsteFichero	Cambiamos de grupo al fichero "EsteFichero"
chown -c javier.users fich	Cambiamos el usuario y el grupo.
man chown	Al ejecutar este comando, obtendremos ayuda para chown.

También puede utilizarse **chmod** con números OCTALES, que realizarán las mismas funciones que las letras.

rwx rwx rwx = 111 111 111	rwx = 111 en binario = 7
rw-rw-rw-= 110 110 110	rw-= 110 en binario = 6
rwx ------ = 111 000 000	rx = 101 en binario = 5
y así sucesivamente ...	r - = 100 en binario = 4

Por ejemplo se desea que todos las personas puedan ver escribir, leer y ejecutar sobre el archivo creditos.tex, entonces hacemos:

 # chmod a+wrx creditos.tex o su equivalente en números

 # chmod 666 credítos.tex

2.13. Búsqueda de ficheros. Comando find

El comando find se utiliza para buscar el directorio donde se encuentra un archivo. Un ejemplo de uso es:

find / -name proyecto -print

A continuación del nombre de la orden hay que especificar el directorio desde dónde tenemos que empezar a buscar (en el ejemplo "/" directorio raíz), después de name seguido del nombre del fichero y, por último, -print que indica que visualice el camino.

Además del argumento -name existen otros:

-user nombre_de_usuario	que se refiere a los ficheros propiedad de nombre de usuario.
-group nombre_de_grupo	Que selecciona los ficheros que pertenecen al grupo nombre_d e_grupo.
-mtime número_de_días	Hace referencia a los ficheros que han sido modificados hace número_de_días.
-size bloques_tamaño	Visualiza ficheros de tamaño bloques_tamaño.

Tanto al número bloques_tamaño como número_de_días podemos anteponerles el carácter - o + que indicará que selecciónalos ficheros mayores o menores a ese número. Por ejemplo:

find / -size +100 -print	buscaría los ficheros de tamaño superior a 100 bloques.

3. Espacio ocupado en el disco. Comandos du y df

El comando du permite conocer el espacio ocupado en el disco por un determinado directorio y todos los subdirectorios que cuelgan de él.

La sintaxis de este comando es:

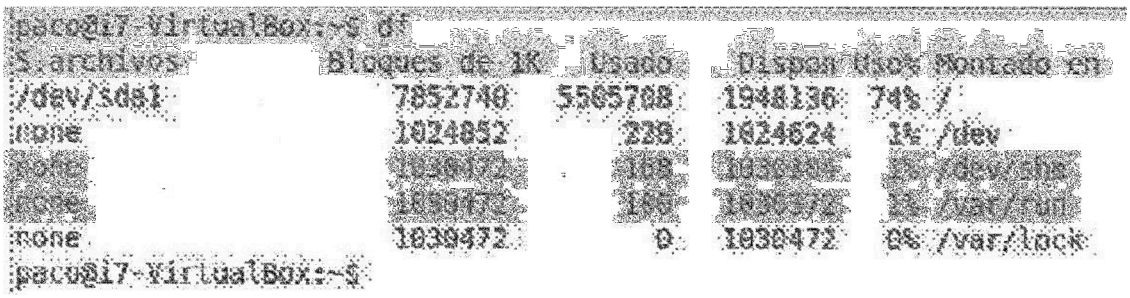
du [opciones] ... [fichero]...

Los usos más habituales del comando du son:

du	Al ejecutar el comando du sin parámetros, nos muestra el espacio de disco utilizado por el directorio donde nos encontramos, dado en número de bloques.
du –h	Al ejecutar du con la opción —h nos muestra el espacio en disco que ocupa el directorio, pero en número de bytes.
du <directorío>	Nos muestra el número de bloques lógicos de 1 KB que ocupa el directorio indicado
du -a	Nos muestra el número de bloques lógicos de 1 KB que ocupan los ficheros contenidos en el directorio actual, o el directorio que se pase como argumento.
du -s	obtenemos un resumen sin que aparezcan los detalles

El comando **df** por el contrario informa del espacio total, ocupado y disponible para cada uno de los sistemas de ficheros. La sintaxis de este comando es:

df [opción] ... [fichero] ...



4. Visualizar sin formato un fichero, cat

Este comando permite visualizar el contenido de uno o más ficheros de forma no formateada.

También permite copiar uno o más ficheros como apéndice de otro ya existente. Algunas formas de utilizar este comando son las siguientes,

cat fichero	Saca por pantalla el contenido del fichero fichero.
cat file1 file2...	Saca por pantalla, secuencialmente y según el orden especificado, el contenido de los ficheros indicados.
cat file1 file2 >file3	El contenido de los ficheros fíle1 y file2 es almacenado en file3.
cat filel fíle2 »file3	El contenido de file1 y file2 es añadido al final de file3.
cat >filel	Acepta lo que se introduce por el teclado y lo almacena en fíle1 (se crea file1). Para terminar se emplea <ctrl>d
cat -n fichero	permite numerar las líneas

5. Comando head y tail

El comando head muestra las primeras líneas de un archivo. Por omisión muestra las **10 primera líneas**, pero el número se puede modificar agregando la opción -n X, donde X es el número de líneas que se desean mostrar. La sintaxis del comando head es la siguiente:

head [opciones] [archivo]

Un ejemplo de uso es:

head -7 fichero escribe las 7 primeras líneas del fichero filename

El comando **tail** sirve para mostrar las últimas líneas de un fichero. Por omisión muestra las **últimas 10 líneas** del archivo, pero este número se puede modificar de igual forma que en head, agregando la opción -nX donde X es el número de líneas que se quieren mostrar. La sintaxis del comando **tail** es la siguiente:

tail [opciones] [archivo]

Un ejemplo de uso sería:

tail -4 fichero escribe las 4 últimas líneas de fichero.

6. Visualizar ficheros con formato. Comando pr

Este comando, a diferencia de cat, imprime por consola el contenido de los ficheros de una manera formateada, por columnas, controlando el tamaño de página y poniendo cabeceras al comienzo de las mismas. Está muy en relación con el comando lp de salida por impresora. Las formas más importantes que admite son las siguientes:

pr file	Produce una salida estándar de 66 líneas por página, con un encabezamiento de 5 líneas (2 en blanco, una de identificación y otras 2 líneas en blanco).
pr -ln file	Produce una salida de n líneas por página (cuando el tamaño de papel de impresora, por ejemplo, tiene un número de líneas distinto de 66)
pr -p file	Hace una pausa para presentar la página, hasta que se pulsa <return> para continuar
pr -t file	Suprime las 5 líneas del encabezamiento y las del final de página.
pr -wn file	Ajusta la anchura de la línea a n posiciones. pr -d file Lista el fichero con espaciado doble.
pr -h `caracteres`file	el argumento o cadena de caracteres `caracteres` se convertirán en la cabecera del listado.
pr +n file	Imprime el fichero a partir de la página n.

Además de los ejemplos anteriores, se pueden combinar varias opciones en un mismo comando, como por ejemplo en: **pr -dt file** la salida de este comando es por la consola, pero puede redireccionarse a otro fichero, por ejemplo, si ejecutamos el comando: **pr file1 > file2** se crea un fichero nuevo llamado file2 que es idéntico a file1, pero con formato por páginas y columnas.

7. Visualización de ficheros, more y less

Estos comandos permiten visualizar un fichero pantalla a pantalla. El **número de líneas por pantalla** es de 23 líneas de texto y una última línea de mensajes, donde aparecerá la palabra **more**. Cuando se pulsa la barra espaciadora (el espacio en blanco), se visualizará la siguiente pantalla. Para salir de este comando (terminar la visualización) se pulsa **<ctrl>d** o **q.**

Por ejemplo:

more fichero

El comando **less** es muy similar al anterior pero permite el desplazamiento a lo largo del texto empleando las teclas de cursores pudiendo desplazarse hacia arriba o abajo de un fichero.

Opciones en el uso de **more/less**:

h	se obtiene la ayuda
q	se sale del comando
xG	se posiciona en la línea "x"
G	se posiciona en la última línea
/cadena y la tecla de retorno de carro	se posiciona en "cadena".
n	busca la siguiente cadena
N	busca la anterior cadena

8. Comandos de impresión. Comando lpr

El comando **lpr** se emplea para imprimir una serie de ficheros. Si se emplea sin argumentos imprime el texto que se introduzca a continuación en la impresora por defecto. Por el contrario,

lpr nombre_fichero imprime en la impresora por defecto el fichero indicado.

9. Compresión: Comandos tar y gzip

Tanto el comando **tar** como **gzip** son ampliamente empleados para la difusión de programas y ficheros en Linux. El primero de ellos agrupa varios ficheros en uno solo archivo, mientras que el segundo los comprime.

El uso general del comando **tar** para crear el archivo de empaquetado es:

tar-cvf nombre_archivo.tar fichero1 fichero2 ...

donde fichero1, fichero2 etc. son los ficheros que se van a añadir al archivo tar.

Si se desea extraer los ficheros se emplea

tar-xpvf nombre_archivo.tar fichero1...

El uso de **tar no comprime** el fichero resultante, con lo que no se ahorra espacio. Si queremos aplicar compresión, debemos asociar un compresor a tar.

Para comprimir se utiliza el comando **gzip**, que **comprime un único fichero**, con lo que la información se mantiene pero se reduce el tamaño del mismo. El uso del comando gzip es el siguiente:

gzip fichero	con lo que se comprime fichero (que es borrado) y se crea un fichero con nombre fichero**.gz**.

Si lo que se desea es descomprimir un fichero emplea:

gzip -d fíchero.gz	descomprimiendo el archivo y recuperando el archivo inicial.

Se puede utilizar gzip a continuación de tar, obteniendo ficheros con extensión tar.gz o tgz que contienen varios ficheros de forma comprimida (similar a un fichero zip). El comando tar incluye la opción z para estos ficheros de forma que para extraer los ficheros que contiene:

tar -zxf fichero.tar.gz	Extrae los fichero empaquetados con tar y comprimidos con gzip.

9.1. Ejemplos de uso de tar, gzip y bzip2.

Aquí vamos a tratar con ejemplos los distintos modos o los más usados que son utilizados para comprimir archivos o directorios.

Los compresores tienen distintos niveles de compresión siendo proporcional el nivel de compresión con el tiempo que se tarda en descomprimir el fichero, es decir a más compresión, más tiempo para descomprimirlo.

tar -cvf copía_de_seguridad.tar /home/usuario/documentos/*.*

c = compress (empaquetar)

v = verbose (muestra las acciones del compresor)

f = file (empaquetar en un fichero)

Este ejemplo lo que haría sería concentrar o juntar todos los archivos existentes en /home/usuario/documentos/ y meterlos en un solo archivo, copia_de_seguridad.tar .

tar -cvfz copia_de_seguridad.tar.gz /home/usuario/documentos/*.*

Este ejemplo lo que haría sería concentrar o juntar todos los archivos existentes en /home/usuario/documentos/ y meterlos en un solo archivo, copia_de_seguridad.tar y a continuación lo comprime con el compresor Gzip (.gz).

tar -tzvf archivo.tar.gz Mostraría el contenido del fichero comprimido. gzip nombre_del_archívo Comprime un archivo, dándole la extensión gz.

Para conseguir una compresión más potente que la obtenido con Gzip. Se emplea el compresor Bzip2. Este compresor es algo más potente que Gzip y se usa bastante junto con Tar.

tar cvfj copia.tbz2 archivo_a_comprimír Empaqueta el "archivo_a_comprimir" y le aplica el compresor Bzip2

tar -je directorio/ >directorio.tar.bz2 Empaqueta y comprime con tar y bzip2

Para desempaquetar un archivo tar, y para poder descomprimirlo, se utiliza el comando tar y gzip con las siguientes opciones.

tar -xvf archivo.tar Desempaqueta el archivo.tar. **x = extract** (desempaquetar),

gzip-de archivo.tar.gz | tar-xvf - Descomprime un fichero comprimido con gzip.

tar -xvzf archivo.tar.gz Descomprime el archivo.tar.gz.

Si la compresión se ha realizado con Bzip2, las opciones que utilizan con tar son las siguientes:

tar xvfj archivo.tbz2 Descomprime el archivo archivo.tbz2 tar tvfj archivo.tbz2 Muestra el contenido del archivo comprimido.