CONTENIDO

COMANDOS BÁSICOS DE GNU-LINUX. *4*

Introducción. **4**

Formas de obtener ayuda en linux. **6**

Usuarios y grupos. **7**

Fichero de usuarios: /etc/passwd 9

El fichero de contraseñas /etc/shadow. 10

El fichero de grupos /etc/group. 11

El fichero de contraseñas de grupos /etc/gshadow. 11

Comandos para administrar usuarios y grupos **13**

Comando useradd 13

Comando userdel 13

Comando passwd 13

Comando usermod 14

Comando chfn 14

Comandos groupadd y groupdel 14

Comando groupmod 14

Comando gpasswd 15

Comando su 15

Comando suDO 15

Comando newgrp 16

Comando id 16

Comandos para gestionar ficheros y directorios. **17**

Comando ls 17

Comando cd 18

Comando pwd 18

Comando mkdir 18

Comando mv 18

Comando cp 19

Comando rm 19
Comando chown 20
Comando chgrp 20
algunos ejercicios sobre estos comandos 21
Comandos para paginar, visualizar y editar ficheros 22
Comando cat 22
Comandos more y less 22
Editor vi 22
Comando echo 23
Flujos de datos. 24
Tuberías 26
Comandos para hacer búsquedas de ficheros y patrones 27
Comando grep 27
Comando find 27
Comando locate 28
Comandos para filtrar ficheros 29
Comando file 29
Comando stat 29
Comando sort 29
Comando uniq 30
Comandos tail y head 30
Comando wc 30
Comando cut 31
Comando sed 32
Comando tr 40
Comandos para compactar y agrupar ficheros 41
Comandos gzip y gunzip 41
Comando tar 41
Comandos para la comunicación entre usuarios 43
Comando write 43

Comando wall....43
Comandos para desconectarse del sistema....44
Comando logout....44
Comando shutdown....44
Comandos varios....45
Comando alias....45
Comando tty....45
Comando du....45
Comando who....46
Comando w....46
Comando finger....46
Comandos de red....48
Comando ping....48
Comando ifconfig....48
Comando route....49
Comando ip....50

COMANDOS BÁSICOS DE GNU-LINUX.

INTRODUCCIÓN.

Una vez instalado e inicializado un sistema Linux se dispone de dos vías fundamentales de interacción: una gráfica (si se instaló una interfaz X y se configuró adecuadamente) y una texto conocida como consola o terminal.

Al igual que Unix, Linux ofrece el mecanismo de consolas o terminales virtuales. Este consiste en que a partir de una entrada estándar (el teclado) y con una salida estándar (el monitor) se simulen varias terminales, donde el mismo, o distintos usuarios puedan conectarse indistintamente. De esta forma es posible tener más de una sesión abierta en la misma máquina y trabajar en ella indistintamente. Este mecanismo también facilita la característica multiusuario del sistema Linux pues las diferentes conexiones se pueden establecer con diferentes usuarios.

Por defecto, las consolas desde la uno a la seis tienen asociado un programa que permite conectarse al sistema en modo texto, mientras que la siete, si se instaló y activó el ``modo gráfico'', constituye una consola gráfica.

El cambio de una consola a otra se realiza a través de la combinación de teclas Alt y Fx (las teclas de Función), donde x oscila entre 1 y 12. De esta forma se pueden acceder un total de 24 consolas virtuales: para las doce primeras se utiliza el Alt izquierdo y para las otras doce el derecho. Por ejemplo, para llegar a la consola 16 se presionarían las teclas Alt derecho y F4.

Desde una consola gráfica para cambiar a otra de tipo texto se debe además presionar la tecla Ctrl, pues las combinaciones Alt + Fx son capturadas e interpretadas por las aplicaciones gráficas de otra forma.

Así, si pulsamos Ctrl – Alt – F1 accedemos a la consola número 1 (tty1), con Ctrl – Alt – F5 accedemos a la consola número 5 (tty5), etc. Si estamos corriendo Debian en una máquina virtual Vmware nos encontraremos con el problema que dicho software utiliza la combinación Ctrl – Alt como hotkey (atajo de teclado) para liberar el cursor del ratón, por lo que tendremos que acceder a la configuración del Vmware y cambiar este hotkey, a por ejemplo, Ctrl – Alt – Mayus.

Con la tecla Alt izquierda combinada con los cursores (derecho e izquierdo) se puede, además, realizar un movimiento circular entre todas aquellas consolas que tengan un proceso asociado (texto, gráfico, etc.). Esta combinación no funcionará una vez que entremos en la consola gráfica.

Si accedemos a una consola en modo texto podremos apreciar que en ella se muestra el nombre de la distribución, la versión de la misma, la versión del kernel y la arquitectura de la máquina. También aparecerá el nombre que se le asignó al sistema en la instalación y la palabra login.

Aquí puede entrarse el nombre de un usuario del sistema. Luego se pedirá la contraseña o password de dicho usuario (tened cuidado ya que al entrar dicho password no se muestra ningún eco en la pantalla). Si ambos son válidos se establecerá la conexión y se mostrará lo que se conoce como prompt del sistema, con forma similar a esta:

```
usuario@pinguino:~$
```

Aquí ha abierto sesión un usuario con nombre "usuario", en una máquina que se llama "pinguino", está actualmente en el directorio "~" y sabemos que es un usuario normal y no el root por que el prompt termina con un símbolo "$" (si fuera root terminaría con un símbolo "#").

Este entorno de texto donde nos encontramos y que nos permite introducir comandos es conocido comúnmente como Shell (caparazón). Este Shell es capaz de interpretar una gran gama de comandos y sentencias. Constituye a su vez un poderoso lenguaje de programación mediante scripts.

GNU-Linux tiene la filosofía de no obligar al usuario a utilizar un programa determinado para cada acción, sino que siempre da la libertad de elegir el programa que queremos utilizar. Lo mismo ocurre con el Shell que vayamos a utilizar para acceder al sistema. El Shell que más se usa es conocido como bash, aunque existen una gran variedad de ellos, como por ejemplo csh, ksh, etc.

Algunas características que merece la pena conocer de bash son:

- Auto completar durante la escritura. Al teclear uno o varios caracteres se puede pulsar TAB con el objetivo de que en caso de que pueda completarse de forma unívoca un comando, nombre de fichero o una variable (en dependencia del contexto), complete de forma automática (se escriba el resto de la palabra). Si existieran varias posibilidades para completar la palabra, se oirá un sonido y volviendo a pulsar TAB se mostrarán en pantalla todas las posibilidades existentes. En caso de existir muchas posibilidades (por defecto más de 100) se pregunta si se desea mostrarlas todas o no.

- Historial de comandos. Esta es una facilidad de muchos otros shells que permite el movimiento a través de los últimos N comandos ejecutados, en la sesión actual o en las anteriores. N por defecto es 1000, pero puede modificarse. Para moverse arriba y abajo se suelen utilizar los cursores, y podemos realizar búsquedas con Control [r]. Podemos indicar que se repita una línea de comando con ¡n y también podemos indicar que se repita una línea de comando que empieza por determinada palabra ¡palabra.

- Poderosas estructuras de control para realizar scripts. (Procesos por lotes).

- Definición de funciones y alias para comandos. Las funciones permiten definir subrutinas programadas usando el lenguaje de bash y los alias, asociar nombres a llamados a comandos con ciertas opciones y argumentos de forma más nemotécnica o abreviada.

FORMAS DE OBTENER AYUDA EN LINUX.

Existen múltiples y variadas formas de obtener ayuda en un sistema Linux. A continuación se describen algunas de ellas:

Muchos comandos poseen una opción para mostrar una ayuda breve acerca de su utilización. Esta opción usualmente consiste en utilizar el parámetro -h, --help o -? tras el nombre del comando.

```
mkdir  --help
```

El comando help, que muestra en algunos comandos integrados del bash un manual propio.

```
help alias
```

El comando info que muestra información sobre los comandos en una estructura de hipertexto.

```
info mkdir
```

El comando whatis que nos da una ayuda rápida sobre comandos.

```
whatis cp
```

El comando apropos, que dada una palabra busca los comandos relacionados sobre ella.

```
apropos delete
```

El comando man muestra un manual bastante amplio acerca de comandos, formatos de ficheros de configuración, llamadas al sistema, etc. Los manuales están disponibles y pueden instalarse en múltiples idiomas. Estos se dividen internamente en secciones. Un mismo objetivo puede estar representado en varias secciones. De no especificarse ninguna sección a través del primer argumento del comando se tomará la primera donde aparezca.

```
man mkdir
```

USUARIOS Y GRUPOS.

Como ya se ha afirmado Linux es un sistema multiusuario, lo cual permite que varios usuarios puedan conectarse y trabajar en él de forma simultánea. Las conexiones como ya se ha visto se pueden realizar a través de varias terminales locales o utilizando servicios de red como el Telnet y SSH.

Un usuario se caracteriza por su login el cual debe indicar para conectarse al sistema, y por su password o contraseña. Además, puede poseer un conjunto de datos adicionales como el domicilio, teléfono, etc.

El usuario con más privilegios en Linux es **root**. Este es el único con derechos suficientes para crear o eliminar a otros usuarios, además de acceder a todo el sistema de ficheros sin ninguna restricción.

En Linux existen grupos de usuarios que permiten otorgar los mismos privilegios a un conjunto de usuarios. Siempre que se añada un usuario al sistema se creará un grupo con su mismo nombre, llamado grupo primario. Durante la creación o posteriormente, se podrá incorporar el usuario a otros grupos secundarios. Así cuando creamos el usuario Olegario, el sistema creará automáticamente un grupo llamado Olegario que contará como único miembro con el usuario creado.

Tanto los usuarios como los grupos se identifican por el sistema a través de un identificador (ID) numérico. El usuario root siempre tiene el ID cero. Cada usuario cuando se conecta al sistema posee un identificador de usuario asociado (uid) y uno o varios identificadores de grupo (gid).

Al añadir un usuario también se creará un directorio base para el mismo con el nombre de su login. Este directorio se coloca por defecto en el directorio /home/*nombredelusuario* excepto para root, cuyo directorio base es /root.

La información asociada a los usuarios en un sistema Linux se guarda en el fichero **/etc/passwd** y las contraseñas y datos afines en **/etc/shadow**. Por su parte la información de los grupos, sus miembros y passwords están en **/etc/group** y **/etc/gshadow** respectivamente. (Vemos a la derecha un ejemplo del fichero /etc/passwd).

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
news:x:9:13:news:/etc/news:
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
```

Para crear un usuario se usa el comando **useradd**, aunque es mucho mejor usar un script que viene instalado en Linux, que se llama **adduser**. Este script controla como funciona useradd y permite realizar funciones avanzadas, como crear automáticamente el directorio del usuario, configurar su perfil, etc.

Como ejemplo, vamos a realizar el siguiente ejercicio (abrir la consola de root para realizarlo):

1) Creamos dos usuarios, uno con nombre margarita y otro con nombre floripondio. En ambos usamos de contraseña 123 por ejemplo.

```
adduser   margarita
adduser   floripondio
```

2) Creamos un grupo con nombre flores, y añadimos a los dos usuarios anteriores a dicho grupo.

```
addgroup flores
adduser margarita flores
adduser floripondio flores
```

3) Visualizamos el fichero /etc/passwd y comprobamos como se han creado dos líneas, una para cada uno de los usuarios que hemos creado.

```
cat /etc/passwd
```

4) Visualizamos el fichero /etc/shadow y veremos cómo se han creado también dos líneas, una para cada uno de los usuarios.

```
cat /etc/shadow
```

5) Visualizamos el fichero /etc/group y comprobamos como se ha creado una línea para el grupo creado, donde además comprobamos que se han añadido como miembros los usuarios.

```
cat /etc/group
```

Para comprobar que los usuarios se han creado bien, vamos a realizar lo siguiente:

A. acceder al 4º terminal (Control – Alt – F4), y hacer un login con el usuario margarita. Una vez abierta sesión, ejecutar el comando `whoami` y el comando `id`. Deberíamos comprobar que efectivamente estamos en el grupo flores. Para cerrar la sesión usamos el comando logout.

B. Acceder al 5º terminal, hacer un login con floripondio y comprobad lo mismo.

(Para volver al terminal grafico desde un terminal de texto, accedemos al 7º terminal).

Comprobaremos como toda la gestión de usuarios y grupos de Linux, en realidad se basa en la modificación de una serie de ficheros de texto que están en el directorio /etc. Esto es algo que como veremos, es común a todo Linux, y siempre que configuremos algo, en el fondo estaremos editando ficheros de texto.

Estos ficheros los trataremos un poco más adelante, de momento vamos a ver los comandos básicos que necesitaremos para trabajar.

FICHERO DE USUARIOS: /ETC/PASSWD

Este fichero consta de 7 campos separados por el símbolo dos puntos (:).

Nombre	Contraseña	UID	GID	Información	Dir. Personal	Shell

- **Nombre**: Indica el nombre de la cuenta de usuario.
- **Contraseña**: indica una X siempre, ya que en los sistemas operativos modernos la contraseña no se almacena en este mismo fichero, ya que sería un fallo de seguridad muy grande. En su lugar la contraseña se almacena en el fichero /etc/shadow.
- **UID**: es el número que se le asigna a cada usuario en el sistema.
- **GID**: es el número del grupo principal al que pertenece este usuario. (El grupo principal suele tener el mismo nombre del usuario).
- **Información**: varios campos separados por coma, y se utilizan a título informativo, no tienen ningún significado propio importante.
- **Dir. Personal**: directorio donde se almacenará el perfil del usuario (sus documentos y configuraciones). Suele ser habitual que sea /home/*Nombredelusuario.*
- **Shell**: el programa que se debe ejecutar para ofrecerle un terminal de acceso al sistema a este usuario, normalmente es el bash ya que es el shell que utilizamos por defecto. Si no queremos que un usuario pueda tener acceso a estos terminales, bastaría con indicarle aquí un nombre falso, como por ejemplo /bin/false y de este modo este usuario no podría interactuar con el sistema aunque realizara correctamente el login.

Vemos aquí un ejemplo de las últimas 5 líneas de un fichero /etc/passwd.

```
usuario@debian740:~$ tail -5 /etc/passwd
saned:x:111:117::/home/saned:/bin/false
Debian-gdm:x:112:118:Gnome Display Manager:/var/lib/gdm3:/bin/false
usuario:x:1000:1000:usuario,,,:/home/usuario:/bin/bash
hplip:x:113:7:HPLIP system user,,,:/var/run/hplip:/bin/false
mysql:x:114:121:MySQL Server,,,:/nonexistent:/bin/false
usuario@debian740:~$ _
```

EL FICHERO DE CONTRASEÑAS /ETC/SHADOW.

Este fichero consta de 8 campos separados por dos puntos.

Nombre	Contraseña	Ult. cambio	Mínimo	Máximo	Aviso	Inactivo	Caducidad

- **Nombre**: Nombre del usuario.
- **Contraseña**: es la contraseña del usuario encriptada. Esta contraseña hoy en día se representa en tres partes distintas, separadas por el símbolo del dólar ($).
    - Id. Esto nos indica el método de encriptación que fue utilizado. Un valor de 1 indica MD5, un valor de 2 Blowfish, 3 es NT Hash, 5 es SHA-256 y 6 es SHA-512.
    - Salt. Este valor se usa por los algoritmos de encriptación y puede ser de hasta 16 caracteres. Este valor se crea aleatoriamente en cada generación de contraseña.
    - Hash. La contraseña en sí misma. MD5 usa 22 caracteres, SHA-256 usa 43, y SHA-512 usa 86.

    Si nos encontramos con un asterisco en este campo significa que la cuenta está bloqueada. Lo mismo se puede conseguir colocando una exclamación (!) al principio de la contraseña. Una cuenta sin contraseña la veremos o bien con todo este campo en blanco, o bien con dos exclamaciones consecutivas (!!).

- **Ult. Cambio**: indica el día en que se cambió la contraseña por última vez. (Este número indica los días que han transcurrido desde el 1-1-1970).
- **Mínimo**: El mínimo número de días que hay que esperar para cambiar la contraseña.
- **Máximo**: El máximo número de días antes de que la contraseña caduque.
- **Aviso**: Antes de que la contraseña caduque, avisaremos al usuario al llegar a este número de días.
- **Inactivo**: Una vez que la contraseña caduque esperaremos este número de días antes de bloquearla definitivamente.
- **Caducidad**: Es una fecha, en la cual la cuenta automáticamente quedará bloqueada. (Este número indica los días que han transcurrido desde el 1-1-1970).

```
root@debian740:/home/usuario# tail -5 /etc/shadow
saned:*:16186:0:99999:7:::
Debian-gdm:*:16186:0:99999:7:::
usuario:$6$OaLI6P1D$G8TcjA3iODe0J4sQ6JmGy1nmKJwYlw0.0JNREi4dbv3a48b02EBLNw8Bt2C/
VKuaqxw83aTLuIyhhzm1n2IOG/:16186:0:99999:7:::
hplip:*:16372:0:99999:7:::
mysql:!:16393:0:99999:7:::
root@debian740:/home/usuario#
```

Dado que estos campos son un poco especiales, es más fácil consultarlos con el comando chage.

```
chage -l usuario
```

EL FICHERO DE GRUPOS /ETC/GROUP.

Este fichero consta de 4 campos separados por dos puntos.

Nombre	Contraseña	GID	Miembros

- **Nombre**: Es el nombre del grupo
- **Contraseña**: al igual que ocurre con los usuarios, los grupos también pueden tener contraseña, y lo normal es no incluir esta contraseña en este fichero, sino en su propio fichero `/etc/gshadow`.
- **GID**: Es el identificador numérico del grupo.
- **Miembros**: Los nombres de los miembros del grupo separados por coma.

```
root@debian740:/home/usuario# tail -5 /etc/group
usuario:x:1000:
lpadmin:x:119:
ssl-cert:x:120:
mysql:x:121:usuario,margarita
margarita:x:1001:
root@debian740:/home/usuario#
```

EL FICHERO DE CONTRASEÑAS DE GRUPOS /ETC/GSHADOW.

Este fichero consta de 4 campos separados por dos puntos.

Nombre	Contraseña	Administradores	Miembros

- **Nombre**: El nombre del grupo.
- **Contraseña**: Una contraseña encriptada con las mismas características que la existente en el fichero shadow. Si existe una contraseña en un grupo, un usuario que se quiera introducir como miembro del grupo tendrá que saber dicha contraseña ya que el sistema se la pedirá (comando `newgrp`). Si un grupo no tiene una contraseña, solo el root o los administradores del grupo podrán asignar miembros a dicho grupo.
- **Administradores**. Una lista de usuarios separados por coma que pueden añadir o eliminar miembros al grupo.
- **Miembros**: Una lista de usuarios separados por coma. Estos son los usuarios que pueden acceder al grupo sin que se le pida la contraseña. Esta lista debe ser la misma que la que se encuentra en `/etc/group`.

- Miembros. Lista de miembros del grupo separados por coma.

```
root@debian740:/home/usuario# tail -5 /etc/gshadow
usuario:!::
lpadmin:!::
ssl-cert:!::
mysql:!::usuario,margarita
margarita:!::
root@debian740:/home/usuario#
```

COMANDOS PARA ADMINISTRAR USUARIOS Y GRUPOS

COMANDO USERADD

El comando useradd permite añadir nuevos usuarios al sistema, además de establecer la información por defecto de los nuevos usuarios que se añadan. Se encuentra enlazado simbólicamente por el nombre **adduser.** Ambos nombres se pueden emplear indistintamente.

Sintaxis: useradd [opciones] [login]

Ejemplos:

`adduser pepe`	crea el usuario pepe con las propiedades por defecto
`useradd -D`	muestra las propiedades por defecto de los nuevos usuarios
`adduser -D -b /dir`	cambia el directorio base por defecto de los nuevos usuarios (/home) a /dir. El directorio /dir debe existir previamente
`adduser pedro amigos`	añade a pedro al grupo amigos. Esto es una opción que se le añadió a adduser para poder añadir usuarios a grupos rápidamente.

COMANDO USERDEL

El comando userdel permite eliminar definitivamente un usuario del sistema.

Sintaxis: userdel [opciones] <login>

Ejemplo:

`userdel -r pepe`	elimina al usuario pepe y borra su directorio base. Por defecto el directorio base se mantiene

COMANDO PASSWD

El comando passwd permite cambiar el password de un usuario. También puede bloquear, desbloquear y deshabilitar una cuenta. Si se invoca sin argumentos se asume el usuario actual.

Sintaxis: passwd [opciones] [login] Ejemplos:

`passwd pepe`	coloca una contraseña para pepe
`passwd -d pepe`	deshabilita la cuenta del usuario pepe eliminando su password
`passwd -l pepe`	bloquea la cuenta del usuario.
`passwd -u pepe`	desbloquea la cuenta del usuario pepe

COMANDO USERMOD

El comando usermod se emplea para modificar algunas propiedades de los usuarios como: el login, el directorio base, el shell que se inicia al conectarse, los grupos a los que pertenece, la fecha de expiración de la cuenta, etc. También bloquea y desbloquea una cuenta.

Sintaxis: usermod [opciones] <login>

Ejemplos:

`usermod -s /bin/csh pepe`	coloca el shell csh para el usuario pepe
`usermod -G users,disk pepe`	señala como grupos secundarios de pepe a users y disk
`usermod -e 2001-10-20 pepe`	indica que la cuenta de pepe expirará el 20 de octubre de 2001

COMANDO CHFN

El comando chfn permite cambiar la información de contacto de un usuario. Esta incluye aspectos como: el nombre completo, la oficina de trabajo y los teléfonos. Se almacena en el fichero de usuarios /etc/passwd.

Sintaxis: chfn [opciones] [login]

Ejemplo: `chfn pepe`

COMANDOS GROUPADD Y GROUPDEL

Los comandos groupadd y groupdel añaden y eliminan un grupo en el sistema respectivamente.

Sintaxis:

groupadd [opciones] <grupo>	groupdel <grupo>

Ejemplos:

```
groupadd admin
groupdel admin
```

COMANDO GROUPMOD

El comando groupmod permite modificar en un grupo su identificador y nombre.

Sintaxis: groupmod [opciones] <grupo>

Ejemplo: `groupmod -n usuarios users` cambia el nombre del grupo users a usuarios

COMANDO GPASSWD

El comando gpasswd permite administrar los grupos. Se puede utilizar para añadir y eliminar usuarios, señalar un administrador e indicar un password de grupo.

Sintaxis: gpasswd [opciones] <grupo>

Ejemplos:

`gpasswd -A pepe admin`	señala como administrador del grupo admin al usuario pepe
`gpasswd admin`	cambia el passwd del grupo admin
`gpasswd -a joe admin`	añade el usuario joe al grupo admin

COMANDO SU

El comando su permite ejecutar un shell (u otro comando) cambiando los identificadores del grupo y del usuario actual. Si se le pasa - como primer argumento ejecuta el shell como un login shell, o sea se creará un proceso de login tal y como ocurre naturalmente cuando un usuario se conecta al sistema. Si no se especifica el login del usuario se asume root.

Sintaxis: su [opciones] [login]

Ejemplos:

`su -`

`su pepe`	ejecuta un shell haciéndonos pasar por el usuario pepe
`su -c "cat /etc/shadow"`	ejecuta un comando con los privilegios de root en lugar de un shell

COMANDO SUDO

El programa sudo es una utilidad de los sistemas operativos tipo Unix, como GNU/Linux, BSD, o Mac OS X, que permite a los usuarios ejecutar programas con los privilegios de seguridad de otro usuario (normalmente el usuario root) de manera segura.

Los usuarios deben confirmar su identidad al ejecutar sudo dando **su propia contraseña** antes de ejecutar el programa requerido. Una vez se ha autenticado el usuario, y si el archivo de configuración /etc/sudoers permite dar al usuario acceso al comando requerido, el sistema lo ejecuta y lo registra. En un entorno gráfico, se usa `gksudo`.

El archivo de configuración /etc/sudoers especifica qué usuarios pueden ejecutar qué comandos en nombre de qué otros usuarios. Como sudo es muy estricto con el formato de este archivo, y cualquier error podría causar problemas serios, existe la utilidad `visudo`; ésta permite al usuario root editar el archivo y luego revisar su corrección antes de guardarlo.

En Debian no suele venir instalado el paquete sudo (depende de la versión), si queremos instalarlo debemos ejecutar en una terminal el siguiente comando como root.

```
apt-get install sudo
```

Una vez instalado el sudo, comprobaremos con visudo (o bien editando directamente el fichero /etc/sudoers) que debe contener unas líneas como las siguientes

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

Si queremos por ejemplo que el usuario anselmo pueda ejecutar el sudo, bastaría con añadir una línea debajo de la anterior como la siguiente:

```
anselmo ALL(=ALL:ALL) ALL
```

Si queremos ver las distintas posibilidades que podríamos usar para no escribir ALL, tendriamos que mirar la ayuda de Linux sobre este fichero, en este caso bastaría con ejecutar el siguiente commando

```
man sudoers
```

COMANDO NEWGRP

El comando newgrp permite iniciar un nuevo shell cambiando el identificador del grupo (gid) del usuario actual. Sólo podrán hacerlo los usuarios que pertenezcan al grupo o en caso de que este tenga una contraseña, aquellos que la conozcan.

Sintaxis: newgrp [grupo]

Ejemplo:

```
newgrp bin
```

COMANDO ID

El comando id, imprime dado un usuario, sus identificadores de usuario y de grupo principal (gid y uid) así como del resto de los grupos a los que pertenece. Sin argumentos se asume el usuario actual.

Sintaxis: id [opciones] [login]

Ejemplo:

```
id pepe
```

COMANDOS PARA GESTIONAR FICHEROS Y DIRECTORIOS.

COMANDO LS

El comando ls permite listar el contenido de un directorio.

Sintaxis:

```
ls [opciones] [directorio | fichero]
```

Algunas opciones:

- -l: muestra la salida en formato largo.
- -R: lista recursivamente un directorio.
- -a: lista además los ficheros incluidos los ocultos (sus nombres comienzan con punto).
- -h: muestra el tamaño de los ficheros en forma más legible (Ej.: 16M, 4k, etc.)
- -i: muestra el identificador del i-nodo (bloque índice) asociado a cada elemento.

Ejemplos:

```
ls -hl  /etc
ls -R  /usr
ls -al
ls -ali   ..
```

Un grupo de opciones que se suele utilizar bastante es lia (ls –lia)

En linux se considera que un fichero o directorio que comienza por punto es un fichero oculto, y ls no lo mostrará a menos que se lo indiquemos. Hay bastantes órdenes en el sistema que no trabajaran con estos ficheros ocultos a menos que se lo indiquemos explícitamente.

En cualquier momento, el shell bash nos permite usar comodines. Estos comodines son los siguientes:

Comodin	significado	ejemplo
?	Se sustituye por **un carácter** cualquiera.	ls carta?.jpg
*	Se sustituye por **cualquier** número de caracteres, incluyendo **ninguno**.	ls carta*
[a,b,c]	Se sustituye por el **carácter** a, o por el **carácter** b, o por el c……	ls carta[1,2,3,4,5]
[a-b]	Se sustituye por el rango de **caracteres** entre a y b	ls carta[1-5]
{a,b,c}	Se sustituye por a, o por b, o por c…..	ls [*.jpg , *.txt]
[!a]	Se sustituye por cualquier cosa, pero **no** por a (! indica negación)	ls carta[!4]
\a	Se usa para **proteger** a, de modo que bash interprete a como un carácter simple.	echo *

COMANDO CD

El comando cd se utiliza para cambiar el directorio actual.

Sintaxis:

```
cd [directorio]
```

Ejemplos:

```
cd /tmp
cd
cd ~
```

El comando cd o cd ~ nos lleva directamente a nuestro directorio home de usuario. Este directorio está situado normalmente en /home/nombredeusuario para los usuarios normales y en /root para el usuario root. No hay que confundir este directorio home del usuario con el directorio /home del sistema.

Cada usuario tiene control sobre los ficheros y directorios de su home de usuario, y en el caso de los usuarios normales es el único sitio de todo el árbol donde pueden crear ficheros y directorios.

COMANDO PWD

El comando pwd indica el camino absoluto del directorio en el cual nos encontramos actualmente.

COMANDO MKDIR

El comando mkdir se utiliza para crear directorios.

Una opción:

- -p crea los directorios intermedios en caso de que no existan.

Ejemplos:

```
mkdir  bin
mkdir  /bin
mkdir  -p  docs/linuxdocs/howtos/pdf
```

COMANDO MV

El comando mv mueve un fichero hacia otro, o varios ficheros hacia un directorio. Este permite a su vez renombrar ficheros o directorios.

Sintaxis:

```
mv [opciones] <fuente> <destino>
mv [opciones] <ficheros> <directorio>
```

Algunas opciones:

- -i: ejecuta el comando de forma interactiva, o sea, pregunta ante de sobrescribir el destino si existiera.

- -u: actualiza (upgrade) el destino con el fuente solo si este es más reciente.

Ejemplos:

```
mv   mail.cf   mail.cf.old
mv  -i   *.txt   /tmp
mv  -u   program.c   src/
```

COMANDO CP

El comando cp permite copiar un fichero en otro, o varios ficheros en un directorio.

Sintaxis:

```
cp [opciones] <fuente> <destino>
cp [opciones] <ficheros> <directorio>
```

Algunas opciones:

- -R: copia recursivamente un directorio.
- -i: utiliza una forma interactiva (pregunta antes de sobrescribir el destino).
- -l: hace enlaces fuertes a los ficheros fuentes en lugar de copiarlos.

Ejemplos:

```
cp  /etc/passwd  .
cp  -i  /usr/bin/*sh  /
```

COMANDO RM

El comando rm se utiliza para borrar ficheros y directorios (remove)

Sintaxis:

```
rm [opciones] <ficheros | directorios>
```

Algunas opciones:

- -r: borra recursivamente un directorio.
- -f: borra forzosamente en caso de que no se tenga permiso de escritura en forma directa.
- -i: ejecuta el comando de forma interactiva.

Ejemplos:

```
rm   prueba
rm -i   bin/*
rm  -rf   temp/
```

COMANDO CHOWN

El comando chown se utiliza para cambiar el dueño y el grupo de un fichero. El dueño de un fichero solo lo puede cambiar el usuario root mientras que el grupo además de root, lo puede cambiar el propio dueño, siempre que pertenezca al nuevo grupo.

Sintaxis:

```
chown [opciones] dueño[.grupo] <ficheros>
```

Opción:

- -R: en los directorios cambia el dueño y/o el grupo recursivamente.

Ejemplos:

```
chown  pepe.pepe  tesis
chown  -R  mar.users   /tmp/oculto
chown  jose carta.txt
```

COMANDO CHGRP

El comando chgrp se utiliza para cambiar el grupo de un fichero.

Sintaxis:

```
chgrp [opciones] grupo <ficheros>
```

Opción:

- -R: en los directorios cambia el dueño y/o el grupo recursivamente.

Ejemplos:

```
chgrp –R arquitectos /usr/planos/
```

ALGUNOS EJERCICIOS SOBRE ESTOS COMANDOS.

Vamos a proponer algunos ejercicios para reforzar los conocimientos de los comandos para manipular ficheros y directorios.

1) Crear un directorio ejercicio1 en vuestro home, y copiar en el mismo todos los ficheros con extensión conf del directorio /etc. Realizad esta acción con un solo comando, y posicionados en el directorio home del usuario.
2) Crear un directorio ejercicio2 dentro de un directorio prueba dentro de un directorio control dentro de un directorio alumno dentro de vuestro home. Realizad esta acción con un solo comando y posicionados en el directorio home de vuestro usuario.
3) Moved todos los ficheros que contengan una letra a en su nombre del directorio ejercicio1 al directorio ejercicio2. Un solo comando y posicionados en el raíz de vuestro árbol.
4) Cread un usuario rigoberto. Modificar todos los ficheros del directorio ejercicio2 para que pasen a ser propiedad del usuario rigoberto y del grupo rigoberto.
5) Copiad el directorio alumno de vuestro home, y todo lo que contiene, a un directorio copia_alumno que debe crearse en vuestro directorio home. Un solo comando para la copia.
6) Borrar el directorio alumno de vuestro home con un solo comando y sin que pida confirmación.
7) Cread un fichero con nombre ocultos en la raíz de vuestro árbol, que contenga el nombre de todos los directorios ocultos que están en vuestro home. (Los ficheros y directorios ocultos son aquellos cuyo nombre comienza por un punto, y no son mostrados normalmente por ls).

COMANDOS PARA PAGINAR, VISUALIZAR Y EDITAR FICHEROS

COMANDO CAT

El comando cat nos permite concatenar (conCATenate) ficheros, es decir, coger varios ficheros de texto y unirlos en uno solo. Por defecto cat manda los ficheros a la salida estándar del sistema (stdout) que es la pantalla, por lo que normalmente utilizamos esta orden para visualizar ficheros de texto.

Sintaxis:

```
cat [opciones] <ficheros>
```

En un punto posterior veremos la forma de usar redirecciones, lo que nos permitirá usar el cat de un modo más interesante, pero de momento lo dejamos aquí.

COMANDOS MORE Y LESS

Los comandos more y less paginan (dividen en páginas) uno o varios ficheros y los muestran en la terminal (pantalla). Se diferencian en las facilidades que brindan. Por ejemplo more es más restrictivo en cuanto al movimiento dentro del texto, mientras que less no limita este aspecto pues acepta el empleo de todas las teclas de movimiento tradicionales. Cuando se alcanza el final del último fichero a paginar, more termina automáticamente, no así less.

Algunas de las funciones que podemos realizar con less los siguientes:

Ejemplos:

```
less /etc/passwd
more /etc/passwd
```

Comodin	significado
q	Sale del programa.
/<cadena>	Realiza búsquedas de la cadena <cadena>
/	Repite la última búsqueda.
<n>b	Back (retrocede) n páginas.
<n>f	Forward (avanza) n páginas.

El comando man, para dar formato a su salida, utiliza por defecto el visualizador less.

EDITOR VI

El editor vi es el editor estándar de Unix y está orientado a comandos. Existe una versión conocida como vim (Vi IMproved) que permite la edición de múltiples ficheros, edición automática para varios lenguajes de programación, ayuda en línea, selección visual, varios niveles de deshacer, etc. Para algunos usuarios, vi resulta incómodo pues para utilizar todas sus potencialidades es necesario conocer muchas combinaciones de teclas, pero si se llega a dominar resulta muy funcional.

Su principal virtud es que encontraremos vi en prácticamente cualquier versión de Unix que usemos, cosa que no se puede decir de otros editores (joe, pico, nano, edit, gedit, etc.).

Básicamente vi posee dos modos de interacción: el de inserción (edición) y el de comandos. Para pasar al modo comando se pulsa Esc y para pasar al de inserción se pulsa i.

Algunos comandos útiles en vi (pulsando ESC para pasar al modo de comandos, que veremos en la última línea de la pantalla).

Comando	Acción
dd	Borra la línea actual.
D	Borra desde la posición actual hasta el final de la línea.
dG	Borra hasta el final del fichero.
u	Deshace el último comando.
:q	Sale del editor (si se han hecho modificaciones y no se ha salvado se genera un error).
:q!	Sale sin salvar.
:w	Salva.
:wq	Salva y sale.
:x	Salva y sale.
<n><comando>	Ejecuta el <*comando*> *n* veces.

Hay que tener mucho cuidado al usar vi con los cursores. Estos sólo funcionan en el modo de comandos, de modo que si estamos en inserción debemos acostumbrarnos a no usar los cursores ya que se interpretarán como caracteres y empezaran a aparecernos caracteres extraños en pantalla.

COMANDO ECHO

Es un comando que nos permite escribir algo directamente por la salida estándar.

Opciones:

-n no salta de línea tras escribir
-e activa la interpretación de los caracteres especiales precedidos por \

Ejemplos:

```
echo Hola Mundo
echo Hola Mundo -n
echo adiós
echo Hola \\t Mun \\t do
```

Sí queremos imprimir por pantalla caracteres especiales, tendremos que protegerlos. Dependiendo del carácter, nos bastará con introducirlos entre comillas dobles, o comillas simples, o en el peor de los casos protegerlos escribiendo una contrabarra (\) delante del carácter a proteger.

FLUJOS DE DATOS.

En Linux al igual que en Unix todos los procesos (programas en ejecución) tienen asociados tres flujos (streams) de datos principales. Estos son:

- La entrada estándar. (stdin) Es donde un proceso puede tomar los datos que maneja y que no se indican mediante argumentos u opciones. Por defecto se toma del teclado.
- La salida estándar. (stdout) Es donde un proceso escribe los resultados de su ejecución. Por defecto es la terminal (pantalla) donde se invocó el programa correspondiente.
- La salida de errores. (stderr) Es donde un proceso escribe los posibles errores durante su ejecución. Por defecto es la terminal (pantalla) donde se invocó el programa correspondiente.

Los flujos de datos se almacenan en descriptores de ficheros que se identifican por un número en la forma siguiente:

- 0: representa la entrada estándar.
- 1: representa la salida estándar.
- 2: representa la salida de errores.

En bash, al igual que en otros shells, los flujos de datos se pueden redireccionar libremente hacia distintos ficheros. En esencia este mecanismo consiste en que la salida de un proceso (estándar o de errores) puede escribirse en un fichero en lugar de la terminal asociada, así como la entrada puede tomarse también a partir de un fichero en lugar de utilizar lo escrito mediante el teclado.

Para indicar un redireccionamiento se utilizan los signos de comparación < y >. De esta forma se generan dos tipos de construcción:

- instrucción > salida: indica el redireccionamiento del flujo de datos de la instrucción al fichero nombrado salida.
- Instrucción >> salida: indica el redireccionamiento del flujo de datos de la instrucción al fichero nombrado salida, añadiendo al contenido actual de dicho fichero el flujo de datos.
- Instrucción < entrada: indica el redireccionamiento del contenido del fichero nombrado entrada como flujo de datos de entrada para la instrucción.
- Instrucción 2> salida: indica el redireccionamiento del flujo de datos DE LOS ERRORES de la instrucción al fichero nombrado salida.

Ejemplos:

`# cat /etc/shadow > cont`	Se redirecciona la salida estándar hacia un fichero con nombre cont
`$ ls > /dev/null`	Se redirecciona la salida estándar al dispositivo especial de caracteres /dev/null provocando su desaparición.
`$ ls /tmp 2> fich-errores`	Se redirecciona la salida de errores hacia un fichero fich-errores.
`$ echo Hola Mundo > saludo.txt`	Crea un fichero saludo.txt que contendrá Hola Mundo
`$ echo Adios a todos > saludo.txt`	Crea un fichero saludo.txt que contendrá Adios a todos
`$ echo Jerez >> saludo.txt`	Añade una linea al fichero saludo.txt con el texto Jerez. (Si saludo.txt no existe lo crea).

La redirección < que nos permite redireccionar la entrada tiene usos específicos. Así, por ejemplo, si escribimos el siguiente comando:

```
cat < fichero.txt > fichero2.txt
```

le estamos indicando al sistema que concatene (cat) la entrada (fichero.txt) a la salida (fichero2.txt). Esto vendría a ser lo mismo que cp fichero.txt fichero2.txt.

TUBERÍAS

Las tuberías (pipes) son un poderoso mecanismo del shell en Unix. Este en esencia permite tomar la salida de un comando y pasársela como entrada a otro.

Muchos de los comandos mencionados anteriormente, que reciben como argumento un fichero, en caso de omitirse este, utilizan su entrada estándar. Esta entrada puede provenir a su vez de la salida de otros comandos. Gracias a esto las tuberías permiten realizar filtrados de los más diversos tipos

Las tuberías pueden estar formadas por un número ``ilimitado'' de comandos. Estos no se ejecutan secuencialmente, o sea no se espera a que termine uno para ejecutar el siguiente, sino que se va haciendo de forma concurrente. El carácter que se emplea para separar un comando de otro mediante una tubería es |. (Alt Gr + 1 ó Alt + 124)

Ejemplos:

`$ ls -l /dev | less`	Toma la salida de ls y la envia como entrada a la orden less, con lo que se consigue un listado paginado.

Es importante tener en cuenta la diferencia entre tubería y redirección. Las tuberías separan comandos, es decir, a izquierda y derecha de una tubería hay comandos de linux. La redirección direcciona un comando a un fichero o directorio, es decir, bien a la izquierda o bien a la derecha de una redirección no existirá un comando de linux, sino un fichero o directorio.

Estos ejemplos por lo tanto estarían mal y son erróneos:

```
ls -lia | fichero.txt
echo hola Mundo | carta.txt
cat /etc/passwd > less
```

Tomemos por ejemplo el comando sort, que permite ordenar un flujo de texto. Si queremos ordenar el contenido del fichero carta.txt lo podemos hacer así:

sort carta.txt

pero también sería válido hacerlo así:

cat carta.txt | sort

lo que estaría mal sería lo siguiente:

cat carta.txt > sort (nos crearía un fichero con nombre sort y el contenido de carta.txt)

COMANDOS PARA HACER BÚSQUEDAS DE FICHEROS Y PATRONES

COMANDO GREP

El comando grep (Globally Regular Expressions Pattern) busca patrones en ficheros. Por defecto devuelve todas las líneas que contienen un patrón (cadena de texto) determinado en uno o varios ficheros. Utilizando las opciones se puede variar mucho este comportamiento. Si no se le pasa ningún fichero como argumento hace la búsqueda en su entrada estándar.

Sintaxis:

```
grep [opciones] <patrón> [ficheros]
```

Algunas opciones:

O.	Resultado.
-c	Devuelve sólo la cantidad de líneas que contienen al patrón.
-i	Ignora las diferencias entre mayúsculas y minúsculas.
-H	Imprime además de las líneas, el nombre del fichero donde se encontró el patrón. Es así por defecto cuando se hace la búsqueda en más de un fichero.
-l	Cuando son múltiples ficheros sólo muestra los nombres de aquellos donde se encontró al patrón y no las líneas correspondientes.
-v	Devuelve las líneas que no contienen el patrón.
-r	Busca en un directorio de forma recursiva.
-n	Imprime el número de cada línea que contiene al patrón.

Ejemplos:

```
grep linux /usr/share/doc
grep root  /etc/passwd
grep -n error /var/log/messages
grep -i pepe /etc/passwd
grep -c root /etc/group
grep -l -r -i img /var/www/html/
cat /etc/passwd | grep -w root
cat /etc/passwd | grep -c /bin/bash
```

COMANDO FIND

El comando find es uno de los más poderosos en un sistema Linux. Permite buscar de forma recursiva en un directorio todos los ficheros que cumplan ciertas condiciones. Las condiciones pueden estar relacionadas con el nombre de los ficheros, el tamaño, los permisos, el tipo, las fechas de acceso y modificación, etc.

Sintaxis:

```
find [ruta desde donde se busca] [opciones de búsqueda]
```

Algunas opciones:

Opción.	Resultado.
-name <expresión>	Permite especificar patrones para los nombres de los ficheros a buscar.
-iname <expresión>	Igual que el anterior, pero ignora mayúsculas/minúsculas
-type <tipo>	Permite indicar el tipo de fichero a buscar.
-size +/-<n>	Permite indicar el tamaño máximo y/o mínimo de los ficheros a buscar.
-perm [+|-]<modo>	Permite referirse a aquellos ficheros con unos permisos dados. El valor de <modo> se expresa en forma numérica.
-exec <comando> ;	Permite definir un comando a ejecutarse para cada resultado de la búsqueda. La cadena {} se sustituye por el nombre de los ficheros encontrados. El carácter; permite indicar la finalización del comando. (Tanto {} como; tienen que ir entre comillas o entre contrabarras para evitar que sea sustituido por el shell).

Ejemplos:

```
find /etc -name '*.conf'
find / -size +10240k -size -20480k
find -perm +1000 -type d
find / -name core -exec rm -i “{}”  “;”
```

COMANDO LOCATE

El comando locate busca en una base de datos, actualizada periódicamente, todos los paths en la jerarquía de ficheros que contengan una cadena determinada. Para crear esta base de datos o actualizarla se debe invocar por root el comando updatedb (o locate -u). En las distribuciones actuales esta tarea se hace de forma automática.

Ejemplo:

```
locate passwd
```

COMANDOS PARA FILTRAR FICHEROS

COMANDO FILE

El comando file determina con cierto grado de precisión el tipo de un fichero que se le pasa como argumento.

Ejemplos:

```
file /etc/passwd
file /usr/sbin/adduser
file /usr/sbin/useradd
```

COMANDO STAT

El comando stat muestra las características de un fichero. Por ejemplo: su nombre, permisos, tamaño en bytes, número del i-nodo que lo representa, las fechas de modificación y acceso, el tipo, el dueño, el grupo, etc.

Ejemplos:

```
stat /etc/shadow
```

COMANDO SORT

El comando sort ordena las líneas de un fichero mostrándolas por la salida estándar. De no especificarse un fichero toma la entrada estándar.

Sintaxis:

```
sort [opciones] [fichero]
```

Algunas opciones:

-r: ordena al revés.

-f: trata las mayúsculas y minúsculas por igual.

Ejemplo:

```
sort -f /etc/passwd
```

Como ejercicio, cread un archivo llamado lista-desordenada con el vi y meter dentro 5 nombres desordenados. Comprobad como con la orden sort lista-desordenada vemos por pantalla el fichero pero ordenado. Comprobar que en realidad el fichero no ha cambiado visualizándolo con cat.

COMANDO UNIQ

El comando uniq elimina las líneas repetidas de un fichero ordenado, imprimiéndolo por la salida estándar o en otro fichero argumento. De no especificarse un fichero toma la entrada estándar.

Sintaxis:

```
uniq [opciones] [fichero] [salida]
```

Algunas opciones:

-c: utiliza como prefijo en cada línea su número de ocurrencias.

-d: solo imprime las líneas duplicadas.

Ejemplo:

```
uniq -d lista.txt
```

COMANDOS TAIL Y HEAD

Los comandos tail y head muestran respectivamente el final y el comienzo (10 líneas por defecto) de uno o varios ficheros. De no especificarse al menos un fichero toman la entrada estándar.

Sintaxis:

```
tail  [opciones] [ficheros]
head  [opciones] [ficheros]
```

Algunas opciones:

-q no coloca los encabezamientos con el nombre de los ficheros cuando se indican varios ficheros.

-<n> imprime las n últimas (tail) o primeras (head) líneas en lugar de las diez establecidas por defecto.

Ejemplos:

```
tail -f /var/log/messages
tail -20 /var/log/secure
head -50 /var/spool/mail/pepe
head -2 -q /etc/*.conf
```

COMANDO WC

El comando wc (Word Count) imprime el número de líneas, palabras y bytes de uno o varios ficheros. Si son varios ficheros hace también un resumen de los totales. De no especificarse un fichero toma la entrada estándar.

Sintaxis:

```
wc [opciones] [ficheros]
```

Algunas opciones:

O.	Resultado.
-l	Sólo cuenta líneas.
-m	Sólo cuenta caracteres.
-c	Sólo cuenta bytes.
-w	Sólo cuenta palabras.

Ejemplos:

```
wc -l /etc/passwd
wc -w /doc/dicciorario.txt
```

Un pequeño ejemplo que os resultará útil. Teclead las siguientes líneas.

```
echo hola > fichero.txt
wc –m fichero.txt
```

Con la primera orden creamos un fichero denominado fichero.txt cuyo contenido es la palabra hola (hemos usado una redirección, las estudiaremos dentro de poco). Con la segunda orden contamos los caracteres que hay en el fichero y lo mostramos por pantalla. En principio esperaríamos que nos mostrara 4 pero sin embargo nos muestra 5. ¿Por qué?

Escribid la siguiente orden:

```
hd fichero.txt
```

Esta orden nos muestra una representación hexadecimal en raw (en crudo) de nuestro fichero. Fijaros como en el no vemos 4 caracteres, sino 5. Existe un 5º carácter con el número hexadecimal 0a. Este carácter es el retorno de línea que añade echo automáticamente al final, y que es un carácter que si bien nosotros no vemos, si existe para el sistema y wc cuenta automáticamente.

COMANDO CUT

Como su propio nombre indica, el comando cut tiene la característica de poder cortar caracteres y campos, con la posibilidad de usar delimitadores y otras opciones, para finalmente extraer las partes seleccionadas de cada fichero en la salida estándar.

El comando cut nos ofrece los siguientes argumentos:

Argumento.	Resultado.
-b, –bytes=LISTA	Muestra solamente estos bytes
-c, –characters=LISTA	Selecciona solamente estos caracteres
-d, –delimiter=DELIM	Usa DELIM en vez de caracteres de tabulación para delimitar los campos
-f, –fields=LISTA	Selecciona solamente estos campos; también muestra cualquier línea que no tenga un carácter delimitador, a menos que se especifique la opción -s

Ejemplos:

```
echo "Esto es una prueba, 1 2 3, probando"  > prueba.txt
cut –d "," -f 3 prueba.txt
cut  -c2-4 prueba.txt
cut –d “a” –f2 prueba.txt
```

COMANDO SED

En realidad sed no es un comando, sino un editor de textos que está diseñado para ser usado desde línea de comandos. Es muy útil cuando queremos modificar texto de forma no interactiva, veamos un ejemplo:

```
echo "Hola mundo azul" > saludo.txt
sed –e  "s/mundo/amigo/" saludo.txt
```

Sintaxis:

sed [opciones] [delimitador] función [argumentos]

Algunas de las opciones que podemos utilizar con sed son las siguientes:

opción	Resultado.
-e	indica que queremos ejecutar el comando siguiente. Si solo se desea ejecutar una función, no hace falta ponerlo. Si queremos utilizar varias funciones se puede poner –e delante de cada una de ellas, o separarlas con el símbolo punto y coma ";".
-n	indica que no queremos que nos escriba el resultado normal de la orden por pantalla.
-i	sed no modifica los ficheros realmente, sino que muestra su salida por pantalla. Con la opción –i obligamos a que sed modifique el fichero en sí mismo.

El delimitador es un ámbito que puede ser o bien un número de línea, o bien un rango de líneas indicados por num1,num2. Tambien podemos indicar que el ámbito responde a las líneas que coincidan con una cadena introducida entre //.

Algunas de las funciones que podemos utilizar con sed son las siguientes:

Función	Resultado
s	Sustituir texto. /texto_antiguo/texto_nuevo/ es la forma de indicar que queremos sustituir. Si no ponemos nada más solo se sustituirá la primera aparición del texto, si queremos que se sustituyan todas las apariciones, hay que poner g al final. Si queremos que se sustituya solo la primera aparición, hay que poner 1 al final (es el compartamiento por defecto). Si ponemos 2 se sustituirá la segunda aparición, etc.
r	Añade un fichero entero a la línea actual. (r fichero).
p	Imprime líneas de texto
d	borra texto
i	Inserta líneas antes de la línea actual.
a	Inserta líneas debajo de la línea actual.
c	Cambia la línea actual.
w	Escribe la salida a un fichero. Se usa indicando de una línea a otra y luego w.
q	Termina el comando en la línea indicada.
y	Cambia los caracteres de una cadena por los de otra, utilizando el orden de los mismos.
=	Imprime el número de linea

Como delimitador en sed se suele utilizar la barra, para separar funciones, parámetros, etc.

```
usuario@debv3:~/Documentos$ cat dos.txt
Susanita tiene un raton
un raton chiquitin
usuario@debv3:~/Documentos$ sed -n 1p dos.txt
Susanita tiene un raton
usuario@debv3:~/Documentos$ sed -n /chi/p dos.txt
un raton chiquitin
```

Algunos ejemplos de borrado de líneas “d”:

```
usuario@debv3:~/Documentos$ cat fichero.txt
Esta es la linea número uno
Esta es la dos
Y esta de ahora es la tres
La cuatro toca ahora
Y acabamos con la linea cinco
usuario@debv3:~/Documentos$ sed 3d fichero.txt
Esta es la linea número uno
Esta es la dos
La cuatro toca ahora
Y acabamos con la linea cinco
usuario@debv3:~/Documentos$ sed 2,4d fichero.txt
Esta es la linea número uno
Y acabamos con la linea cinco
```

Un par de ejemplos con sustituciones “s”.

```
usuario@debv3:~/Documentos$ cat fichero.txt
Esta es la linea número uno
Esta es la dos
Y esta de ahora es la tres
La cuatro toca ahora
Y acabamos con la linea cinco
usuario@debv3:~/Documentos$ sed s/linea/almohada/ fichero.txt
Esta es la almohada número uno
Esta es la dos
Y esta de ahora es la tres
La cuatro toca ahora
Y acabamos con la almohada cinco
usuario@debv3:~/Documentos$ sed s/linea/almohada/g fichero.txt
Esta es la almohada número uno
Esta es la dos
Y esta de ahora es la tres
La cuatro toca ahora
Y acabamos con la almohada cinco
```

Ejemplo de imprimir “p” y para que se usa la opción “n”.

```
usuario@debv3:~/Documentos$ cat uno.txt
Con cien cañones por banda
viento en popa a toda vela
no corta el mar sino vuela
un velero bergantin.
usuario@debv3:~/Documentos$ sed 2p uno.txt
Con cien cañones por banda
viento en popa a toda vela
viento en popa a toda vela
no corta el mar sino vuela
un velero bergantin.
usuario@debv3:~/Documentos$ sed -n 2p uno.txt
viento en popa a toda vela
```

Algunos ejemplos añadiendo líneas a un fichero “i” y “a”.

```
usuario@debv3:~/Documentos$ cat dos.txt
Susanita tiene un raton
un raton chiquitin
usuario@debv3:~/Documentos$ sed 2i"Hola Monstruo" dos.txt
Susanita tiene un raton
Hola Monstruo
un raton chiquitin
usuario@debv3:~/Documentos$ sed 2a"Hola Monstruo" dos.txt
Susanita tiene un raton
un raton chiquitin
Hola Monstruo
```

Un ejemplo cambiando líneas “c”.

```
usuario@debv3:~/Documentos$ cat uno.txt
Con cien caños por banda
viento en popa a toda vela
no corta el mar sino vuela
un velero bergantin.
usuario@debv3:~/Documentos$ sed 3c"Hola" uno.txt
Con cien caños por banda
viento en popa a toda vela
Hola
un velero bergantin.
```

Un ejemplo insertando un fichero “r”.

```
usuario@debv3:~/Documentos$ cat uno.txt
Con cien caños por banda
viento en popa a toda vela
no corta el mar sino vuela
un velero bergantin.
usuario@debv3:~/Documentos$ cat dos.txt
Susanita tiene un raton
un raton chiquitin
usuario@debv3:~/Documentos$ sed 1r"dos.txt" uno.txt
Con cien caños por banda
Susanita tiene un raton
un raton chiquitin
viento en popa a toda vela
no corta el mar sino vuela
un velero bergantin.
```

Un ejemplo grabando un fichero con “w”.

```
usuario@debv3:~/Documentos$ cat uno.txt
Con cien caños por banda
viento en popa a toda vela
no corta el mar sino vuela
un velero bergantin.
usuario@debv3:~/Documentos$ sed -n 1,2w"salida.txt" uno.txt
usuario@debv3:~/Documentos$ cat salida.txt
Con cien caños por banda
viento en popa a toda vela
```

Un ejemplo de sustitución con la “y”.

```
usuario@debv3:~/Documentos$ cat dos.txt
Susanita tiene un raton
un raton chiquitin
usuario@debv3:~/Documentos$ sed y/abcdefg/1234567/ dos.txt
Sus1nit1 ti5n5 un r1ton
un r1ton 3hiquitin
```

Algunos ejercicios:

1) Queremos sustituir en el fichero novela.txt todas las ocurrencias de “co” por “koko” pero solo en las 3 primeras líneas.
2) Queremos sustituir en el fichero novela.txt todas las vocales por la letra u usando varios –e en la orden sed.
3) Imprimir solo las 3 primeras líneas del fichero novela.txt
4) Crear un fichero canción.txt que contenga “Cuando Fernando Septimo usaba paleton”. Mostrarlo por pantalla con una orden sed sustituyendo todas las vocales por la letra u usando la función “y”.

Solución al primer ejercicio:

```
usuario@debv3:~/Documentos$ cat novela.txt
Muchos años después, frente al pelotón de fusilamiento,
el coronel Aureliano Buendía había de recordar
aquella tarde remota en que su padre lo llevó a conocer el hielo.
Macondo era entonces una aldea de 20 casas de barro y cañabrava
construidas a la orilla de un río de aguas diáfanas que se precipitaban
por un lecho de piedras pulidas, blancas y enormes
como huevos prehistóricos.
usuario@debv3:~/Documentos$ sed 1,3s/co/koko/g novela.txt
Muchos años después, frente al pelotón de fusilamiento,
el kokoronel Aureliano Buendía había de rekokordar
aquella tarde remota en que su padre lo llevó a kokonocer el hielo.
Macondo era entonces una aldea de 20 casas de barro y cañabrava
construidas a la orilla de un río de aguas diáfanas que se precipitaban
por un lecho de piedras pulidas, blancas y enormes
como huevos prehistóricos.
```

Solución al segundo:

```
usuario@debv3:~/Documentos$ cat novela.txt
Muchos años después, frente al pelotón de fusilamiento,
el coronel Aureliano Buendía había de recordar
aquella tarde remota en que su padre lo llevó a conocer el hielo.
Macondo era entonces una aldea de 20 casas de barro y cañabrava
construidas a la orilla de un río de aguas diáfanas que se precipitaban
por un lecho de piedras pulidas, blancas y enormes
como huevos prehistóricos.
usuario@debv3:~/Documentos$ sed -e s/a/u/g -e s/e/u/g -e s/i/u/g -e s/o/u/g nove
la.txt
Muchus uñus duspués, fruntu ul pulutón du fusulumuuntu,
ul curunul Auruluunu Buundíu hubíu du rucurdur
uquullu turdu rumutu un quu su pudru lu lluvó u cunucur ul huulu.
Mucundu uru untuncus unu ulduu du 20 cusus du burru y cuñubruvu
cunstruudus u lu urullu du un ríu du uguus duáfunus quu su prucuputubun
pur un luchu du puudrus puludus, bluncus y unurmus
cumu huuvus pruhustórucus.
```

Solución al tercero:

```
usuario@debv3:~/Documentos$ sed 3q novela.txt
Muchos años después, frente al pelotón de fusilamiento,
el coronel Aureliano Buendía había de recordar
aquella tarde remota en que su padre lo llevó a conocer el hielo.
```

Solución al cuarto:

```
usuario@debv3:~/Documentos$ cat cancion.txt
Cuando Fernando Septimo usaba paleton
usuario@debv3:~/Documentos$ sed y/aeio/uuuu/ cancion.txt
Cuundu Furnundu Suptumu usubu pulutun
```

Un meta comando que se suele utilizar bastante en sed es el ampersand "&". Este comando es un comodín que se sustituye por todo lo que se está buscando. Lo veremos mejor con un ejemplo:

```
usuario@debv3:~/Documentos$ sed -e 1d -e 3,7d -e s/coronel/teniente/ novela.txt
el teniente Aureliano Buendía había de recordar
usuario@debv3:~/Documentos$ sed -n 2p novela.txt
el coronel Aureliano Buendía había de recordar
usuario@debv3:~/Documentos$ sed -e 1d -e 3,7d -e s/coronel/teniente/ novela.txt
el teniente Aureliano Buendía había de recordar
usuario@debv3:~/Documentos$ sed -e 1d -e 3,7d -e s/coronel/teniente\ \&/ novela.txt
el teniente coronel Aureliano Buendía había de recordar
```

En este ejemplo hemos protegido tanto el espacio en blanco como el propio ampersand con contrabarras ya que si no el sistema los interpreta como caracteres especiales e intenta ejecutarlos.

Una forma habitual de trabajar con sed es utilizando expresiones regulares. Este tipo de expresiones regulares no solo se usan con sed, sino con bastantes comandos de Linux, es por ello que vamos a detenernos para verlas un momento.

Estas expresiones regulares son bastante parecidas a los comodines que hemos visto anteriormente.

Carácter	Descripción de la expresión regular.
^	Apunta al comienzo de la línea
$	Apunta al final de la línea
.	Coincide con un único carácter cualquiera exceptuando líneas.
*****	Coincide con cero o más ocurrencias del carácter precedente
[lista]	Coincide con cualquier elemento de lista (deben ser caracteres)
[^lista]	Coincide con cualquier elemento que NO sea de la lista.
exp1|exp2	Coincide con exp1 O con exp2. (Se usa la \ para proteger el carácter |)
\número	Coincide con la sub expresión número. (Ahora veremos sub expresiones).
\n	Coincide con el retorno de línea.
\w	Coincide con cualquier palabra de la clase: [A-Za-z0-9_]
\s	Cualquier carácter de espaciado: espacio, tabulación horizontal o vertical
\S	Uno o varios caracteres de espaciado.
&	Este operador se sustituye por el patrón encontrado en el comando previo.

Algunos ejemplos con estas expresiones regulares, para los cuales usaremos este fichero:

```
usuario@debv3:~/iso$ cat novela.txt
Muchos años después, frente al pelotón de fusilamiento,
el coronel Aureliano Buendía había de recordar aquella tarde remota
en que su padre lo llevó a conocer el hielo.
Macondo era entonces una aldea de 20 casas de barro y cañabrava
construidas a la orilla de un río de aguas diáfanas
que se precipitaban por un lecho de piedras pulidas,
blancas y enormes como huevos prehistóricos.
El mundo era tan reciente, que muchas cosas carecían de nombre,
y para mencionarlas había que señalarlas con el dedo.
```

Sacar solo algunas líneas según su comienzo:

```
usuario@debv3:~/iso$ sed -n /^e/p novela.txt
el coronel Aureliano Buendía había de recordar aquella tarde remota
en que su padre lo llevó a conocer el hielo.
```

Sacar solo algunas líneas según su final:

```
usuario@debv3:~/iso$ sed -n /a$/p novela.txt
el coronel Aureliano Buendía había de recordar aquella tarde remota
Macondo era entonces una aldea de 20 casas de barro y cañabrava
```

Sacar las líneas cuyo segundo carácter sea una letra ele:

```
usuario@debv3:~/iso$ sed -n /^.l/p novela.txt
el coronel Aureliano Buendía había de recordar aquella tarde remota
blancas y enormes como huevos prehistóricos.
El mundo era tan reciente, que muchas cosas carecían de nombre,
```

Sacar las líneas donde aparezca algún digito:

```
usuario@debv3:~/iso$ sed -n /[0123456789]/p novela.txt
Macondo era entonces una aldea de 20 casas de barro y cañabrava
```

Sacar solo las líneas pares:

```
usuario@debv3:~/iso$ sed -n 1~2p novela.txt
Muchos años después, frente al pelotón de fusilamiento,
en que su padre lo llevó a conocer el hielo.
construidas a la orilla de un río de aguas diáfanas
blancas y enormes como huevos prehistóricos.
y para mencionarlas había que señalarlas con el dedo.
```

Este formato no lo habíamos visto anteriormente. Cuando queremos indicar un rango de líneas, si utilizamos ~ le indicamos en que línea queremos empezar, y el incremento salto de línea que queremos que use el rango.

Añadir al principio de cada línea tres guiones:

```
usuario@debv3:~/iso$ sed s/^../---\&/ novela.txt
---Muchos años después, frente al pelotón de fusilamiento,
---el coronel Aureliano Buendía había de recordar aquella tarde remota
---en que su padre lo llevó a conocer el hielo.
---Macondo era entonces una aldea de 20 casas de barro y cañabrava
---construidas a la orilla de un río de aguas diáfanas
---que se precipitaban por un lecho de piedras pulidas,
---blancas y enormes como huevos prehistóricos.
---El mundo era tan reciente, que muchas cosas carecían de nombre,
---y para mencionarlas había que señalarlas con el dedo.
```

Expliquemos ahora las sub expresiones. Estas consisten en numerar los patrones reconocidos de modo que puedan ser reemplazados en distinto orden. Simplemente tenemos que encerrar los patrones que deseemos entre paréntesis:

```
usuario@debv3:~/apuntes$ cat nombres.txt
jose
juan
usuario@debv3:~/apuntes$ sed -r 's/(j)/--\1--/g' nombres.txt
--j--ose
--j--uan
usuario@debv3:~/apuntes$
```

En este ejemplo anterior el patrón sustituido (j) es referenciado luego con \1, ya que es el primer patrón sustituido. Si tuviéramos más patrones sustituidos se irían numerando 1,2,3…

Clases de caracteres que se pueden utilizar en las expresiones regulares:

lista	Descripción
[:alnum:]	caracteres alfanuméricos [A-Za-z0-9]
[:alpha:]	caracteres alfabéticos [A-Za-z]
[:digit:]	cifras [0-9]
[:lower:]	caracteres en minúsculas [a-z]
[:upper:]	caracteres en mayúsculas [A-Z]
[:print:]	caracteres imprimibles [-~]
[:punct:]	caracteres de puntuación [!-/:-@[-`{-~]
[:space:]	espacios, tabulaciones y cualquier carácter vacío [\t\v\f]
[:blank:]	espacio y tabulación [\x09]
\<, \>	Inicio, fin de palabra
\b	Posición entre palabras
\B	Posición en medio de una palabra

Un ejemplo de sed utilizando una lista de clase de caracteres:

```
usuario@debv3:~/Documentos$ cat dos.txt
Susanita tiene un raton
un raton chiquitin
usuario@debv3:~/Documentos$ sed s/[[:lower:]]/x/g uno.txt
Cxx xxxx xxxxx xxx xxxxx
xxxxxx xx xxxx x xxxx xxxx
xx xxxxx xx xxx xxxx xxxxx
xx xxxxxx xxxxxxxxx.
```

Un ejemplo de sed que nos permite eliminar caracteres repetidos de un fichero:

```
usuario@debv3:~/iso$ cat saludo.txt
HHEELLLL00
usuario@debv3:~/iso$ sed 's/\(.\)\1/\1/g' saludo.txt
HELL0
```

Todos los ejemplos hasta ahora no han modificado los ficheros, sino que nos ha mostrado por pantalla el resultado del comando sed.

Veamos ahora un par de ejemplos en los que modificamos el fichero directamente.

```
usuario@debv3:~/iso$ sed -n 1p novela.txt
Muchos años después, frente al pelotón de fusilamiento,
usuario@debv3:~/iso$ sed -i s/,$//g novela.txt
usuario@debv3:~/iso$ sed -n 1p novela.txt
Muchos años después, frente al pelotón de fusilamiento
```

```
usuario@debv3:~/iso$ cat telefonos.txt
956765443
856653112
678991192
902123423
usuario@debv3:~/iso$ sed -i 's/^[[:digit:]][[:digit:]][[:digit:]]/(&)/g' telefonos.txt
usuario@debv3:~/iso$
usuario@debv3:~/iso$ cat telefonos.txt
(956)765443
(856)653112
(678)991192
(902)123423
```

Algunos ejercicios sobre sed.

1) Mostrar el fichero /etc/passwd pero eliminando la primera línea.
2) Mostrar el fichero /etc/idmapd.conf pero eliminando todas las líneas de comentarios (comienzan con un #).
3) Mostrar el fichero /etc/idmpad.conf pero eliminando todos los espacios en blanco
4) Mostrar el fichero /etc/passwd pero eliminando todas las líneas donde aparezca el texto *home*
5) Mostrar el fichero /etc/passwd pero comentando todas las líneas donde aparezca el texto *false*
6) Mostrar el fichero /etc/passwd pero sustituyendo el tercer punto y coma por un guion.
7) Mostrar un fichero de texto por pantalla, pero solo las líneas donde aparezca una dirección de correo electrónico.
8) Mostrar un fichero de texto por pantalla, pero solo las líneas donde aparezca una fecha en formato dd-mm-aaaa.
9) Mostrar un fichero de texto por pantalla, pero modificándolo de tal forma que si existen varios espacios en blanco seguido se sustituyan por un solo espacio en blanco.
10) Mostrar un fichero por pantalla, pero asegurándose de que el primer carácter de cada línea está en mayúsculas.

COMANDO TR

Es un comando utilizado para traducir caracteres. Nos permite cambiar unos caracteres por otros, eliminar caracteres, etc.

La sintaxis de la orden es la siguiente:

```
tr [opciones] cadena1 cadena2
```

Si no utilizamos ninguna opción, tr se limita a sustituir todos los caracteres de cadena1 por los caracteres de cadena2 que estén en el mismo lugar.

Este comando desarrolla la misma función que la función –y que hemos visto anteriormente en el sed. Este comando tr está pensado para trabajar sobre flujos de datos, utilizando tuberías (pipes) y no permite su uso sobre ficheros directamente. Es por esto que cuando veamos tuberías veremos algunos ejemplos de este comando.

```
echo canasta de 3 puntos | tr [3] [6]              devuelve      canasta de 6 puntos
echo canasta de 3 puntos | tr [123] [678]          devuelve      canasta de 8 puntos
echo canasta de 3 puntos | tr [a3] [j6]            devuelve      cjnjstj de 6 puntos
```

Una opción de tr es d, que nos permite borrar los caracteres encontrados de cadena1. En este caso no hace falta poner nada en cadena2.

```
echo hola mundo proceloso y bello | tr –d [“ “]
```

Esta orden anterior nos eliminaría los espacios en blanco.

La opción s nos permite eliminar caracteres repetidos.

```
echo aaaddddiiiioooossss | tr -s [ao]  esto nos devuelve  addddiiiiossss
```

La opción –c permite complementar otra opción, de modo que se realiza lo contrario de lo indicado. Así, si añadimos –c a la orden anterior obtendríamos lo siguiente:

```
echo Tengo 23 años y 4 meses | tr -cd [:digit:]esto nos devuelve 234
```

Otra de las cadenas reconocidas por tr es :print: y nos es muy útil, ya que representa caracteres imprimibles, es decir, caracteres que se pueden utilizar correctamente en el sistema.

```
echo camión cataluña | tr -cd [:print:]esto nos devuelve  comn camin catalua
```

COMANDOS PARA COMPACTAR Y AGRUPAR FICHEROS

COMANDOS GZIP Y GUNZIP

Los comandos gzip y gunzip permiten compactar y descompactar (comprimir y descomprimir) respectivamente uno o varios ficheros.

Sintaxis:

```
gzip        [opciones] <ficheros/directorio>
gunzip      [opciones] <ficheros/directorio>
```

Algunas opciones:

-r: dado un directorio comprime todos los ficheros presentes en él recursivamente.
-1 a -9: especifica el grado de la compresión (-1 menor y más rápida -9 mayor y más lenta).
-S <sufijo>: permite especificar un sufijo o extensión para el fichero resultado (por defecto es gz).

Ejemplos:

```
$ gzip -9 big-file
$ gunzip big-file.gz
$ gzip -S .zip -r doc/
$ gunzip -S .zip -r doc/
```

Fijaros como este comando no funciona como sus análogos de Windows a los que estamos acostumbrados, ya que si indicamos que queremos comprimir 10 ficheros, obtendremos 10 ficheros comprimidos, no un solo fichero comprimido que contenga a los 10 ficheros originales. Esta compactación de ficheros (de muchos a uno) se consigue agrupándolos con el comando tar.

COMANDO TAR

El comando tar (Tape Archiver) es una herramienta para agrupar varios ficheros aislados o el contenido de un directorio en otro fichero o dispositivo especial. El comando tar no comprime o compacta absolutamente nada, se limita a agrupar varios ficheros en uno solo, sin comprimirlos. Existe una opción (-z) que automáticamente ejecuta un gzip o un gunzip sobre el fichero agrupado.

Sintaxis: `tar [opciones] <fuentes>`

Algunas opciones:

-c	permite crear (tarear).
-x	permite extraer (destarear).
-v	activa el modo debug, donde se ven todos los mensajes.
-f <fichero>	agrupa o desagrupa en o hacia un fichero y no utilizando la salida o entrada estándar.
-z	compacta o descompacta el fichero resultante una vez agrupado o desagrupado con gzip y gunzip respectivamente.
-t	lista el contenido de un fichero resultado de un agrupamiento.
-M	agrupa en volúmenes.

El comando tar conserva la estructura jerárquica original de lo agrupado excluyendo el carácter / que representa a la raíz. Algunas opciones se pueden emplear sin el carácter -, siempre y cuando no haya ambigüedades entre ellas o con los argumentos.

Ejemplos:

```
tar cvfz /tmp/etc.tgz /etc
tar xvfz /tmp/etc.tgz
tar cf uconf.tar passwd shadow groups
tar xf uconf.tar
tar cM -f /dev/fd0 /tmp/etc.tgz
tar xM -f /dev/fd0
```

COMANDOS PARA LA COMUNICACIÓN ENTRE USUARIOS

COMANDO WRITE

El comando write se utiliza para enviar un mensaje a un usuario conectado al sistema. Por defecto el mensaje se envía a la última terminal donde se haya conectado el usuario. Los usuarios pueden deshabilitar la posibilidad de recibir mensajes utilizando el comando mesg.

Sintaxis: `write <usuario> [terminal]`

Ejemplos:

```
mesg  y                 # habilita la posibilidad de recibir mensajes
write pepe tty3
   Hola que tal
   Ctrl-d
```

Si el usuario pepe está conectado a través de la terminal tty3 y tiene habilitada la posibilidad de recibir mensajes se mostrará en esta terminal:

```
Message from coco@deltha on tty2 at 16:35 ... (o el usuario que sea)
    Hola que tal
    EOF
```

COMANDO WALL

El comando wall se emplea para enviar un mensaje a todos los usuarios conectados en el sistema que tengan habilitada la posibilidad de recibirlos (`mesg y`).

Ejemplos:

```
wall
         Voy a apagar la máquina a
         las 2:00 PM
               El administrador
        Ctrl-d
```

COMANDOS PARA DESCONECTARSE DEL SISTEMA

COMANDO EXIT

El comando exit permite terminar el shell actual. Si se tiene un único shell es equivalente a desconectarse del sistema, pero si se está en un subshell sólo se terminará este, retornando al shell anterior.

COMANDO LOGOUT

El comando logout permite desconectarse del sistema a partir de un login shell (primer shell que se ejecuta al establecer la conexión).

La secuencia de caracteres Ctrl-d permite terminar el shell actual. Si es un login shell equivaldrá a hacer logout y si no, a hacer exit.

COMANDO SHUTDOWN

El comando shutdown en realidad no sirve para desconectarse del sistema, sino para apagarlo totalmente.

Ejemplos:

```
shutdown –h now   Apaga el sistema (-halt) ahora (now)
shutdown -h 18:45 "El servidor se apaga para cambiar gráfica"
shutdown –r –g5   El sistema se reiniciará (-r) en 5 minutos (-g5)
```

COMANDOS VARIOS.

COMANDO ALIAS

El comando alias permite asignarle otros nombres a los comandos. De esta forma se pueden abreviar o llamarlos de forma más nemotécnica.

Sintaxis: `alias [nombre[=valor]...]`

Sin argumentos muestra todos los alias definidos por el usuario actual. Para deshabilitar un alias se emplea el comando `unalias`.

Ejemplos:

```
alias l='ls -l --color'
alias l
alias l='ls -l --color'
alias
alias c='clear'
alias l='ls -l --color'
alias l.='ls .[a-zA-Z]* --color=tty'
alias la='ls -al --color'
```

COMANDO TTY

El comando tty imprime el dispositivo de carácter asociado a la terminal en la que se está trabajando.

Ejemplos:

```
tty                 /dev/tty2
tty                 /dev/pts/0
```

COMANDO DU

El comando du permite conocer la longitud (expresada en kilobytes por defecto) de una jerarquía de ficheros a partir de un directorio.

Sintaxis: `du [opciones] [ficheros | directorios]`

Algunas opciones:

```
-h : (human readable view) imprime las unidades de la forma más representativa
-s : sumariza el tamaño de cada fichero/directorio sin profundizar recursivamente
-c : produce un total cuando se utilizan varios argumentos.
```

Ejemplos:

```
du -h *
du -hsc /usr /home
```

COMANDO WHO

El comando who muestra los usuarios conectados al sistema ya sea local o remotamente.

Sintaxis: `who [opciones] [fichero] [am i]`

Sin argumentos who muestra los logins de los usuarios conectados, porque terminal lo han hecho y en qué fecha y hora.

Algunas opciones:

```
-H : imprime un encabezamiento para las columnas.
-w : indica si está activada o no la posibilidad de recibir mensajes
por parte de cada usuario conectado (+ indica que si, - que no y ?,
desconocido).
-i : imprime además para cada usuario conectado que tiempo lleva sin
interactuar con el sistema (idle time). Si lleva menos de un minuto pone
un punto y si es más de 24 horas la cadena ``old''.
-q : sólo muestra los logins de los usuarios conectados y la cantidad
total de ellos.
```

COMANDO W

El comando w muestra también los usuarios conectados al sistema además de lo que están haciendo (proceso que ejecutan en ese momento) y otras informaciones.

Sintaxis: `w [opciones] [usuario]`

Sin argumentos este comando muestra una primera línea con: la hora en que arrancó el sistema, cuánto tiempo lleva funcionando, cuantos usuarios hay conectados (sin incluir las sesiones gráficas) y tres porcientos de carga de la CPU: durante el último, los 5 y los 15 minutos anteriores. A continuación se muestra una tabla cuyas columnas representan: el login de cada usuario conectado, porque terminal lo ha hecho, desde que host, a qué hora, el idle time exacto, la cantidad de segundos de CPU que han empleado todos los procesos que ha ejecutado ese usuario (JCPU) y el tiempo (PCPU) y nombre del comando que ejecuta actualmente.

COMANDO FINGER

El comando finger permite buscar y mostrar información asociada a los usuarios del sistema de acuerdo a sus nombres, apellidos o login. La información que muestra finger para cada usuario es:

El login.
El nombre y los apellidos.
El directorio base.
El shell.
La oficina y el teléfono.
El teléfono de la casa.
La lista de terminales a través de las que está conectado con la fecha, tiempo sin interactuar (iddle time) y si está deshabilitada la posibilidad de recibir mensajes.
La fecha y hora del último nuevo mensaje electrónico recibido y desde cuando no accede al buzón.
El contenido del fichero .plan en el directorio base.

COMANDO HOSTNAME

Nos devuelve el nombre de nuestro host o máquina.

COMANDO TOP

Nos permite ver un listado de los procesos que están corriendo en nuestra máquina y los recursos que están utilizando.

COMANDO DATE

Nos devuelve la hora y la fecha del sistema.

COMANDO CAL

Nos da el calendario del mes actual.

COMANDO CLEAR

Borra la pantalla.

COMANDO UNAME.

Nos da información sobre nuestra máquina. (SO, arquitectura, etc.).

COMANDOS DE RED

COMANDO PING

El comando ping permite enviar paquetes ICMP (Internet Control Message Protocol) del tipo ECHO_REQUEST a otra computadora, con el objetivo de saber si esta es alcanzable a través de la red. Además muestra un resumen estadístico acerca del porciento de paquetes perdidos y las velocidades de trasmisión. Este comando se ejecuta por defecto sostenidamente por lo que para interrumpirlo se debe hacer Ctrl-c.

Sintaxis: `ping [opciones] <máquina>`

Algunas opciones:

```
-c <n>      envía n paquetes exactamente.
-i <n>      espera n segundos entre los envíos.
-s <n>      envía paquetes de n bytes.
-q          sólo despliega el sumario final.
```

COMANDO IFCONFIG

El comando ifconfig permite configurar por parte de root las interfaces de red. Los usuarios distintos de root lo pueden invocar también con fines informativos. Para ello deben especificar el camino completo (/sbin/ifconfig) pues por defecto este no está en el path de los usuarios comunes. Sin argumento ifconfig despliega información acerca de la configuración y funcionamiento actuales de las interfaces de red activas.

La orden ifconfig realmente modifica el fichero /etc/networking/interfaces. Podemos editar directamente ese fichero para modificar las propiedades de la red. Para reiniciar todo el sistema de red y volver a cargar dicho fichero, podemos o bien reiniciar totalmente la maquina con un shutdown –r now o bien podemos reiniciar únicamente el entorno de red con la orden /etc/init.d/networking restart.

A continuación veremos algunos ejemplos del uso de ifconfig.

Ver la configuración de red de un adaptador de red: ifconfig, invocado sin argumentos mostrará el detalle de todas las interfaces activas. Si como argumento pasamos el nombre de una interfaz, veremos los detalles específicos de una interfaz.

```
ifconfig
ifconfig eth0
```

Ver un detalle de todas las interfaces (incluidas las deshabilitadas):

```
ifconfig -a
```

Deshabilitar una interfaz:

```
ifconfig eth0 down
```

Habilitar una interfaz:

```
ifconfig eth0 up
```

Asignar una dirección IP a una interfaz:

```
ifconfig eth0 192.168.0.2
```

Cambiar la máscara de subred:

```
ifconfig eth0 netmask 255.255.255.0
```

Cambiar la dirección de broadcast:

```
ifconfig eth0 broadcast 192.168.0.255
```

Asignar dirección IP, máscara y broadcast al mismo tiempo:

```
ifconfig eth 0192.168.0.2 netmask 255.255.255.0 broadcast 192.168.0.255
```

Cambiar el MTU (unidad máxima de transmisión)

```
ifconfig eth0 mtu XX
```

Activar modo promiscuo: Por defecto cuando una tarjeta de red recibe un paquete comprueba si dicho paquete le pertenece y si no lo es lo descarta. En modo promiscuo, la tarjeta no descarta ese paquete y acepta todos los paquetes. El modo promiscuo se utiliza especialmente para capturar y analizar el tráfico de una red.

```
ifconfig eth0 promisc
```

Para devolverla a su modo normal

```
ifconfig eth0 -promisc
```

Todos estos cambios que realizamos con ifconfig no son permanentes, sino que se pierden cuando reiniciamos la máquina. Para hacer dichos cambios permanentes hay que introducirlos en el fichero /etc/network/interfaces.

COMANDO ROUTE

Este comando nos permite gestionar las tablas de enrutamiento que utiliza nuestro sistema.

```
usuario@debv3:~/iso$ su -c "route"
Contraseña:
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         192.168.127.2   0.0.0.0         UG    0      0        0 eth0
192.168.127.0   *               255.255.255.0   U     0      0        0 eth0
```

Este comando no solo nos permite ver las rutas, sino también modificarlas. Evidentemente tenemos que saber muy bien que es lo que estamos haciendo ya que en caso contrario lo más normal es que nos quedemos sin red.

COMANDO IP

Este comando, mucho más reciente que los comandos de red vistos anteriormente, se supone que viene para sustituirlos ya que permite realizar sus mismas funciones y algunas más. Vamos a ver unas pocas posibilidades de este comando, que tiene muchísimas más como podéis comprobar con un man ip.

Por ejemplo, para comprobar la configuración actual de nuestros interfaces de red (lo que hace ifconfig por defecto) escribiríamos el siguiente comando: `ip addr list`

```
usuario@debv3:~/iso$ ip addr list
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:0c:29:2a:9d:31 brd ff:ff:ff:ff:ff:ff
    inet 192.168.127.135/24 brd 192.168.127.255 scope global eth0
    inet6 fe80::20c:29ff:fe2a:9d31/64 scope link
       valid_lft forever preferred_lft forever
```

También podemos usar el comando `ip link show` para ver la información en capa 2 (data link layer) de las interfaces de red del sistema:

```
usuario@debv3:~/iso$ su -c "ip link set eth0 down"
Contraseña:
usuario@debv3:~/iso$ su -c "ip link set eth0 up"
Contraseña:
```

Podemos activar o desactivar interfaces de red con `ip link set`

```
usuario@debv3:~/iso$ ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN mode DEFAULT
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT qlen 1
000
    link/ether 00:0c:29:2a:9d:31 brd ff:ff:ff:ff:ff:ff
```

Con ip link podemos establecer muchas configuraciones del interfaz. Así, por ejemplo, para establecer el modo promiscuo usamos el comando:

```
ip link set dev eth0 promisc on
```

Con ip addr add podemos especificar la IP, máscara y la IP de broadcast:

```
ip addr add 10.0.0.100/24 broadcast 10.0.0.255 dev eth2
```

Y para eliminar la IP:

```
ip addr del 10.0.0.100/24 dev eth2
```

Podemos ver la tabla de rutas con `ip route show:`

```
usuario@debv3:~/iso$ ip route show
default via 192.168.127.2 dev eth0  proto static
192.168.127.0/24 dev eth0  proto kernel  scope link  src 192.168.127.135
```

Podemos ver la tabla ARP con `ip neighbor show`:

```
usuario@debv3:~/iso$ ip neighbor show
192.168.127.2 dev eth0 lladdr 00:50:56:f1:8a:34 STALE
```