2. SISTEMA DE FICHEROS

El sistema de ficheros es parecido al UFS (Unix File System), por lo que sólo existe un árbol y por tanto sólo tiene una raíz ("/"). El sistema siempre diferencia entre mayúsculas y minúsculas y los ficheros no tienen extensión. Los ficheros y los directorios son lo mismo (ambos tienen los mismos metadatos).

El sistema de ficheros está basado en el concepto de i-nodo. 2.1.I-nodo. Los i-nodos hacen referencia a una estructura de datos, una especie de ficha con diferentes campos (por ejemplo, el bloque de control, los metadatos del fichero...). El i-nodo almacena los metadatos de cada fichero existiendo un i-nodo por cada directorio. Guardan una lista con la localización de los bloques de disco donde está guardado cada fichero. Estos bloques se gestionan en grupos, reduciendo la fragmentación de los ficheros. A través de la ruta de los ficheros llegamos a su nombre que tendrá asociado un número de inodo; dicho número estará recogido en la tabla de i-nodos. Esta tabla se carga durante el arranque. La tabla se almacena en la primera parte del disco y se carga en memoria (pudiendo cargarse por bloques).

Cada tabla empareja nombres y números de inodos. Por ejemplo, si en el directorio raíz tuviéramos la siguiente tabla (el inodo de la raíz no hay que buscarlo, está en lugar conocido):

Directorios	i-nodo
.	1
..	1
bin	3
dev	9
lib	15
Etc	10

Directorios i-nodo . 1 .. 1 bin 3 dev 9 lib 15 Etc 10 En esta tabla los directorios "." y ". ." apuntan a la raíz, por lo que comparten el mismo inodo. Si quiero leer "/bin/ls", voy a raíz y "/bin" está en el inodo 3. Voy a la tabla de inodos y en el inodo 3 están los metadatos de "/bin", donde puedo leer que se encuentra en el bloque 204.

Ficheros	i-nodo
/bin/ nano	8
/bin/ls	4

Leo dicho bloque y tengo otra tabla de directorios de "/bin". Ficheros i-nodo /bin/nano 8 /bin/ls 4 Se accede al inodo 4 y tenemos todos los metadatos de "/bin/ls" y nos dice que se encuentra en los bloques 125 y 126, y allí encontramos el fichero. Existe una caché de bloques para que no tener que acceder siempre al disco. Dentro de un fichero, cada bloque apunta a su siguiente. No se requiere consultar continuamente la Tabla de i- nodos. Debido a la gestión de los i-nodos los bloques tienden a almacenarse cercanos y en grupos lo que reduce la fragmentación.

2.2.Tipos de ficheros

Existen diferentes tipos de ficheros que Linux identifica con un carácter.

Utiliza un "-" para los ficheros regulares.
Una "d" para los directorios (Linux los muestra de color azul).
Una "b" o "c" para los dispositivos.
Una "l" para los enlaces SIMBÓLICOS o blandos.

En ellos el contenido del bloque del fichero enlace es la ruta del fichero enlazado, sería lo que son los accesos directos en Windows. Se crean con "**ln -s**" (de symbolic). El fichero enlazado guarda la ruta de acceso del fichero al que está enlazado, por lo que, si borro este fichero, el enlazado no valdrá para nada.

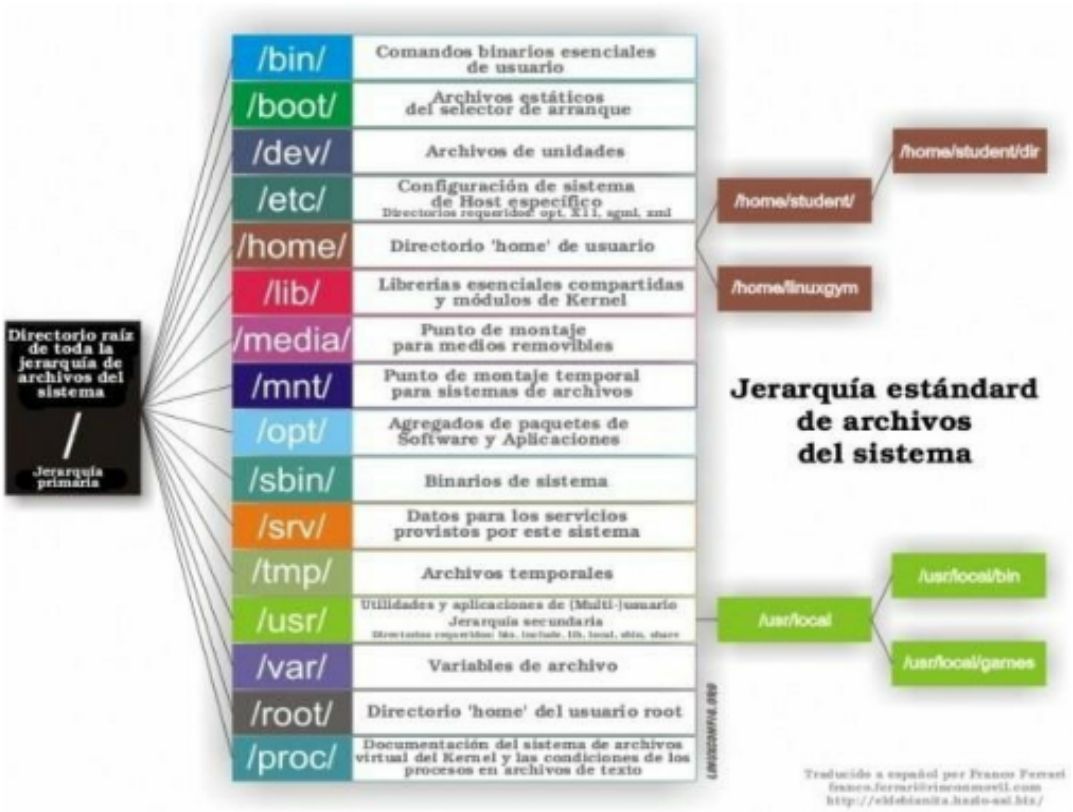
La sintaxis será la siguiente. Si tengo un archivo, por ejemplo "Notas_Trim2014", y quiero mostrarlo con un acceso directo a "n014", escribo: ~$ ln -s Notas_Trim2014 n014 De esta forma, cuando haga un ls, me aparecerá: lrw-rw-r-- alumno alumno ... n014 •Un "-" para los enlaces FÍSICOS o duros. En los enlaces físicos un mismo inodo está referenciado desde dos ficheros diferentes. Los enlaces son la forma de crear accesos a ficheros en diferentes lugares. Para los enlaces FÍSICOS se crea un nombre de ficheros que apunte a un inodo ya referenciado. Por ejemplo, si "ls" apunta al inodo 5, y creo "lista" que hace referencia a ese mismo inodo, tendré un fichero con dos nombres (en Linux el número de ficheros que hay es el número de inodos existentes). No se pueden crear enlaces físicos a directorios. Los enlaces se crean con el comando "ln" (link). Por ejemplo, si tengo el fichero "/tmp/hola", y quiero enlazarlo con un fichero que aún no existe llamado "adios", para que "adios" apunte al mismo inodo que "hola", escribiré: ~$ ln /tmp/hola /tmp/adios Si ejecutamos "ls -i" nos devolverá el número de inodo que corresponde a ese nombre. Y los dos ficheros aparecerán con un fondo azul: 432455 hola432455 adios Siguiendo con el ejemplo anterior, si se escribe: ~$ echo saludo > /tmp/hola Volcará en el fichero "hola" el contenido "saludo".

Automáticamente, al estar "hola" y "adios" enlazados, "saludo" también estará en "adios". Si ahora se escribe: ~$ rm /tmp/hola

Eliminaré "hola", pero no ese inodo, pues el sistema de ficheros no libera un inodo hasta que no se han eliminado todos los nombres que tiene referenciado. Para liberar ese inodo habrá que escribir también: ~$ rm /tmp/adios Para ver los metadatos del fichero y los nombres que apuntan a cada inodo se utiliza "ls -l".

3. ESTRUCTURA DE DIRECTORIOS EN LINUX

Es “muy importante” conocer la estructura de directorios en Linux, ya que ello nos permite saber qué tipo de información contiene. Esto nos ayudará a tener una mejor visión de cómo está organizado el sistema operativo GNU/Linux, además con ello sabremos dónde localizar lo que necesitamos. En esta imagen podemos ver una breve descripción de la estructura de directorios de linux:



/: (Raíz) Es el nivel más alto dentro de la jerarquía de directorios (PRINCIPAL). De aquí parten los demás directorios y subdirectorios.

/bin: (Binarios) Contiene los binarios básicos, que son los ejecutables del sistema operativo.

/boot: (Arranque) núcleo y ficheros necesarios para cargar el núcleo y ficheros de configuración del gestor de arranque.

/dev: (Dispositivos) En esta carpeta se encuentran todos los archivos que nos permiten interactuar con los dispositivos hardware de nuestro ordenador. Por ejemplo, los usb, sda (o hda) con la información de cada uno de ellos.
/etc: (Etcétera) Aquí se guardan los ficheros de configuración del sistema (y del sw instalado).

/home: (Hogar) Contiene las carpetas por defecto de los usuarios, (sería algo así como el "Documents and Settings" del sistema operativo Windows).

/lib: (Bibliotecas) Contiene las librerías del sistema y los drivers. Librerías compartidas.

/lost+found: (Perdidos y encontrados) localizaciones de disco marcadas como en uso en las estructuras de datos del disco, pero no están listadas en ningún directorio (contiene "referencias" a los ficheros marcados como erróneos al chequear el sistema de ficheros).

/media: (Media/medios) Directorio donde puede ser utilizado como punto de montaje para las Unidades Extraíbles. Por ejemplo, los dispositivos USB, disqueteras, unidades de CD/DVD.

/mnt: (Montaje) Es un directorio que se suele usar para montajes temporales de unidades. Por ejemplo, Directorios compartidos dentro de una red, alguna partición de Windows, etc.

/proc: (Procesos) Información de los datos del kernel del sistema de ficheros de Linux, Virtualización y procesos en ejecución.

/sys: (Sistema) Información sobre los dispositivos tal y como los ve el kernel Linux.

/root: (Casa del Administrador) Es el directorio /home del administrador. Es el único /home que no está incluido -por defecto- en el directorio HOME.

/sbin (Binarios del Sistema): Son los ejecutables de administración (que normalmente sólo el administrador puede ejecutar), tales como mount, umount, etc.

/srv: (Servicios) En este directorio residen las carpetas accesibles por el programa cliente de un determinado servicio ofrecido por algunos servidores configurados en el sistema. Por ejemplo, Apache, ProFtpd, etc.

/tmp: (Temporales) Es un directorio donde se almacenan ficheros temporales. Cada vez que se inicia el sistema, son borrados.

/usr: (Compartido de Usuarios) Es el directorio padre de otros subdirectorios de importancia. Se encuentran la mayoría de los archivos del sistema, aplicaciones, librerías, manuales, juegos… Es un espacio compartido por todos los usuarios del sistema.

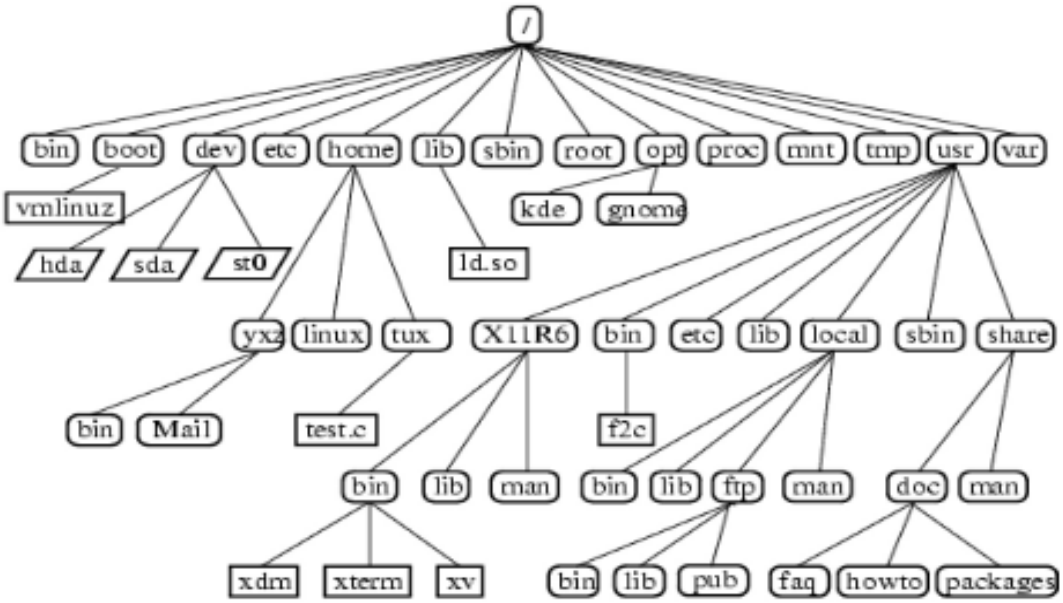
/usr/bin ficheros ejecutables por todos los usuarios

/usr/sbin ficheros ejecutables de administración

/usr/include ficheros cabecera de C

/usr/lib librerías /usr/local software local
/usr/share datos compartidos (o que pueden ser compartidos por varios ordenadores en red)
    /usr/share/man
    /usr/share/doc
/usr/src código fuente, como el del kernel

/var: (Variables) Ficheros y datos que cambian frecuentemente (logs, bases de datos, colas de impresión…) Estos son subdirectorios de de spool (impresión, e-mail, cron, atd, log, etc.) Esto constituye una breve descripción de los directorios principales, lo cual no quiere decir que dentro no haya subdirectorios. Algo un poco más completo:



Dentro de la categorización de los directorios encontramos:

Estáticos: Contiene archivos binarios, bibliotecas, y otros archivos están en Read Only (Solo lectura) que no cambian sin la intervención del administrador. /bin, /sbin, /boot, /opt.

Dinámicos: Son aquellos que los archivos dentro de estos van cambiando. Generalmente se encuentra como Read-Write (Lectura-Escritura). /var/spool, /var/lock, /var/mail, /home.

Compatibles: Se pueden encontrar archivos comunes que van a estar en cualquier distribución. /usr/bin, /opt

No Compatibles: Contiene archivos que no son compatibles con otras distribuciones. /etc, /boot, /var/run, /var/lock.

4. 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 a partir 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 < 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.

Instrucción &> redirecciona ambas salidas. Ejemplos:

#cat /etc/shadow > cont	Se redirecciona la salida estándar hacia un fichero con nombre cont
$write pepe < msj.txt	Toma la entrada en lugar de desde el teclado, desde el fichero msj.txt
$cat > teléfonos	Toma de la entrada estándar (teclado) y lo envía a la salida que será el fichero teléfonos. (Terminar la entrada con **Ctrl D**).
$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.
$find / -iname "carta" > sa 2> err	Ejecuta el comando find, mandando la salida normal de la orden al fichero sa y mandando la salida de errores al fichero err.

Siempre que se emplee la forma [x]>salida, si el fichero salida existe se sobrescribirá y si no, se creará. Para añadirle algo más a su contenido anterior (append) en lugar de sobrescribirlo, se puede emplear >>

$echo "Hola" >> fichero	Se redirecciona el texto Hola hacia fichero. Si fichero ya existe se añadirá el texto al final, si no

5. SUPERUSUARIO O ADMINISTRADOR

El administrador es el usuario que tiene siempre todos los privilegios sobre cualquier fichero, instrucción u orden del sistema.

En Linux es el usuario root que pertenece al grupo root HOME: /root (en modo monousuario es /)

Convertirse en administrador:

O Entrar al sistema como usuario root
O Ejecutar orden **su** → pide la contraseña del root y lanza una shell como root

En Windows es el usuario Administrador (o Administrador de dominio) que pertenece al Grupo administrador Convertirse en administrador: entrar al sistema con el usuario administrador.

Herramienta: **sudo**
Permite a otros usuarios ejecutar órdenes como si fuese el administrador

/etc/sudoers fichero de configuración
visudo orden para modificar el fichero de configuración /etc/sudoers
sudo orden→ pide contraseña del usuario

Comunicarse con los usuarios:
Write enviar un mensaje a un usuario
talk conversar con un usuario, incluso aunque esté en otra máquina Linux/Unix
mesg habilitar/deshabilitar la llegada de mensajes al terminal

Fichero /etc/motd → contiene el mensaje del día que se imprime justo después de entrar al sistema. (En modo texto)

Fichero $HOME/.hushlogin permite evitar el mensaje del día

Fichero /etc/issue → contiene el mensaje que se muestra antes del login, normalmente muestra las versiones de Linux y núcleo instaladas. (En modo texto)

INTRODUCCIÓN COMANDOS LINUX

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 de texto conocida como consola o terminal. Linux ofrece el mecanismo de consolas o terminales virtuales, consiste en que a partir de una entrada (el teclado) y con una salida (el monitor) se simulan 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 ellas. 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). Desde una consola gráfica para cambiar a otro 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. 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.). El 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.

Tipos de shell

Existe una gran variedad de shells. Las diferencias entre unas y otras radican en algunas características tales como la velocidad de ejecución, los atajos de teclado, el autocompletado de nombres, el manejo del historial de órdenes, la capacidad de evaluación aritmética y muchos otros detalles sutiles. La shell más común en sistemas Linux es la shell Bash, pero en nuestro sistema podemos disponer de otras shells o instalarlas a voluntad

desde los repositorios de nuestra distribución. Entre las más conocidas encontramos las siguientes:

bash Bourne again shell (bash) es un intérprete de ordenes basado en la shell de Unix, desarrollado para el proyecto GNU. Es el intérprete de comandos predeterminado en la mayoría de las distribuciones de Linux. Su nombre hace un juego de palabras con referencia a la Bourne shell, que fue el primer intérprete de comandos importante de los sistemas Unix.

sh Bourne Shell (sh) puede ser considerada "la madre de todas las shells", pues fue el intérprete de comandos más popular de los sistemas Unix. Incorporó diferentes mejoras al Thompson shell, cuyo ejecutable tenía el mismo nombre: sh. En muchos sistemas Linux, sh es un enlace simbólico a algún otro shell como bash o dash.

dash Debian Almquist shell (dash) es un intérprete de comandos más ligero y rápido que bash, aunque con menores funcionalidades. Está basado en ash, un shell muy popular en los sistemas BSD (un sistema operativo derivado de Unix, muy parecido a Linux).

zsh Z shell (zsh) es uno de los más recientes intérpretes de comandos para Unix, Linux y BSD. Orientado principalmente a su uso interactivo, más que programático. Está basado en bash, pero ha incorporado abundantes características de otras shells y bastantes funcionalidades originales.

Korn shell (ksh) es otro intérprete de comandos basado en el Bourne Shell, manteniendo plena compatibilidad con éste, pero incorporando sustanciales mejoras en su orientación a su uso como lenguaje de programación. Es el shell por defecto en los sistemas AIX (Unix de IBM).

Algunas características que merece la pena conocer de bash, shell que vamos a utilizar son:

Autocompletar 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 (echo HISTSIZE), pero puede modificarse (vi ~/.bash_profile, indicando HISTSIZE=X y HISTFILESIZE=X, siendo X el número de comandos que se desea que el sistema almacene)

Poderosas estructuras de control para realizar scripts. (Procesos por lotes). Se pueden utilizar instrucciones if, for, while, select, case, etc.

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 abreviada.

¿Cómo conocer o modificar la shell activa?

En el terminal de Linux disponemos de algunas instrucciones que nos permitirán conocer la shell que estamos utilizando, los diferentes tipos de shell disponibles y cambiar el tipo de shell activa. En los siguientes ejemplos, pulsando sobre cada una de las opciones veremos los comandos que podemos utilizar para obtener esta información.

¿Cómo identificar la shell activa?

Mediante el siguiente comando conoceremos la shell activa:

```
$ echo $SHELL
/bin/bash
```

¿Cómo conocer las shells disponibles?

Este otro comando nos mostrará las diferentes shells disponibles:

```
$ cat /etc/shells
# /etc/shells: valid login shells
/bin/sh
/bin/dash
/bin/bash
/bin/rbash
$
```

¿Cómo podemos modificar la shell activa?

Para modificar la shell activa a, por ejemplo, dash (o alguna otra de las listadas en /etc/shells).

Podemos ejecutar la siguiente instrucción:

```
$ chsh -
s
/bin/das
```

En este caso, el sistema pedirá la contraseña de nuestro usuario. No podremos ver los efectos del cambio hasta que hagamos un nuevo login, por cuyo motivo será más conveniente comprobarlo en una consola virtual que en un terminal de escritorio.

Atajos de teclado

Existen diferentes atajos de teclado que nos ayudarán durante la edición de la línea de comandos.

Opción	Descripción
Supr	Elimina el carácter sobre el cursor
Retroceso	Elimina el carácter a la izquierda del cursor
Inicio o (Ctrl+A)	Sitúa el cursor al inicio de la linea que estás editando
Fin o (Ctrl+E)	Sitúa el cursor al final del texto
Flechas Izquierda y Derecha	Retrocede o avanza un carácter la posición del cursor. Combinadas con Ctrl, el avance o retroceso se produce de palabra en palabra
Flechas Arriba y Abajo	Recorre el historial de las últimas instrucciones introducidas
Ctrl+l o Ctrl+L	Limpia el terminal al igual que la instrucción clear
Ctrl+t	Permuta la posición del carácter sobre el cursor con el de su izquierda
Ctrl+c	Para la instrucción en curso (también cancela programas en ejecución)
Ctrl+d	Cierra la sesión de terminal de la misma manera que el comando exit

Formas generales de obtener ayuda en Linux.

Existen múltiples y variadas formas de obtener ayuda en un sistema Linux, algunas son:

1) Muchos comandos poseen una opción para mostrar una ayuda breve acerca de su utilización. Esta opción usualmente es **-h**, **--help** ó **-?**.

2) El comando **man** formatea y despliega un manual bastante amplio acerca de comandos, formatos de ficheros de configuración, llamados 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. Ejemplo: $ **man chmod**

3) Existe un conjunto de comandos integrados (**builtin**) al bash que no poseen un manual propio. Para ellos se puede emplear el comando **help**. Algunos de los comandos integrados al bash son: **cd, fg, bg, logout, exit, umask, set, help, source, alias, echo, kill, jobs y export**. Ejemplo: $ **help logout**

4) El programa **info** despliega información acerca de comandos, en ocasiones más amplia que la que brinda **man**. Las páginas de info poseen una estructura arbórea (nodos), a través de las cuales se puede navegar con ayuda de comandos especiales. Ejemplo: $ **info ln**

5) El comando **whatis** realiza una búsqueda en las secciones del **man** y muestra la descripción abreviada del comando en cada una de las secciones que aparezca. Este comando trabaja con una base de datos que se actualiza periódicamente y se crea con el comando **makewhatis**. Ejemplo: $ **whatis shadow**

6) El comando **apropos**, dada una palabra busca en toda la base de datos del **whatis** desplegando todo lo encontrado. Ejemplo: $ **apropos passwd**

7) Muchos programas se empaquetan con su documentación en diversos formatos. Normalmente estas ayudas se agrupan en el directorio /usr/share/doc. También existen los HOWTOs (Como lograr...) escritos en muchos idiomas y formatos para varios temas disponibles. Algunos se empaquetan como cualquier otro programa o se localizan en Internet,

8) En Internet de forma general existen una gran cantidad de grupos de noticias, listas de discusión, sitios Web y FTP sobre Linux. En muchos de ellos incluso se pueden encontrar páginas de ayuda sobre los comandos en castellano, mientras que en nuestra versión de Linux es posible que nos aparezcan en Inglés.

2. Usuarios y grupos.

El usuario con más privilegios en Linux es aquel cuyo login 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 además existen grupos de usuarios también administrados por root o por un usuario designado por este. Los grupos 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. 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 un identificador de grupo (gid). Para comprobar el **uid** de nuestro usuario, y el gid del grupo principal al que pertenece, se ejecuta la orden id.

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 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.
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.

3. Comandos para manipular 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 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 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**)

Comando cd El comando cd se utiliza para cambiar el directorio actual. Sintaxis: cd [directorio]

Ejemplo:

```
$ cd /tmp
$ cd # cambia al directorio home del usuario.
$ cd ~ # (alt + 126)cambia al directorio home del usuario.
```

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

Ejemplo:

```
$ pwd# nos devuelve algo como /home/pepe/backup/pruebas
```

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# renombra un fichero

$ mv -i *.txt /tmp# mueve ficheros terminados en .txt al directorio /tmp

$ mv -u program.c src/# Actualiza el fichero destino si es menos reciente que el fuente.

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 .# copia un fichero en el directorio actual

$ cp -i /usr/bin/*sh /tmp # copia interactivamente los ficheros terminados en sh en un directorio llamado /tmp

Comando rm

El comando rm se utiliza para borrar (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.

Ejemplos:

$ rm prueba

$ rm -rf temp/

Comando chown

Se utiliza para cambiar el dueño y el grupo de un fichero. Existe también el comando **chgrp** que se emplea de forma similar pero para cambiar el grupo solamente. 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.

El propietario y el grupo de un fichero se puede comprobar con un ls -l Sintaxis: chown [opciones] <dueño>[.grupo] <ficheros>

Opción:

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

Ejemplos:

chown pepe.pepe tesis/ cambia el propietario de tesis a pepe y el grupo de tesis al grupo pepe
chown -R aso1.aso /tmp/oculto#otorga al usuario aso1 y al grupo aso la propiedad de /tmp/oculto de forma recursiva
chgrp ftp /usr/ftp #otorga al grupo ftp la propiedad de /usr/ftp

4. 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)

$ **ls -l /dev | less**

Toma la salida de ls y la envía como entrada a la orden less, con lo que se consigue un listado paginado.

$ **ls -l /dev | grep ma**

Toma la salida de la orden ls –l /dev y la filtra, realizando sobre ella una búsqueda de la cadena ma, de modo que solo aparecerán en la pantalla los ficheros que contengan dicha cadena.

$ **ls -R /etc | sort**

Lista recursivamente el contenido del directorio /etc y ordena la salida.

$ **ls -R /etc | sort | more**

Lista recursivamente el contenido del directorio /etc y ordena la salida, paginándola.

$ **ls -R /etc | sort | more | uniq**

Lista recursivamente el contenido del directorio /etc y ordena la salida paginándola y mostrando únicamente las líneas únicas. (Sin líneas duplicadas).

$ **locate user | grep bin**

Busca en la jerarquía de ficheros a aquellos que contengan en su nombre la cadena "user" y de estos selecciona los que contengan además "bin"

$ **grep /bin/bash /etc/passwd | wc -l**

Cuenta los usuarios que tienen como shell el bash

5. Comandos para paginar, visualizar y editar ficheros

Comando cat

El comando cat concatena (conCATenate) ficheros y los imprime en la salida estándar. Si no se le pasa ningún argumento lee de la entrada estándar. Existe también **zcat** que hace lo mismo pero con ficheros comprimidos. Si solo se da el origen a cat, utiliza como salida la pantalla. Es decir, **cat hola** muestra por pantalla el fichero hola. Si solo se da la salida a cat (**cat > fichero**) utiliza como **entrada el teclado**.

Ejemplo:

> # **cat /etc/passwd /etc/shadow** # muestra la salida estándar los 2 ficheros concatenados
>
> $ **cat > fichero#al** no decirle entrada, nos deja escribir por el teclado, cuando finalicemos de escribir (se indica con Control – Z) manda la entrada introducida a la salida del fichero.

Editor vi

El editor vi es el editor estándar de Unix y está orientado a comandos. Existe una versión conocida como **vim** (Vi IMproved) muy poderosa 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, edit, gedit, nano, emacs, 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, se ven en la última línea de la pantalla).

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 c n veces.

Comandos more y less

El comando more y less paginan (dividen en páginas) uno o varios ficheros y los muestran en la terminal (pantalla). De no indicarles un fichero, paginan la entrada estándar (que se manda mediante una tubería). 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. También **more** muestra sucesivamente el porcentaje del fichero visto hasta el momento. Tanto less como more proveen una serie de comandos para moverse con facilidad dentro del texto paginado.

Ejemplos:

$ less /etc/passwd
$ more /etc/passwd
$ cat fichero|less

Algunas teclas que podemos usar mientras usamos estos programas son:

q - permite interrumpir el proceso y salir.
/p - realiza búsquedas del patrón p dentro del texto. Para repetir la búsqueda del mismo patrón sólo es necesario escribir /.
[n]b - en more permite regresar n páginas (por defecto n es 1).
[n]f - en more se adelantan n páginas y en less, n líneas.

Man, para dar formato a su salida, utiliza por defecto el paginador less. Existen además los comandos zless y zmore que permiten paginar con less y more respectivamente, a los ficheros comprimidos sin necesidad de descomprimirlos previamente.

6. 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:

-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 aso1 /etc/passwd
$ grep -c root /etc/group
$ grep -l -r -i img /var/www/html/
$ ls -lia|grep "carta roja"
$ apt-serach mp3 | grep códec
```

Comando find

Permite buscar de forma recursiva en un directorio a 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:

-name <expresión> permite especificar patrones para los nombres de los ficheros a buscar.

-iname <expresión> permite especificar patrones para los nombres de los ficheros a buscar sin tener en cuenta mayúsculas y minúsculas.

-type <tipo> permite indicar el tipo de fichero a buscar. Este puede ser d para directorios, f para ficheros regulares, l para enlaces simbólicos, b para dispositivos de bloque, c para dispositivos de carácter, p para tuberías y s para sockets.

-size +/-<n> permite indicar el tamaño máximo y/o mínimo de los ficheros a buscar. Por defecto el tamaño se expresa en bloques de 512 bytes, pero si se precede este por un carácter c se referirá a bytes, k a kilobytes, w a palabras de dos bytes y b a bloques.

-perm [+|-]<modo> permite referirse a aquellos ficheros cuyos permisos sean exactamente modo, incluya todos los de modo (signo -) o incluya alguno de los de <modo> (signo +). 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).

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, aunque el usuario root normalmente tiene que ejecutarlo al menos una vez para crear la base de datos).

Ejemplo: $ **locate passwd**

7. 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, vamos a crear un archivo llamado lista-desordenada con el **vi** y meter dentro 5 nombres desordenados. Comprobad como con la orden **sort lista-desordenada > lista-ordenada** creamos un fichero llamado lista-ordenada y que contiene la lista ordenada.

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:

-f para el caso de tail se ejecuta de forma sostenida o sea se continúa visualizando el final del fichero hasta que se interrumpa el proceso (Ctrl-c).
-q no coloca los encabezamientos con el nombre de los ficheros cuando se indican varios (quiet).
-<n> imprime las n últimas (primeras) 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 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:

-l sólo cuenta líneas.
-c sólo cuenta bytes.
-w sólo cuenta palabras. ç

Ejemplos:

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

Comando cut

Permite 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.

Algunas opciones:

-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" | cut -d "," -f 1
$ echo "Esto es una prueba, 1 2 3, probando" | cut -d "," -f 3
$ echo "Esto es una prueba, 1 2 3, probando" | cut -c2-4
```

Comando tr

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

Sintaxis: 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.

Ejemplos:

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 canasta de 6 puntos

De este modo, si quisiéramos que en un fichero se cambiarán todas las palabras minúsculas por mayúsculas, podríamos simplemente escribir algo como esto: cat fichero.txt | tr [abcdefghijklmnopqrstuvwxyz] [ABCDEFGHIJKLMNOPQRSTUVWXYZ]

8. Comandos para compactar y agrupar ficheros

Comando 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 * # Comprime todos los ficheros del directorio actual (su extensión cambia a .gz)
$ gunzip big-file.gz # descomprime el fichero big-file.gz Todo lo compactado con gzip se puede descompactar con el Winzip de los sistemas Windows. También existen los pares de comandos zip y unzip (compatibles en ambos sentidos con Winzip), y compress y uncompress.

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), es decir, agrupar ficheros en uno solo.
-x permite extraer (destarear), es decir, desagrupar ficheros.
-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 como es por defecto. (Ojo, esta opción la usaremos siempre).
-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 cvzf grande * # crea un fichero grande donde estarán agrupados todos los ficheros del directorio actual y que además estará comprimido.
$ tar xvzf grande # desagrupa en el directorio actual el fichero grande y además lo descomprime.
# tar cf uconf.tar /etc/passwd /etc/shadow /etc/groups # agrupa en el fichero uconf.tar los ficheros passwd shadow y groups
# tar tf uconf.tar
# muestra los ficheros agrupados en uconf.tar
# tar xf uconf.tar # desagrupa el fichero conf.tar
# $ tar cMf /dev/fd0 /tmp/etc.tgz $ tar xMf /dev/fd0

9. Comandos para desconectarse del sistema

Comando exit
El comando exit permite terminar el shell actual, 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).

Comando halt, reboot y poweroff
Estos comandos nos permiten suspender, reiniciar o apagar el sistema.

Comando shutdown

Este comando nos permite "echar abajo" el sistema.

Algunas opciones interesantes:

-c cancela un shutdown que está en proceso de ejecución
-f Reinicia más rápido, ya que no controla la integridad de los sistemas de archivos
-h Cuando el sistema se apaga, apaga el ordenador (o lo intenta)

-r Cuando el sistema se apaga, intenta reiniciarlo

Después de estas opciones, se le indica cuando queremos apagar el sistema:

now lo apaga inmediatamente, ahora. 20:00 lo apaga a las 8 de la tarde
+10 lo apaga en 10 minutos
shutdown -h +4 10.

Comandos varios

Comando echo
El comando echo muestra en su salida todo lo que se le pase como argumento.

Ejemplo:
$ echo Hola amigo...
Hola amigo...

Comando set
Sirve para gestionar las variables y funciones del shell. Si se ejecuta set directamente sin parámetros, nos devuelve los nombres y valores de varias funciones del shell.
set

En bash no es necesario asignar variables mediante set variable=valor, sino que se puede usar directamente directamente variable=valor. Sin embargo, con otros shells, como csh en los que es obligatorio usar el set para asignar valores a las variables.

Comando history
El comando history nos permite controlar el histórico de comandos que se van almacenando en el shell. El histórico se almacena por defecto en el fichero ~/.history lo que nos indica que existe un histórico diferenciado para cada usuario. Si ejecutamos el comando history nos devolverá la relación de comandos para ese usuario.

history

Una gran posibilidad es enviar la salida de la orden history a un fichero, con lo que conseguimos tener una relación de los comandos que hemos ejecutado.

Comando date
El comando date sirve para consultar y cambiar la fecha y hora del sistema.
date
El comando date tiene muchas opciones que permiten realizar muchas tareas.

Algunos ejemplos:
$ date --date="2 days ago"#nos da la fecha de hace dos días
$ date --date="3 months 1 day"#nos da la fecha que será en 3 meses y un día.
$ date --date="25 Dec"#nos da el dia que será el 25 de diciembre del año actual
$ date "+%B %d"#nos devuelve el nombre del mes actual y el número de día del mes.

$ date —set="2006-6-20 11:59 AM"#ajusta la fecha del sistema $ date --set="+3 minutes"# adelante la hora del sistema en 3 minutos.

Comando time

El comando time sirve para cronometrar cuanto tiempo tarda en ejecutarse un comando cualquiera, y que recursos consume. Puede ser una orden interesante para comprobar el rendimiento de los sistemas.

```
# time wc /etc/passwd
```

Comando uptime

El comando uptime nos indica las el "uptime" del sistema, esto es, el tiempo que un equipo está funcionando. En un servidor es muy importante que este uptime sea lo más elevado posible. Uptime nos devuelve cuánto tiempo lleva el equipo levantado, el número de usuarios que están conectados, y la media de las cargas de sistema para los últimos 1, 5, y 15 minutos.

```
# uptime
```

Comando top

El comando top nos muestra los procesos que se están ejecutando en nuestra máquina, y cuántos recursos están ocupando. Nos permite ver como se encuentra nuestro sistema de "cargado", qué procesos son los que están consumiendo más recursos, etc. Si pulsamos h en la pantalla de top, veremos una pantalla de ayuda donde podemos ordenar la lista de procesos según su consumo de cpu, memoria, etc.

```
# top
```

Comando dmesg

Nos da una lista con todos los mensajes que se han producido durante el arranque del sistema.

```
# dmesg
```