

Comandos

Para configurar el sistema postinstalación: **bsdconfig** y **bsdinstall**

pwd: muestra el directorio y la ruta en que se encuentra actualmente.

cd: permite moverse entre directorios, subiendo o bajando niveles.

exit o logout: para cerrar una sesión.

ee: permite la edición de los ficheros.

cat: muestra lo que contiene un fichero. Se carga en memoria.

more: muestra las 25 líneas siguientes.

date: muestra la fecha y la hora del sistema. Existen variaciones que permiten personalizar el formato en que se muestra y se basan en variables del sistema:

%m: mes del año, 01 al 12.

%d: día del mes.

%y: año , del 00 al 99.

%D: para formato de fecha.

%T: para formato de hora.

%H: horas, de 00 a 23.

%M: minutos, de 00 a 59.

%S: segundos , de 00 a 59.

%j: número de día, del año.

%r: notación AM o PM.

Ej: date "+La fecha de hoy es: %d del mes %m del año %y"

Para ajustar la fecha del sistema, se utiliza el siguiente comando:

date YYMMDDHHmm.seg

who: muestra quienes están conectados, en qué ttys y la hora desde que iniciaron sesión. Los parámetros más utilizados son:

who -H: muestra el nombre de las columnas.

who -m: muestra quién está conectado en la terminal donde se ejecuta.

who -q: muestra la cantidad de usuarios que están conectados y los nombres.

Whoami: se utiliza para verificar información del usuario conectado. Desde la creación de ID, se encuentra obsoleto.

Logname: despliega el nombre del usuario conectado.

Passwd: permite cambiar el password del usuario. En caso de ser root, puede modificar el password de cualquier usuario.

Last: muestra los últimos usuarios que se conectaron al sistema. Contiene las siguientes opciones:

-d: muestra los usuarios que se han logeado durante el día.

-n: muestra los últimos n usuarios que se han logeado.

Id: se utiliza para conocer información del usuario conectado. Muestra los nombres y los identificadores del usuario y de los grupos. Contiene las siguientes opciones:

-c: muestra solamente el nombre del usuario logeado.

-g: muestra solamente el ID del grupo del usuario.

-u: muestra solamente el ID del usuario.

-p: muestra la información en columnas

File: determina al tipo de archivo que es, si es un fichero, un directorio, un link, etc. Contiene las siguientes opciones:

-b: muestra únicamente el tipo de archivo, no especifica el nombre.

History: lleva un registro de los comandos que ha escrito el usuario en la Shell.

uptime: muestra la hora actual del sistema, cuanto tiempo lleva el sistema funcionando, el número de usuarios conectados y estadísticas del sistema.

echo: imprime en pantalla.

ls: lista el contenido de un directorio. Los parámetros más utilizados son:

ls -a: lista todos los archivos, incluidos los . y ..

ls -l: lista el formato detallado de los ficheros y directorios.

ls -s: lista el tamaño de los ficheros.

ls -t: muestra los archivos ordenados por la fecha de la última modificación.

ls -R: lista recursivamente todos los ficheros y directorios que contiene un directorio.

ls -c: lista la información en columnas.

ls -G: muestra la información marcando con color los directorios

mkdir: permite crear directorios. Posee el **mkdir -p**, para crear la ruta completa.

rmdir: comando para eliminar directorios, siempre y cuando estén vacíos.

rm: comando para eliminar ficheros. Los parámetros más utilizados son:

rm -r: borrado recursivo. Utilizado para eliminar directorios que no están vacíos.

rm -i: Pregunta si está seguro de eliminar el fichero.

touch: comando para crear ficheros. Solamente creación, no edición.

wc (Word count): comando que permite obtener información acerca de los ficheros. Devuelve 3 valores: líneas, palabras, bytes del fichero.

cp: copia ficheros, en la ubicación especificada. Se puede conservar el mismo nombre o asignársele uno diferente a la hora de copiarlo. También se pueden copiar varios archivos a la vez a una misma ruta.

mv: mueve un fichero a la ubicación especificada. Se puede conservar el mismo nombre del fichero o cambiárselo por uno nuevo. También se pueden mover varios archivos a la vez a una misma ruta.

Cat:

grep: comando utilizado para buscar palabras dentro de un fichero. Se pueden realizar búsquedas en varios ficheros a la vez. Los parámetros más utilizados son:

grep -c: imprime la cantidad de líneas que contienen la palabra seleccionada.

grep -H: muestra el nombre del fichero que contiene la palabra.

grep -h: esconde el nombre del fichero que contiene la palabra.

grep -i: no diferencia entre mayúsculas y minúsculas.

pipe (|): permite la ejecución de un comando, utilizando la salida de otro comando. Ej: ls | more

tail: este comando se utiliza para extraer las últimas líneas de un fichero. Por defecto son las últimas 25 líneas, pero uno puede modificar esa cantidad.

Ej: tail -n 4 /etc/passwd o bien tail -4 /etc/passwd

Cut: comando que permite extraer líneas de un fichero. Utiliza varios comodines, como por -d para establecer el delimitador que deseamos tomar en cuenta o -f para indicarle la cantidad de filas que deseamos que extraiga o -c para extraer caracteres.

Por ejemplo, para sacar la primera y la tercera columna de un fichero, estableciendo el espacio en blanco como delimitador, se utiliza el siguiente comando:

cut -d " " -f 1,3 <fichero>

Para extraer los primeros 10 caracteres de todas las líneas, se utiliza:

Cut -c 1-10 <fichero>

Sort: permite un el contenido de un fichero, antes de mostrarlo en pantalla. La salida estándar es un ordenamiento de menor a mayor. Si se desea ordenar de mayor a menor, se utiliza el -r. Si se desea que no tome en cuenta líneas repetidas, se utiliza -u. Si se desea las mayúsculas de las minúsculas, se utiliza -f. Si se desea que lo ordene, con base a valores numéricos, se utiliza -n. Si se desea que lo ordene de forma aleatoria, se utiliza -R.

Metacaracteres

Se utilizan para realizar búsquedas, con el fin de abreviar las búsquedas o encontrar ficheros aplicándoles filtros.

?: cuando se utiliza este signo, sustituye a un solo carácter. Se pueden usar varios signos de pregunta consecutivos, pero la cantidad debe coincidir exactamente con la cantidad de caracteres del fichero que se está buscando. Ls

- Ej: ls otr? | ls otro????

*****: se utiliza para buscar ficheros que inicien con determinados caracteres y el resto no se conocen o no interesan.

- Ej: ls otr* | ls o* | ls ot*

[]: se utilizan para hacer variantes en el nombre de un fichero, las cuales pueden contener varios posibilidades, filtros o restricciones.

- Ej: `ls [abc]*`, buscará todos los archivos que comiencen con a, b o c sin importar la cantidad de caracteres que contenga el nombre del fichero.
- También se pueden usar rangos. Ej: `ls [a-e]*`
- `ls o[tr]???`: busca los ficheros que empiezan con o, el siguiente carácter es **t** o **r** y el nombre del fichero está compuesto por 5 caracteres.

^: utilizado para negar la expresión. O sea, que NO contenga los caracteres especificados.

Copia de archivos.

```
cp nota nota.copia
```

copia el archivo `nota` en `nota.copia`. Si el directorio propio del usuario es `/home/usuario1`,

```
cp ./nota /home/usuario1/nota.copia
```

hace lo mismo indicando las rutas. Si el archivo destino existe lo sobrescribe y realiza la copia sin comentario ni advertencia.

```
mkdir dir1
```

```
cp nota dir1
```

si el destino es un directorio, el archivo se copia en ese directorio.

```
cp nota LEAME dir1
```

pueden copiarse varios archivos hacia un directorio.

```
mkdir dir2
```

```
touch dir2/arch2 dir2/arch3
```

```
ls dir2
```

```
cp dir2/* .
```

copia todos los archivos en `dir2` hacia el directorio actual.

Para copiar un directorio es preciso usar la opción `-r`, de "recursivo":

```
cp -r dir1 dir3
```

```
ls -l dir3
```

copia el directorio `dir1` y todo el contenido en el `dir3`.

```
rmdir dir3
```

fracasa porque `dir3` no está vacío.

```
rm -r dir3
```

con la opción `-r`, "recursivo", se puede borrar completamente un directorio y todo su contenido, sean estos archivos o subdirectorios.

```
rm -r dir1/* dir2
```

borra todos los archivos bajo `dir1` y `dir2`; borra también el directorio `dir2` pero no el `dir1`.

```
rm arch? nota.copia
```

borra del directorio actual los archivos `arch1`, `arch2`, ..., y `nota.copia`.

Mover, cambiar de nombre.

```
echo "Hola todos" > hola
cat hola
mv hola saludo
cat saludo
```

`mv` cambia el nombre del archivo `hola` por el de `saludo`.

```
mv saludo dir1/saludo.1
```

mueve hacia otro directorio cambiando el nombre.

```
mv dir1 dirnuevo
```

cambia de nombre archivos o directorios.

```
rm -r dirnuevo
```

elimina `dirnuevo` y todos sus archivos.

Enlaces "hard".

Un enlace "hard" (hard link) es una nueva referencia a un mismo archivo; consiste en una nueva entrada de directorio que apunta a un archivo ya existente y apuntado desde otra entrada de directorio. El contenido se encuentra en un solo lado, pero el archivo puede ser accedido desde varios lugares, eventualmente con diferentes nombres.

```
echo 'Hola todos de nuevo!' > adios
ln adios adios.ln0
```

`ln` crea un segundo nombre referido al mismo archivo; no se hace copia, crea un enlace (link). Verificar que

```
ls -l adios*
```

muestra un conteo de 2 en la columna de enlaces. Al crear el enlace con el comando `ln`, observar que el archivo existente va en primer lugar, luego uno o más nombres de enlaces a crear.

```
ln adios adios.ln1
ln adios adios.ln2
ls -l adios*
```

muestra el conteo de enlaces hard en estos archivos.

```
mkdir dir2
ln adios dir2/adios
```

hace aparecer en `dir2` un enlace llamado `adios` hacia el archivo `adios` del directorio actual. Un enlace hard puede hacer aparecer un mismo archivo en más de un directorio, con igual o distinto nombre; el contenido está en un solo lado, no está repetido.

```
rm adios*
ls -l dir2/adios
```

Cuando se borra un archivo con varios enlaces, sólo se borra el enlace. El archivo es realmente borrado sólo cuando desaparece el último enlace.

```
ln dir2 dir2ln
```

da error; no se puede crear un enlace hard para un directorio.

Enlaces simbólicos.

Un enlace simbólico (symbolic link) es un archivo que apunta a otro archivo o directorio. El archivo de enlace simbólico contiene solamente la ruta del otro archivo o directorio.

```
ln -s nota nota.ls0
```

crea `nota.ls0` como enlace simbólico.

```
ln -s dir2 dir2ls
```

crea `dir2ls` como enlace simbólico hacia un directorio.

```
ls -l
```

muestra la letra `l` en el tipo de archivo para indicar que es un enlace simbólico.

```
ls dir2
```

```
ls dir2ls
```

muestran el mismo contenido de archivos.

```
ls -l dir2ls
```

muestra que es un enlace simbólico e indica hacia donde apunta.

```
cd dir2ls
```

```
ls -l
```

muestra el contenido de `dir2`, enlazado desde `dir2ls`.

```
pwd
```

indica el directorio con nombre del enlace, pero

```
/bin/pwd
```

muestra el directorio verdadero, no el enlace simbólico; `pwd` indica la ruta por la que se llegó al directorio, `/bin/pwd` indica la ruta de acceso al verdadero directorio. Conviene usar `/bin/pwd` para evitar ser engañado por enlaces simbólicos a directorios. El comando `pwd` es interno del shell, `/bin/pwd` es un programa aparte.

```
cd ..
```

```
mv dir2 DIR2
```

cambia el nombre del directorio real; el enlace simbólico sigue apuntando a `dir2`, por lo que

```
cd dir2ls
```

da error, ya que no existe el objeto real.

```
mv DIR2 dir2
```

repone el nombre original al que apunta el enlace.

```
rmdir dir2ls
```

da error porque no es un directorio sino un enlace.

```
rm dir2ls
```

borra el enlace. Un enlace simbólico se borra con `rm`, y no con `rmdir`, puesto que el enlace simbólico es un archivo y no un directorio, aún cuando sea un enlace simbólico hacia un directorio.

```
ls dir2
```

cundo se borra un enlace simbólico, sólo se borra un archivo puntero, y no el objeto real.

Opciones del comando `ls`.

```
ls -l
```

muestra un ítem por línea. Cuando `ls` se usa en interconexiones, la salida de `ls` presenta un ítem por línea por defecto.

```
ls -C
```

fuerza el despliegue encolumnado.

```
ls -a
```

muestra todos los archivos, incluyendo los que comienzan con punto, normalmente no mostrados. El directorio corriente `.` y el directorio padre `..` son entidades reales.

```
ls -F /bin
```

```
ls -F
```

agrega sufijo que indica el tipo de archivo: `/` directorio, `*` ejecutable, `@` enlace simbólico.

```
ls -R
```

muestra recursivamente todos los archivos, subdirectorios y archivos dentro de subdirectorios. Para ver un listado más largo,

```
ls -CR /etc | more
```

```
ln -s dir2 dir2ls
```

```
ln nota nota.ln0
```

```
ls -l
```

listado largo, muestra atributos de los archivos: tipo de archivo (primer carácter), permisos (9 caracteres), enlaces hard (en archivos), dueño, grupo, tamaño en bytes, fecha y hora, nombre.

```
-rw-r--r-- 1 esteban other 138 Apr 5 19:34 LEAME
drwxr-xr-x 2 esteban other 138 Apr 5 19:34 dir2
lrw-r--r-- 1 esteban other 138 Apr 5 19:34 dir2ls ->./dir2
-rw-rw-rw- 1 esteban other 138 Apr 5 19:34 nota
```

El símbolo `->` indica "apunta hacia" para enlaces simbólicos.

Cambio de dueño y grupo.

```
chown usuariol nota
```

cambia el dueño del archivo `nota` adjudicándoselo a `usuariol`.

```
chown usuariol arch1 arch2
```

cambia el dueño de la lista de archivos indicada. No puede revertirse, ya que el usuario actual dejó de ser dueño de esos archivos. Si el archivo tiene permiso de lectura, es posible copiarlo para disponer de una copia propia. El cambio de dueño no está permitido a usuarios comunes, sino reservado al supervisor para evitar que a un usuario se le adjudiquen archivos sin su consentimiento.

```
chgrp tecnicos nota
```

cambia de grupo el archivo `notas`, adjudicándoselo al grupo `tecnicos`. El dueño debe pertenecer al grupo `tecnicos`.

```
chgrp tecnicos arch1 arch2
```

cambia el grupo de la lista de archivos.

Modos de permisos.

Los permisos de archivos y directorios se cambian con el comando `chmod`. Pueden expresarse de dos maneras: simbólica y absoluta.
En la forma simbólica, se usa la siguiente sintaxis:

```
[ugoa...][[+|=][rwxstugo...]
```

```
+  agrega permiso a los existentes
-  quita permiso de los existentes
=  únicos permisos asignados al archivo
r  lectura
w  escritura
x  ejecución, o acceso si es directorio
s  usa el id del dueño o grupo del archivo al ejecutar
t  fijar "sticky bit" a un directorio: sólo dueños pueden borrar
u  usuario (dueño)
g  grupo
o  otros
a  todos (dueño, grupo y otros)
```

Ejemplos de permisos en notación simbólica:

```
u+r      g+w-r      ug+r      ugo+w-rx      u+rwx,g+r-wx,o-rwx
```

```
chmod u+rwx,g+rw-x,o+r-wx arch1
```

```
chmod u=rwx,g=rw,o=r arch1
```

cambian los permisos de los archivos indicados.

En modo absoluto se usa un código de 4 dígitos octales 0..7, en la forma Nnnn.

N, primer dígito:

```
4  fijar ID del dueño al ejecutar
2  fijar ID del grupo al ejecutar
1  fijar "sticky bit" a un directorio
```

nnn, 3 dígitos: permisos del usuario, grupo y otros, con el código

```
4  lectura
```

```
2  escritura
```

```
1  ejecución, o acceso si es directorio
```

Ejemplos de permisos en notación absoluta:

```
0777      0755      0764      0640      0710
```

```
chmod 0764 arch1
```

cambia los permisos como en el ejemplo anterior.

Listado de permisos de los directorios.

```
ls -l
```

muestra los permisos y otros atributos.

Para obtener información del directorio `dir1` ubicado en el directorio actual,

```
ls -l
```

lista todos los archivos, entre los que aparece `dir1`;

```
ls -l dir1
```

lista el contenido del directorio `dir1`.

```
ls -ld dir1
```

trata la entrada `dir1` como archivo, sin listar su contenido, y dando sus atributos como directorio.

```
ls -ld .
```

muestra atributos del directorio actual.

Cambio de permisos.

```
chmod u-w nota
```

quita permisos de escritura al dueño.

```
chmod u+w nota
```

concede permisos de escritura al dueño.

```
chmod u-w+x nota
```

```
chmod u-wx+r nota
```

cambian varios permisos del dueño al mismo tiempo.

```
chmod u+w nota
```

```
chmod g+w nota
```

```
chmod o+w nota
```

concede permiso de escritura solo al usuario, al grupo o a otros.

```
chmod u+rw-x,g+r-wx,o+r-wx nota
```

fija todos los permisos en una sola operación. Los permisos que no se mencionen quedan como estén.

```
chmod 0644 nota
```

realiza lo mismo en notación absoluta.

El comando `chmod` admite las opciones

`-v` verboso, describe los permisos cambiados

`-f` no da error sobre permisos que no pueden cambiarse

`-R` recursivo, cambia permisos de directorios y sus archivos

```
chmod -v u+rw,g+rw-x,o+r-wx arch1
```

```
chmod -v 764 arch1
```

```
chmod -v 444 arch1
```

```
chmod u=rw,go=r arch1
```

fija permisos en `rw-r--r--`.

```
chmod u=rwx,u+s arch1
```

ejecutará `arch1` con permiso del dueño de `arch1`.

```
chmod -vR a+r dir1
```

da permiso de lectura a todos los directorios bajo `dir1`, anunciando en forma verbosa lo hecho.

```
chmod ugo+rwx dirtodos
```

```
chmod a+t dirtodos
```

fija "sticky bit": a pesar de tener el directorio permisos totales para todos, sólo pueden borrarse los archivos propios del usuario, no los ajenos.

```
chmod 1777 dirtodos
```

realiza lo mismo.

Los permisos `s` (`setuid`, `setgid`) hacen que un archivo ejecutable ejecute con permisos del usuario dueño o del grupo dueño del archivo, cualquiera sea el usuario y grupo de quien lo invoca. El permiso `setgid` sobre un directorio hace que los archivos y subdirectorios creados en él pertenezcan al grupo del directorio, y no al grupo primario del usuario, como es normal; esto es útil para los grupos de trabajo. El permiso `t` (sticky bit) se aplica a directorios con permisos totales; limita la operación de borrado al dueño del archivo o subdirectorio. Los modos `S` y `T`, que pueden aparecer en `ls -l`, indican modos sin sentido: `setuid` o `setgid` sobre archivos no ejecutables, sticky bit sobre directorio sin permisos para todos.

basename, dirname

```
echo $EDITOR
```

muestra la variable de ambiente que contiene el nombre del editor por defecto. Si no aparece nada, inicializarla:

```
which vi
```

para obtener la vía hacia el editor `vi`.

```
EDITOR=/usr/bin/vi
```

```
echo $EDITOR
```

muestra el nombre del editor a usar por defecto.

```
MIEDITOR=`basename $EDITOR`  
echo "Mi editor es $MIEDITOR"
```

muestra el nombre del archivo sin ruta; `basename` quita la ruta a un nombre de archivo calificado con ruta.

```
DIREEDITOR=`dirname $EDITOR`
```

`dirname` separa la ruta del nombre completo.

```
echo "Mi editor es $MIEDITOR en el directorio $DIREEDITOR"
```

muestra el uso en comandos de variables de ambiente.

```
echo "Mi editor es "`basename $EDITOR`" en "`dirname $EDITOR`"
```

muestra el uso del operador grave para ejecutar comandos dentro de otros comandos.

Archivos de dispositivos.

El sistema de archivos de UNIX provee una interface estándar entre los dispositivos de hardware y el sistema operativo que actúa igual que un archivo común: todas las operaciones de entrada y salida se hacen escribiendo y leyendo sobre un archivo. No se trata de un archivo común en disco; actúa como una ruta hacia un canal de entrada/salida del hardware. Así como se redirige la salida de un comando hacia un archivo, puede redirigirse también hacia un disco, una cinta o un módem con sólo mencionar el nombre del dispositivo correspondiente.

```
tty
```

devuelve el nombre del dispositivo asociado al terminal, por ejemplo `/dev/tty1`. Tomando ese mismo nombre

```
ls -l /dev/tty1
```

obtenemos un listado de atributos del "archivo" controlador de este dispositivo, algo así como

```
crw-rw-rw- 1 usuario1 otros .... /dev/tty1
```

El primer caracter indica el tipo de dispositivo: c si es orientado a caracteres, b si es orientado a bloques. Terminales, impresoras y modems son orientados a caracter, discos y cintas son orientados a bloque.

```
cat - < /dev/tty1
```

para escuchar en la propia sesión de terminal; escribir varias líneas y finalizar con Ctrl-D.

```
cat - < `tty`
```

produce el mismo efecto. Luego probar

```
$ cat - > `tty`
```

¿puede explicarse el comportamiento?

El comando mesg.

```
ls -l `tty`
```

Si otros usuarios pueden leer y escribir en este dispositivo, también pueden ver lo que se está haciendo o aún escribir directamente en el terminal redirigiendo su salida. Esto se usa en los comandos `wall` (write all, escribir a todos) y `write` (escribir mensajes entre usuarios).

```
mesg
```

muestra si está habilitada o no la recepción de mensajes.

```
mesg y
```

cambia permisos de grupo y otros en el dispositivo, aceptando mensajes.

```
mesg n
```

rechaza mensajes. Verificar con

```
ls -l `tty`
```

que muestra los permisos que regulan la terminal del usuario.

```
ls -l /dev/tty?
```

muestra los permisos de las terminales principales;

```
ls -l /dev/tty??
```

muestra una cantidad de tipos de terminal disponibles.

Otros dispositivos.

```
cat /etc/passwd
```

```
cat /etc/passwd > /dev/null
```

El dispositivo `/dev/null` actúa como una papelera de tamaño ilimitado que recibe la salida a descartar. No guarda nada; lo que allí se envía se pierde para siempre.

Todos los dispositivos de hardware del sistema tienen una representación en el directorio `/dev`.