INTRODUCCIÓN AL SISTEMA OPERATIVO LINUX

1 – INTRODUCCIÓN....2
2 – INSTALACIÓN....3
3 – CONCEPTOS BÁSICOS....3
4 – ÓRDENES RELACIONADAS CON ARCHIVOS Y DIRECTORIOS....6
5 – CARACTERES COMODÍN O WILDCARDS....11
6 – REDIRECCIONAMIENTO Y TUBERÍAS....11
7 – CAMBIO DE PERMISOS DE LOS FICHEROS....13
8 - ENLACES....13
9 – EL DIRECTORIO RAÍZ: ESTRUCTURA JERÁRQUICA DEL SISTEMA....16
10 – LA EDICIÓN DE TEXTO. EL EDITOR vi....18
11 – ARCHIVAR Y COMPRIMIR FICHEROS....20
12 – CONTROL DE TAREAS Y PROCESOS....22
13 – GESTIÓN DE SISTEMAS DE FICHEROS....23
14 – GESTIÓN DE USUARIOS Y GRUPOS....25
16 – GUIONES DEL INTÉRPRETE DE COMANDOS....28

1 – INTRODUCCIÓN

Linux es un sistema operativo gratuito y de libre distribución, basado en el sistema operativo UNIX. Es una versión de Unix, desarrollada inicialmente por Linus Torvalds en la Universidad de Helsinki (Finlandia) con la ayuda de muchos programadores y expertos de Unix a través de Internet.

Se viene desarrollando desde 1991, diseñado especialmente para trabajar en red y con las ventajas de ofrecer una gran estabilidad, excelente administración de recursos y código abierto (open source). Además, el kernel o núcleo es actualizado continuamente.

CARACTERÍSTICAS

Linux es un sistema operativo multitarea y multiusuario, es decir, varios usuarios pueden trabajar simultáneamente en él, y cada uno de ellos puede tener varios programas en ejecución. Además, fue desarrollado buscando la portabilidad, de forma que casi todo el software gratuito desarrollado para Unix se compila en Linux sin problemas, y todo lo que se hace para Linux es de libre distribución.

DISTRIBUCIONES

Por otra parte, uno de los primeros conceptos que aparece al iniciarse en Linux es el de distribución. Una distribución de Linux es un agrupamiento del núcleo del sistema operativo (la parte desarrollada por Linus Torvalds) y otra serie de aplicaciones desarrolladas por determinadas empresas.

Las distribuciones más conocidas son RedHat, Debian, Slackware, SuSE, Mandriva, Ubuntu, Fedora, etc., y por supuesto Guadalinex.

Guadalinex: http://www.guadalinex.org/
Red Hat: http://www.redhat.es/
Debian: http://www.debian.org/
Slackware: http://www.slackware.org/
SuSE: http://www.suse.com/
Linux Mandriva: http://www.mandriva.com/es
Ubuntu: http://www.ubuntu-es.org/
Fedora: http://www.fedora-es.com/

SISTEMA GRÁFICO

Casi todas las distribuciones más recientes incluyen el sistema X Window, encargado de controlar y manejar la interfaz de usuario. Será el programa controlador de ventanas el que gestione el aspecto y comportamiento de X Window. Los controladores de ventanas estandarizados son KDE y GNOME. El programa servidor X será el que dibuje en la pantalla.

EL PROYECTO GNU Y LA GPL

Si Linux es un sistema operativo gratuito, ¿porqué estas empresas cobran dinero por las distribuciones que sacan al mercado?.

Linux está regido por lo que se conoce como la Licencia Pública General (GPL) o General Public License. La GPL fue desarrollada para el proyecto GNU por la Free Software Fundation (Fundación por el Software Gratuito). El proyecto GNU es el que iniciaron los hackers para promover sistemas operativos de libre distribución alternativos al Unix (GNU's Not Unix: guh News not Unix), ellos fueron anteriores a Linus Torvalds, de ahí que sea más justo referirse al sistema como GNU/Linux.

El software regido por la GPL no es de dominio público, es decir, tiene el copyright de su autor o autores. Sin embargo, permite a los usuarios modificar el software, copiarlo y redistribuirlo, aunque cualquier trabajo derivado de un programa GPL estará también regido por la GPL.

Otro concepto diferente es el de software gratuito, o de libre distribución, lo cual no quiere decir que los fuentes estén disponibles, y por lo tanto no es software regido por la GPL.

DISEÑO Y FILOSOFÍA

Linux es un sistema operativo desarrollado por “hackers”, programadores dedicados por afición a explotar sus ordenadores al máximo, con resultados útiles para otras personas (no piratas informáticos). Son un grupo de programadores de todo el mundo, unidos por Internet, al que cualquiera tiene la oportunidad de unirse para ayudar al desarrollo y depuración del núcleo, aportar nuevo software, escribir documentación, etc.

Por regla general hay una nueva versión del núcleo cada mes. Dependerá del número de errores a corregir y de la cantidad de información recibida de los usuarios. Por lo tanto, al contrario de lo que ocurre con otro tipo de software no regido por GPL, es difícil decir que todos los errores hayan sido corregidos en cierta fecha.

Si alguien desarrolla una aplicación o una nueva característica para el núcleo, se añade inicialmente en una etapa “alfa”, es decir, en fase de pruebas y que puede contener errores. El software alfa es enviado normalmente a servidores dedicados a Linux, y los usuarios que se lo llevan lo prueban e informan al autor sobre errores o sugerencias por correo electrónico.

Una vez que se corrigen los problemas iniciales, el código pasa a una etapa “beta”, en la que se considera estable pero incompleto, es decir, funciona pero no incluye todas la funcionalidades previstas. Estas versiones suelen llevar numeración impar (1.1, 0.99).

Finalmente, se puede pasar a una etapa final o de “explotación”, en la que el software se considera terminado. Aunque pudiera haber errores, serían leves e infrecuentes. Estas versiones suelen tener numeración par (1.2).

2 – INSTALACIÓN

Uno de los conceptos principales a tener en cuenta antes de la instalación es el de partición. Si deseamos conservar Windows y todos los programas para este sistema, para obtener un buen rendimiento será preferible instalar Linux sobre una partición diferente a la de Windows, empleando el sistema de archivos propio de Linux: *ext2* o bien *ext3*.

Podemos utilizar un gestor de particiones para crear una partición de 6 GB aproximadamente formateada con sistema de archivos ext2 (ext3). Además, crearemos una partición *Swap* (de intercambio) con 512 KB aproximadamente. Esta partición servirá a Linux de área de intercambio (memoria virtual) cuando se agote la memoria disponible.

Antes de la instalación, si ya tenemos otro sistema operativo instalado, conviene que ejecutemos el Desfragmentador y Scandisk previamente al particionado.

3 – CONCEPTOS BÁSICOS

Dado que Linux es un sistema operativo multiusuario, cada uno de ellos tendrá que identificarse en el sistema. Para ello deben presentarse mediante un Login y una Contraseña.

Sólo desde la cuenta de root se puede crear una cuenta de usuario. Además, para el uso cotidiano lo más aconsejable es utilizar una cuenta distinta a la de root, y reservar ésta para el uso de comandos privilegiados y para el mantenimiento del sistema.

Por otra parte, cada sistema Unix tiene un *hostname*, nombre del sistema, asignado. El hostname le da nombre a la máquina y es usado para identificar máquinas en una red. Pero incluso si la máquina no está en red debería tener un nombre.

El prompt del sistema, además de nuestro login y el hostname, nos mostrará por defecto el símbolo # para el usuario root, y el símbolo $ para cualquier otro usuario.

Un usuario puede utilizar de forma temporal la cuenta de otro usuario. Para ello utilizará el comando **su**.

- Si hemos entrado como root, escribimos

  #su <nombre de usuario>

  y no se nos pedirá la contraseña. Para abandonar esta cuenta escribimos **exit**.
- Si hemos entrado como un usuario no privilegiado, escribimos

  #su <nombre de usuario>

  y se nos pedirá la contraseña. Si no escribimos un nombre de usuario, por defecto se toma el root, por lo que en la contraseña debemos teclear la de root. Para abandonar también escribimos exit.

CONSOLAS VIRTUALES

La consola del sistema es el conjunto formado por el monitor y teclado conectados directamente al sistema.

Dado que Linux es un sistema operativo multiusuario, puede tener otros terminales conectados al sistema, pero que no serán la consola.

Mediante las consolas virtuales (VC's), Linux permite tener más de una sesión de trabajo activa desde la consola a la vez. Es decir, podemos tener activas varias cuentas de usuario distintas en el mismo momento, aunque sólo se puede mirar una consola virtual cada vez.

Es posible habilitar hasta 12 consolas virtuales, una por cada tecla de función del teclado.

Para cambiar de una consola virtual a otra, pulsamos la combinación ALT+ <tecla de función>.

Al hacerlo, el sistema nos pregunta nuevamente el login. Para ir a una sesión iniciada en otra consola virtual, pulsamos ALT más la tecla de función correspondiente.

Para cambiar de modo gráfico a texto, pulsamos la combinación CTRL+ALT+<tecla de función>, la tecla de función por defecto puede ser cualquiera de las seis primeras. Por defecto, podremos volver al modo gráfico pulsando ALT+F7.

INTÉRPRETES DE COMANDOS

Un intérprete de comandos es un programa que toma la entrada del usuario, por ejemplo las órdenes que teclea, y la traduce a instrucciones.

Podemos compararlo con el COMMAND.COM de MS-DOS, que realiza exactamente la misma tarea.

El intérprete de comandos será una de las interfaces con Unix, y el X Window será otra interfaz que nos permite ejecutar órdenes usando el ratón y el teclado.

Además de ser un intérprete interactivo de los comandos que tecleamos, el intérprete de comandos es también un lenguaje de programación, el cual nos permite escribir guiones que permiten juntar varias órdenes en un fichero. Similar a los ficheros batch de MS-DOS.

En Unix existen varios tipos de intérpretes de comandos. Los dos más importantes son:

- Sh o Bourne shell: utiliza una sintaxis similar a la usada en los primeros sistemas Unix.
- Csh o C shell: utiliza una sintaxis diferente a la de sh, similar al lenguaje de programación C.

En Linux también disponemos de varios intérpretes de comandos, con algunas diferencias respecto a los de Unix. Los más usados son:

- Bash o Bourne Again Shell: es equivalente a Bourne shell, pero con muchas características avanzadas de la C shell. Cualquier guión (script) escrito para Bourne funcionará en bash.
- Tsch (exTended C shell): es una versión extendida del C original.

Sólo aparecerán diferencias entre unos y otros a la hora de escribir guiones. Es decir, en lo que respecta a los comandos usuales es indiferente el tipo de intérprete de comandos usado.

El formato general de una orden en Linux es:

comando [-opciones] [argumentos]

Las opciones sirven para influir en el comportamiento del comando. Generalmente van precedidas de un guión, y cuando hay varias es posible acumularlas. Veamos un ejemplo:

$ls –la

En esta orden, el comando es ls, y las opciones son l y a.

Los argumentos comunican al comando los datos necesarios sobre los que actuar, como por ejemplo un directorio o fichero.

SALIDA Y CIERRE DEL SISTEMA

Para finalizar la sesión iniciada por un usuario en modo texto escribimos **exit**.

También podemos escribir **logout** si hemos iniciado el sistema en modo consola.

Para apagar el equipo, al igual que ocurre con otros sistemas operativos, habrá que hacerlo correctamente para que el sistema de archivos no se vea dañado. Esta operación se puede realizar de varias formas:

- Usuario root:

  reboot: reinicia el sistema.

  halt y **poweroff** apagan el sistema; si el equipo es antiguo y no se apaga automáticamente, apagamos el equipo cuando aparezca un mensaje indicándonos que podemos hacerlo.
- Usuario no privilegiado: solamente podrá reiniciar el sistema mediante la combinación de teclas CRTL+ ALT+ SUPR.

Por otra parte, si hemos iniciado el sistema en modo texto podemos iniciar el servidor X Window escribiendo **startx**. Cuando apaguemos la sesión desde el modo gráfico volveremos al modo texto.

CAMBIAR LA PALABRA DE PASO

Podemos cambiar la contraseña de usuario empleando la orden **passwd**. Nos pedirá la clave actual (current) y la nueva. Volverá a pedir una segunda vez la nueva para validarla.

El usuario root podrá cambiar la contraseña de cualquier otro usuario. Sin embargo, los usuarios no privilegiados solamente podrán cambiar su propia clave.

FICHEROS Y DIRECTORIOS

Bajo la mayoría de los sistemas operativos existe el concepto de fichero. Un fichero es un conjunto de información al que se le ha asignado un nombre que lo identifica.

En Linux no hay un formato estándar para los nombres de los ficheros como lo hay en otros sistemas operativos. En general puede contener cualquier carácter (excepto */*), y pueden tener una longitud de hasta 256 caracteres.

Linux tiene tres tipos diferentes de ficheros, cada uno de ellos nombrados de acuerdo a la misma convención:

- Ficheros regulares u ordinarios: aquellos donde se guarda información como programas ejecutables, bases de datos, ficheros de texto, etc.

- Directorios: aquellos que contienen la tabla o lista de ficheros de un sistema de ficheros. Cada entrada de la tabla tendrá asociados el nombre del fichero y su número de i-nodo.
- Ficheros especiales: identifican dispositivos de hardware, y controladores de software.

Por otra parte, un directorio será una colección de ficheros. También tiene asignado un nombre que lo identifica, y mantienen una estructura de árbol. Es decir, los directorios se pueden anidar.

Así, podemos hacer referencia a un fichero empleando su nombre con camino. El camino será la ruta que se debe tomar para localizar al fichero.

Los distintos directorios del camino y el fichero irán separados por el carácter **/** (recuerda que en MS-DOS se utiliza ****).

EL ÁRBOL DE DIRECTORIOS

La mayoría de los sistemas Unix tienen una distribución de ficheros estándar, de forma que los ficheros y los recursos puedan ser localizados fácilmente.

Esta distribución forma el árbol de directorios, que comienza en el directorio **/**, conocido como *directorio raíz*.

Directamente colgando de él hay algunos subdirectorios importantes como */bin*, */etc*, */dev*, */usr* entre otros. Estos contienen a su vez otros directorios con ficheros de configuración del sistema, programas, etc.

En particular, cada usuario tiene un directorio *personal*. En este directorio guardará el usuario sus ficheros. El directorio *personal* usualmente cuelga del directorio */home* y recibe el nombre del usuario al que pertenece. Por lo tanto, el directorio *personal* de *usu1* será */home/usu1*. Cuando entramos en el sistema, el directorio de trabajo inicial es nuestro directorio *personal*.

Al igual que en MS-DOS, el directorio de trabajo actual es aquel en el que nos encontramos situados, y sobre el que se realizarán las operaciones si no se indica lo contrario mediante los *caminos*. El camino a un fichero puede ser relativo o absoluto:

- Camino o ruta relativos: cuando el nombre del fichero comienza con un carácter distinto a **/**, el sistema supone que nos estamos refiriendo al fichero con su posición relativa a su directorio de trabajo.
- Camino o ruta absolutos: si el nombre del fichero comienza con **/**, el sistema lo interpreta como el camino completo al fichero desde el directorio raíz.

Por último, bajo bash y tcsh podemos referenciar el directorio *personal* usando el carácter de la tilde de la letra ñ (**~**). Por ejemplo *~/tmp* y *~usu1/tmp* equivalen a */home/usu1/tmp*.

El uso de la tilde es una ayuda sintáctica proporcionada por el intérprete de comandos, en realidad no existe ningún directorio llamado **~**.

4 – ÓRDENES RELACIONADAS CON ARCHIVOS Y DIRECTORIOS

Para cada uno de los comandos que vamos a ver a continuación sólo citaremos aquellas opciones más usuales. Podemos encontrar información sobre los comandos consultando la utilidad que proporciona Linux conocida como *páginas de manual*. Para acceder a las páginas de manual utilizamos la orden **man** seguida del nombre del comando.

cd Cambia el directorio de trabajo actual.

SINTAXIS: cd <directorio>

OPCIONES: **cd –** cambia al último directorio en el que estuvimos antes del actual.

cd.. cambia al directorio padre.

cd cambia al directorio de origen, el home.

cd / cambia al directorio raíz

ls Muestra información sobre los ficheros o directorios indicados. Los símbolos que aparecen junto al nombre del fichero indican * ejecutable, / directorio, @ enlace simbólico, ningún símbolo indica fichero normal.

SINTAXIS: **ls** <fichero1> <fichero2> ... <ficheroN>

donde <fichero1> ... <ficheroN> son los ficheros o directorios a listar

OPCIONES: **-a** permite ver los nombres de ficheros que comienzan por un punto (ocultos).

-l muestra un listado largo con tamaño, propietario, permisos, etc.

-g igual que "l", sin mostrar el dueño.

-h junto con "l" muestra el tamaño en la unidad de medida mayor.

-i muestra el i-nodo del fichero.

-n igual que "l", mostrando GID y UID en lugar de los nombres.

En este punto, antes de continuar con el resto de órdenes, veamos las categorías de usuarios y los tipos de permisos.

Cada usuario es dueño de su directorio personal y será dueño también de los archivos que incluya en él.

Un usuario en Linux podrá configurar permisos en sus archivos. Por ello, distinguiremos por un lado tres categorías de usuarios, y por otro los tipos de permisos que cada uno de ellos puede tener sobre un archivo y/o directorio.

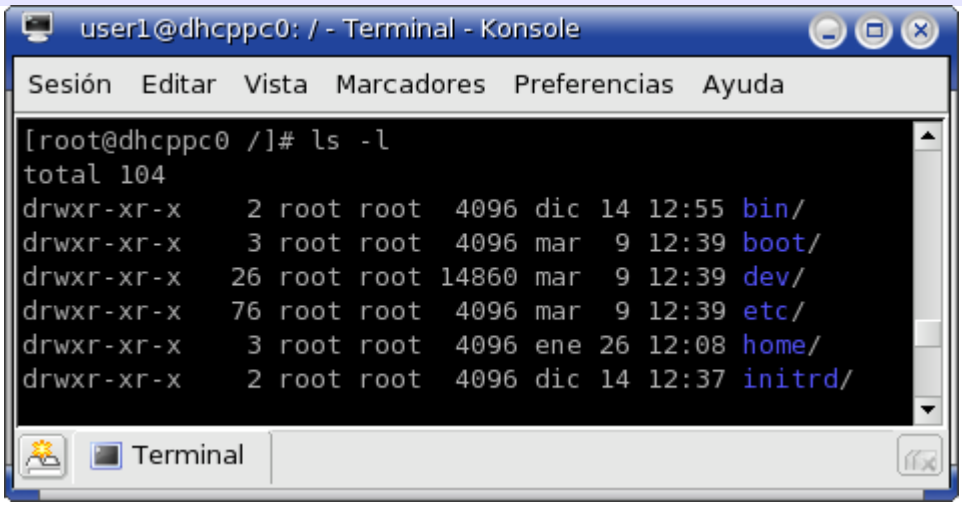
Categorías de usuarios

- *Dueño* del archivo.
- *Grupo dueño*, formado por todos los usuarios que son miembros de un grupo asociado al archivo.
- *Resto de usuarios*, todos los usuarios que no son ni el dueño ni miembros del grupo dueño.

Tipos de permisos

- *Lectura* (r de Read, leer): para un archivo permite leer su contenido, para un directorio permite que se muestren los archivos que contiene.
- *Escritura* (w de Write, escribir): para un archivo permite que se modifique su contenido, para un directorio permite agregar y quitar archivos.
- *Ejecución* (x de eXecute, ejecutar): para un archivo permite su ejecución, para un directorio permite que el usuario lo recorra (que entre y pase por él) – si no tiene permiso de lectura, aunque pueda entrar no podrá ver el contenido.
- *S:* para los ficheros que tienen permiso de ejecución, indica que el usuario que ejecuta el fichero tiene los mismos privilegios que el usuario que lo creó.
- *T:(pegajoso)* sólo puede borrar el archivo el usuario que lo creó.

Cuando ejecutamos el comando **ls –l**, podemos ver la configuración de permisos de los archivos:



- El primer carácter indica el tipo de archivo: “d” si es directorio, “-“ si es un archivo regular, “l” si es un enlace simbólico.
- Los siguientes nueve caracteres indican los permisos para el dueño, el grupo dueño y otros (rwxrwxrwx); si aparece un guión, indica que el permiso correspondiente no está habilitado.
- El siguiente número indica el número de vínculos.
- Nombre del dueño y nombre del grupo dueño.
- Tamaño en bytes.
- Fecha de la última modificación.
- Nombre del archivo.

cp Copia un fichero o ficheros en otro fichero o directorio.

SINTAXIS: cp <fichero1> <fichero2> ... <ficheroN> <destino>

donde <ficheroX> es el fichero a copiar y <destino> es el fichero o directorio de destino.

Podemos utilizar **.** y **..** para referirnos al directorio actual y al directorio padre respectivamente.

mv Mueve un fichero o ficheros a otro fichero o directorio. Es equivalente a una copia seguida del borrado del original. Puede ser usado para renombrar ficheros.

SINTAXIS: mv <fichero1> <fichero2> ... <ficheroN> <destino>

rm Borra ficheros, que además no serán recuperables, por defecto pide confirmación.

SINTAXIS: rm <fichero1> <fichero2> ... <ficheroN>

OPCIONES: **-f** no pide confirmación antes de borrar un fichero.

-r borra directorios recursivamente (borran el directorio y todo su contenido).

mkdir Crea directorios nuevos.

SINTAXIS: mkdir <directorio1> <directorio2> ... <directorioN>

rmdir Borra directorios vacíos. Si no está vacío, el sistema nos lo indicará con un mensaje.

SINTAXIS: rmdir <directorio1> <directorio2> ... <directorioN>

man Muestra la página de manual del comando o recurso especificado.

SINTAXIS: man <comando>

Una vez en la página, pulsamos la tecla **q** (quit) para volver al prompt, las teclas **Intro**, **b** , barra espaciadora y los **cursores** nos permiten movernos por la página. Pulsando la tecla "/", podremos

realizar búsquedas, escribiendo tras ella la cadena que deseamos encontrar en la página, y pulsando la tecla "n" seguiremos buscando el mismo item.

more Muestra el contenido de los ficheros indicados, una pantalla cada vez.

SINTAXIS: more <fichero1> <fichero2> ... <ficheroN>

La teclas que nos permiten movernos por el fichero son: **b** va a la página anterior, **barra espaciadora** va a la página siguiente, **flechas de cursor** arriba y abajo, **q** finaliza la ejecución de more.Pulsando la tecla "/", podremos realizar búsquedas, escribiendo tras ella la cadena que deseamos encontrar en la página, y pulsando la tecla "n" seguiremos buscando el mismo item.

cat Muestra el contenido de los ficheros de una vez. (Además, nos servirá para crear y concatenar –concatenate- ficheros de texto utilizando los redireccionamientos y tuberías).

SINTAXIS: cat <fichero1> <fichero2> ... <ficheroN>

sort Muestra en la salida, en orden ascendente, la lista de argumentos que se le pasa a través de la entrada. (Además, nos servirá para crear y concatenar –concatenate- ficheros de texto utilizando los redireccionamientos y tuberías).

less Muestra el contenido de los ficheros de la misma forma que more, con la diferencia de que podremos movernos por ellos utilizando también las flechas de cursor izquierda y derecha.

SINTAXIS: less <fichero1> <fichero2> ... <ficheroN>

pwd Muestra la ruta del directorio en el que nos encontramos (Print Working Directory).

du Muestra el espacio en disco que ocupa el archivo o directorio.

SINTAXIS: du <fichero1> <fichero2> ... <ficheroN>

OPCIONES: **-s** para un directorio, no muestra el espacio que ocupan cada uno de sus archivos y subdirectorios, sino que realiza la suma y nos la muestra.

cut Muestra trozos de líneas del fichero especificado.

SINTAXIS: cut <opciones> <fichero>.

OPCIONES: **-c <lista>** corta las columnas especificadas en <lista>.

Ejemplo: cut -c 1-5 fichero

Muestra las columnas de la 1 a la 5 de todas las líneas del fichero.

find Muestra el nombre de los archivos encontrados que se correspondan con cierto conjunto de criterios.

SINTAXIS: find <opciones> <directorios> <criterios> <acción>.

OPCIONES: **-follow** seguir los enlaces simbólicos si apuntan a directorios.

CRITERIOS: **-type <tipo>** busca archivos de un tipo dado (f regular, d directorio, l enlace simbólico)

-name <nombre> encuentra los archivos cuyo nombre coincida con el dado (**-iname** para que no distinga entre mayúsculas y minúsculas). Los criterios se pueden combinar con -a, -o, -not.

ACCIONES: **-exec <comando>** ejecuta un comando sobre cada archivo encontrado. La posición del archivo se indica con {} y el comando finaliza con ; que debe ser desactivado para que el shell no lo interprete mediante \

EJEMPLO: find -name *.htm -o -name *.html –exec mv {} /www \;

Busca en el directorio actual todos los ficheros con extensión htm o html y los mueve al directorio /www.

grep Muestra todas las líneas de un fichero dado que coinciden con un cierto patrón.

SINTAXIS: grep <patrón> <fichero1> <fichero2> ... <ficheroN> donde **patrón** es una expresión regular.

OPCIONES: **-r** para buscar de forma recursiva dentro de los ficheros de un directorio.

-n muestra el número correspondiente al lugar que ocupa en el fichero la línea encontrada.

-i no distingue entre mayúsculas y minúsculas.

-v muestra las líneas que no se corresponden con el patrón.

-w el patrón debe aparecer como una palabra completa y no como parte de otra palabra.

head muestra las 10 primeras líneas del contenido de los archivos especificados.

SINTAXIS: head <fichero1> <fichero2> ... <ficheroN>.

OPCIONES: **-n** muestra las n primeras líneas, en lugar de 10.

tail Muestra las 10 últimas líneas del contenido de los archivos especificados.

SINTAXIS: tail <fichero1> <fichero2> ... <ficheroN>.

OPCIONES: **-n** muestra las n últimas líneas, en lugar de 10.

wc Muestra el número de caracteres, palabras y líneas de un fichero.

SINTAXIS: wc <opciones> <fichero1> <fichero2> ... <ficheroN>.

OPCIONES: **-c** cuenta caracteres.

-w cuenta palabras.

-l cuenta líneas.

OTROS COMANDOS BÁSICOS

metacarácter `

La salida de un comando encerrado entre los metacaracteres **`comando`** es incrustada en el lugar que se encuentra como un argumento.

EJEMPLO: cat `date` >>fichero.

Añade la fecha actual al final del fichero

alias Asigna un nombre o etiqueta a la ejecución de un comando con sus opciones.

SINTAXIS: alias etiqueta='orden'.

La orden alias solamente, muestra todos los alias que hay creados. La orden **unalias** elimina el alias especificado.

cal Muestra el calendario de año (o bien mes y año) especificado.

clear Limpia la pantalla.

date Muestra el día y la hora.

echo Muestra por pantalla los argumentos que le pasamos.

SINTAXIS: echo <argumento1> <argumento2> ... <argumentoN>.

uname Junto con la opción **–r** muestra la versión del núcleo.

http://www.linux-es.org/kernel

who Muestra qué usuarios están conectados al sistema en ese momento, en qué terminal y desde qué hora.

whoami Indica cuál es la terminal y la sesión en la que se está trabajando.

5 – CARACTERES COMODÍN O WILDCARDS

Una característica importante de la mayoría de los intérpretes de comandos en Unix es la capacidad para referirse a más de un fichero.

Una forma de hacerlo es utilizando caracteres especiales llamados *comodines*.

Al igual que en MS-DOS, el comodín ***** hace referencia a cualquier carácter o cadena de caracteres en el nombre del fichero. El intérprete de comandos sustituirá el asterisco por todas las combinaciones posibles provenientes de los ficheros en el directorio al cual nos estamos refiriendo. Se dice que está realizando una *expansión de comodines*.

El carácter **?** es también comodín, aunque solamente expande un carácter.

Con ambos caracteres existe una excepción. No afectarán a aquellos ficheros que comienzan por un punto, y que son ocultos para órdenes como **ls**.

Además, podemos utilizar los **corchetes** para referirnos a un conjunto de caracteres o bien un rango de caracteres ASCII.

Ejemplos:

$ls *n*	muestra todos los archivos y directorios, del directorio actual, que contienen el carácter n
$ls *	muestra todos los archivos y directorios del directorio actual
$ls tm?	muestra todos los archivos y directorios del directorio actual que comienzan por *tm* y contienen tres caracteres
$ls tabla[123]a	muestra todos los archivos y directorios del directorio actual que comienzan por *tabla*, seguidos del carácter *1*, *2* ó *3*, y terminan en *a*
$ls ??base[A-Z][5-9]*	muestra todos los archivos y directorios del directorio actual que comienzan con dos caracteres cualesquiera, seguidos de la cadena *base*, a continuación una letra mayúscula, seguida de un número del *5* al *9* y por último una cadena de caracteres (uno, varios o ninguno)

6 – REDIRECCIONAMIENTO Y TUBERÍAS

Muchos comandos de Unix toman su entrada de la ENTRADA ESTÁNDAR (stdin) y envían su salida a la SALIDA ESTÁNDAR (stdout).

El intérprete de comandos configura el sistema de forma que la stdin es el teclado y la stdout la pantalla.

Ejemplo 1: Si al comando **cat** no le pasamos argumentos, actuará mostrando en pantalla todo lo que hayamos tecleado antes de un Intro. Para indicarle al sistema que queremos finalizar la ejecución de cat, pulsamos la combinación de teclas CTRL+ D.

```
$cat
hola        lo que recibe de la stdin
hola        lo que devuelve a la stdout
CTRL+ D     fin de cat
$
```

Ejemplo 2: El comando **sort** actúa de forma parecida. Si lo ejecutamos e introducimos un conjunto de líneas desde la stdin, cuando pulsemos la combinación CTRL+ D devolverá a la stdout las mismas líneas pero de forma ordenada.

```
$sort
méndez
luque
```

```
rodríguez
CTRL+ D     fin de entradas
luque
méndez
rodríguez
$
```

REDIRECCIONAMIENTO DE LA ENTRADA Y LA SALIDA

Al igual que en MS-DOS, podemos utilizar los caracteres **>** y **<** para redireccionar la salida y la entrada estándar respectivamente.

Para redireccionar la stdout podemos utilizar:

a. Redirección destructiva: crea un nuevo fichero, o sobrescribe el contenido de uno que ya existe, con aquella información que el comando recibe de la stdin.

```
$comando >fichero
```

b. Redirección no destructiva: crea un nuevo fichero o añade al final del contenido de uno que ya existe la información que el comando recibe de la stdin.

```
$comando >>fichero
```

Para redireccionar la stdin utilizamos el carácter **<**. Con cat no tiene mucho sentido, ya que el resultado que se obtiene es igual al que produce sin utilizar este redireccionamiento. Con sort, mostrará en pantalla el contenido del fichero que le indiquemos de forma ordenada.

Ejemplos:

```
$cat >fichero
a
c
b
CTRL+D
$cat >>fichero
x
z
y
CTRL+D
$cat fichero
a
c
b
x
z
y
$sort < fichero
a
b
c
x
y
z
$
```

TUBERÍAS (PIPES)

Hemos visto que el comando sort, aunque simple, actúa como filtro, devolviendo a la salida lo que recibe desde la entrada de forma ordenada.

Las tuberías pueden ser utilizadas para combinar comandos, de forma que la salida del primero es enviada a la entrada del segundo y así sucesivamente.

De esta forma, podemos aplicar un filtro a la stdout del comando **ls** enviándola a la stdin de **sort**. Lo que conseguimos es conectar una cadena de comandos en una tubería.

Para crear las tuberías utilizamos el carácter **|** (barra vertical, carácter de canalización).

Ejemplos:

$ls /usr/bin |more muestra el contenido de /usr/bin por pantallas.

$ls |sort -r muestra un listado del directorio actual ordenado alfabéticamente de mayor a menor.

$ls |sort -r |head -1 veremos el primer fichero de un listado del directorio actual ordenado alfabéticamente de mayor a menor.

7 – CAMBIO DE PERMISOS DE LOS FICHEROS

Ya sabemos que los permisos de cada fichero se pueden ver con el comando **ls –l**. Sólo el propietario podrá cambiar los permisos de los ficheros.

chmod Permite establecer los permisos de un fichero o conjunto de ficheros.

SINTAXIS: chmod {a,u,g,o}{+ | -}{r,w,x,X,s,t,u,g,o} <fichero1> <fichero2> ... <ficheroN>

Es decir, primero indicamos a quién afecta el permiso que queremos cambiar: **u** (user), **g** (group), **o** (other), **a** (all). Si no escribimos nada, por defecto *chmod* toma *a*.

Seguidamente indicamos si queremos dar (**+**) o quitar (**-**) un permiso. Y finalmente establecemos el permiso o permisos que queremos dar o quitar: **r** (permiso de lectura), **w** (permiso de escritura), **x** (permiso de ejecución), **X** (permiso de ejecución sólo si el fichero es un directorio), **s** (el que ejecuta el fichero tiene los mismos permisos que el que lo creó), **t** (el fichero se mantiene siempre en memoria y sólo puede borrarlo el dueño), **u** (los mismos permisos que el dueño), **g** (los mismos permisos que el grupo dueño), **o** (los mismos permisos que el resto de usuarios).

chown Permite cambiar el propietario de un conjunto de ficheros. Los nombres de propietario que admite son aquellos que están almacenados como usuarios en el fichero */etc/passwd*.

SINTAXIS: chown <nuevo dueño> <fichero1> <fichero2> ... <ficheroN>

chgrp Permite cambiar el grupo dueño de un conjunto de ficheros Los nombres de grupo que admite son aquellos que están almacenados como usuarios en el fichero */etc/group*.

SINTAXIS: chgrp <nuevo grupo> <fichero1> <fichero2> ... <ficheroN>

Ejemplos:

```
$chmod a+r <fichero>
$chmod +r <fichero>
$chmod og-x <fichero>
$chmod u+rwx <fichero>
$chmod o-rwx <fichero>
$chmod o-rwx <directorio>
```

8 - ENLACES

Los enlaces nos van a permitir realizar copias de los ficheros con otro nombre, para poder acceder a ellos desde lugares distintos a su ubicación original, con un ahorro de espacio muy importante con respecto al comando **cp**.

En Unix todos los ficheros tienen asignada una tabla en la cual se guardan como máximo 10 direcciones de bloques de disco que ocupa el fichero. A esta tabla se la conoce como i-nodo. Si el fichero ocupa más de 10 bloques se utilizan i-nodos de segundo nivel y si necesita más, se utilizan i-nodos de tercer nivel.

El i-nodo contiene toda la información necesaria sobre un archivo o directorio, salvo los datos que contiene. Por ejemplo, define los permisos, la fecha de creación, direcciones a los bloques de datos,etc.

Cada directorio guarda el nombre de los ficheros que contiene y la dirección del i-nodo correspondiente.

En este sentido, podremos crear dos tipos distintos de enlaces a ficheros: *enlaces duros* y *enlaces simbólicos*.

El comando **ln** nos servirá par crear ambos tipos de enlaces.

ENLACES DUROS (HARD LINKS)

Si utilizamos el comando ln sin especificar ninguna opción, por defecto crearemos un enlace duro. Los cambios que realicemos en el fichero enlazado o en el enlace, se reflejarán en el resto, ya que todos tendrán el mismo número de i-nodo, y por lo tanto hacen referencia a los mismos bloques del disco.

La sintaxis es la siguiente:

ln <fichero> [nombre del enlace]

ln <fichero1> <fichero2> ... <ficheroN> <nombre de directorio>

El fichero o ficheros para los que deseamos crear un enlace duro deberán existir.

Si el último argumento es el nombre de un directorio que existe, crearemos un enlace duro a cada fichero, dentro del directorio, y con el mismo nombre de fichero.

Si solamente especificamos el fichero que queremos enlazar, y no indicamos ningún nombre para el enlace, éste se creará con el mismo nombre que el fichero a enlazar.

La ventaja de utilizar enlaces duros radica en que el comando rm únicamente borrará aquel fichero que le indiquemos. La información solamente se borrará por completo cuando borremos todos los enlaces a un i nodo.

La desventaja con respecto a los enlaces simbólicos es que sólo permite crear enlaces dentro del mismo sistema de ficheros.

Los directorios **.** y **..** son enlaces duros al directorio actual y a su directorio padre respectivamente.

Ejemplos:

1 – Creamos el fichero *pruebaln* con la orden cat.

```
$cat >pruebaln
hola
CTRL+ D
```

2 – Creamos un enlace a *pruebaln* que se llame *penlace*.

```
$ln pruebaln penlace
```

3 – Veamos las características de estos ficheros con la orden ls. Utilizamos la opción **–i** para ver el número de inodo. Ambos tendrán el mismo número de inodo con dos enlaces.

```
$ls –i pruebaln penlace
```

4 – Modificamos *pruebaln* y comprobamos si también se modifica *penlace*.

```
$cat >>pruebaln
adios
CTRL+ D
$cat pruebaln
$cat penlace
```

5 – Modificamos *penlace* y comprobamos si también se modifica *pruebaln*.

```
$cat >>penlace
otra vez hola
```

CTRL+ D
$cat penlace

$cat pruebaln

6 – Eliminamos *pruebaln* y comprobamos si *penlace* permanece y contiene la información correspondiente.

$rm pruebaln

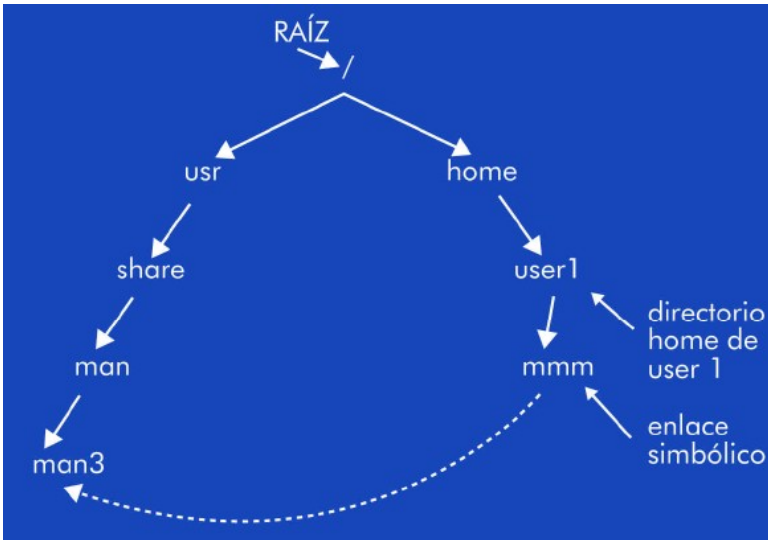
$cat penlace

7 – Si utilizamos la orden ls –i, vemos que *penlace* sigue con el mismo número de inodo, que ahora solamente tendrá un enlace.

ENLACES SIMBÓLICOS

Un enlace es un puente a un archivo o directorio perteneciente al sistema; una referencia que podemos poner en cualquier sitio que nos interese y que actúa como un accesodirecto a cualquier otro. Este mecanismo nos permite acceder a carpetas o archivos de forma más rápida y cómoda, sin tener que desplazarnos por la jerarquía de directorios.Este mecanismo nos permite acceder a carpetas o archivos de forma más rápida y cómoda, sin tener que desplazarnos por la jerarquía de directorios.

Vamos a verlo con un ejemplo: imaginemos que somos un usuario (user1) que necesita acceder frecuentemente al directorio `/usr/share/man/man3/`. En lugar de escribir el largo comando que nos situaría en el directorio en cuestión cada vez que necesitáramos desplazarnos a él, podemos crear un enlace en nuestro propio directorio que nos redireccione directamente hacia allí. El comando "`ln -s /usr/share/man/man3 mmm`" nos crearía este puente, que hemos llamado "`mmm`". El usuario sólo debería escribir (desde su directorio *home*) "`cd mmm`" y automáticamente el sistema lo redirigiría hacia `/usr/share/man/man3/`. Es importante tener en cuenta que al hacer un "`cd ..`" para ir al directorio superior, volveríamos al directorio *home* y no a `usr/share/man/`, ya que hemos accedido a él a partir de nuestro enlace. Podemos ver este esquema de forma gráfica en la siguiente figura:



Si utilizamos la opción –s con el comando ln, es decir **ln –s**, crearemos un enlace simbólico. La sintaxis en este caso es la misma que utilizamos para crear enlaces duros.

Podemos encontrar una similitud entre este tipo de enlaces y los accesos directos que estamos acostumbrados a crear con los sistemas Windows.

En el caso de los enlaces simbólicos, cada fichero tendrá un número de i-nodo distinto, es decir, una tabla ubicada en un lugar distinto a la del fichero original, que guarda como información la entrada en el directorio del i-nodo del fichero original. Por lo tanto, si desaparece el fichero original, el enlace simbólico perderá toda la información. Sin embargo, podremos crear enlaces simbólicos a ficheros de otros sistemas de archivos.

Ejemplos:

1 – Aún tenemos el fichero *penlace*. Creamos un enlace duro a *penlace* que se llame *pruebaln*.

$ln penlace pruebaln

2 – Con la orden ls –li vemos que ambos tienen el mismo inodo, y que este inodo tiene dos enlaces.

$ls -li pruebaln penlace

3 – Creamos un enlace simbólico a *penlace* que se llame *penlacesim*.

$ln –s penlace penlacesim

4 – Con la orden ls –li vemos que tienen distinto número de inodo. Además, el inodo de *penlacesim* sólo tiene un enlace, y el inodo de *penlace* sigue teniendo dos. En la línea correspondiente a *penlacesim* vemos que aparece el fichero al que apunta, y la letra l (“ele”)al inicio.

$ls -li pruebaln penlace penlacesim

5 – Cambiamos *penlace* y comprobamos si cambia *penalcesim*.

$cat >>penlace
otra vez adios
CTRL+ D

$cat penlacesim

6 – Por último borramos *penlace*. Comprobamos que *pruebaln* permanece y que no podemos ver el contenido de *penlacesim*, el sistema nos dirá que no existe. Para que desaparezca totalmente tenemos que borrarlo.

$rm penlace
$cat pruebaln
$cat penlacesim
$rm penlacesim

$rm pruebaln

9 – EL DIRECTORIO RAÍZ: ESTRUCTURA JERÁRQUICA DEL SISTEMA

Si observamos el contenido del directorio raíz veremos la jerarquía del sistema. Los principales directorios y subdirectorios son los siguientes.

- **/bin**: contiene archivos binarios (ejecutables) esenciales del sistema; en él encontramos los comandos básicos que usarán todos los usuarios.
- **/boot**: contiene los archivos que necesita el administrador de arranque; puede o no contener al núcleo, si no lo contiene se encontrará en el directorio raíz.
- **/dev**: contiene los archivos de dispositivos del sistema, es decir, los controladores de dispositivos. Algunos de los dispositivos más usados son:
  - Los que comienzan por fd son controladores de disqueteras. Por ejemplo, fd0H1440 permite acceder a discos de 3,5’’ de alta densidad en la primera disquetera; fd1 la segunda disquetera.

- Los que comienzan por hd acceden a canales IDE. Por ejemplo, hda hace referencia al dispositivo maestro del IDE primario y hda1 a su primera partición; hdb hace referencia al dispositivo esclavo del IDE primario y hdd al dispositivo esclavo del IDE secundario.
- Los que comienzan por sd hacen referencia a dispositivos de almacenamiento extraíbles.
- Los que comienzan por lp acceden a los puertos paralelo. Por ejmplo, lp0 es el equivalente a LPT1 de MS-DOS.
- Los que comienzan por tty hacen referencia a las consolas virtuales del sistema.
- El fichero /dev/null es usado como “agujero negro”, cualquier dato enviado a este dispositivo desaparece.

- **/etc**: contiene todos los ficheros de configuración del ordenador en el que está instalado el sistema. Veremos algunos como /etc/passwd (base de datos de usuarios), /etc/fstab (información sobre los distintos sistemas de ficheros), /etc/lilo.conf (configuración del cargador de arranque LILO).
- **/home**: contiene todos los directorios personales de los usuarios del sistema.
- **/lib**: contiene las librerías compartidas, es decir, ficheros cuyo código compartirán muchos programas. Estas rutinas compartidas se guardan en un lugar común, de forma que los programas ejecutables son menores y se reduce el espacio usado en disco. Además, en /lib/modules se encuentran los módulos del núcleo.
- **/mnt**: contiene los puntos de montaje para los sistemas de archivos temporales.
- **/media**: montaje de dispositivos.
- **/proc**: es un sistema de ficheros virtual, de forma que los ficheros que contiene realmente residen en memoria. Estos ficheros nos permiten obtener información sobre los programas y procesos que están corriendo en un momento dado.
- **/root**. directorio personal del root.
- **/sbin**: contiene los archivos binarios esenciales para arrancar el sistema. Sólo puede utilizarlos el root aunque un usuario no privilegiado también podrá ejecutarlos.
- **/tmp**: lugar habitual en el que se guarda la información temporal que necesitan generar muchos programas.
- **/opt**: aplicaciones que se instalan en un único directorio.
- **/usr**: es estático y compartible. Es el directorio principal de almacenamiento de las aplicaciones que no son necesarias para el arranque o mantenimiento del sistema, pero que hacen que el sistema sea útil e interesante. Algunos subdirectorios importantes son:
  - X11R6: contiene los ejecutables y las bibliotecas necesarias para que funcione el sistema X Window. La configuración que no varía de una máquina a otra se encuentra en este subdirectorio. La configuración específica de cada máquina está en /etc/X11.
  - bin: contiene la gran mayoría de los programas ejecutables del sistema, aquellos que no son necesarios para su mantenimiento y administración. No estarán aquellos que instalemos nosotros, que debemos ubicar en /usr/local.
  - lib: en este directorio debemos instalar nuestras aplicaciones.
  - share: entre otros, están los subdirectorios doc y man, que contienen documentación de las aplicaciones y las páginas Man del sistema.

- **/var**: contiene datos sobre los programas que se están ejecutando. Por ejemplo, en él se guarda el correo electrónico, los archivos a imprimir, etc.

10 – LA EDICIÓN DE TEXTO. EL EDITOR vi

En Linux existen muchos editores de texto disponibles (vi, Emacs, joe), sin embargo será el *visual editor* (vi) el único que siempre encontraremos en cualquier sistema Unix.

vi fue el primer editor de pantalla completa que existió y, aunque no es fácil de usar, es una herramienta extremadamente potente. La versión proporcionada con Mandrake es una versión mejorada de *vi* (vim – VI iMproved).

Para comenzar con *vi* y editar un fichero de texto emplearemos la sintaxis:

$vi <nombre de fichero>

En la pantalla, de 24 líneas, aparecerá una columna de "~" que indican el final del fichero. En la parte inferior veremos el nombre del nuevo fichero.

En un principio no podremos insertar texto, ya que *vi* arranca en el modo órdenes, uno de los tres posibles *modos de operación*: modo órdenes, modo inserción, modo última línea.

- En modo órdenes o modo comandos no podremos insertar texto. Nos permitirá usar ciertas órdenes de edición de ficheros o cambiar a otros modos.
- Al modo de inserción, que nos permitirá escribir y desplazarnos por el archivo, se accede desde el modo comando por ejemplo con la orden **i**. Para volver al modo comando pulsamos la tecla *Esc*.
- El modo última línea, o modo ex, proporciona ciertas órdenes extendidas a *vi*, como por ejemplo salir de *vi* guardando o sin guardar los cambios realizados en el archivo (:wq :q!). Para acceder a este modo, tecleamos **:** desde el modo comando. Para salir de él ejecutamos una orden o borramos todo, incluidos los dos puntos.

INSERTAR TEXTO

Si estamos en modo órdenes podemos pasar al modo de inserción de varias formas:

- Tecla **i**: para insertar texto desde la posición en la que se encuentra el cursor.
- Tecla **a**: para insertar texto comenzando detrás de la posición actual del cursor.
- Tecla **A**: para insertar texto comenzando al final de la línea actual.
- Tecla **I** (i mayúscula): para insertar texto comenzando al inicio de la línea actual.
- Tecla **o**: para insertar texto debajo de la línea actual.
- Tecla **O**: para insertar texto por encima de la línea actual.

En la parte inferior de la pantalla aparecerá la cadena –INSERT—indicándonos que estamos en el modo de inserción. Podremos borrar y suprimir texto, además de movernos por el archivo con las flechas del cursor.

BORRAR TEXTO

Además de las teclas de retroceso y suprimir, podemos utilizar otras órdenes para borrar desde el modo comando:

- Tecla **x**: borra el carácter en el que se encuentra situado el cursor.
- Tecla **X**: borra el carácter que está a la izquierda del cursor.
- Teclas **dd**: borra la línea en la que se encuentra el cursor.
- Tecla **dw**: borra la palabra en la que se encuentra el cursor.

- Tecla **o**: para insertar texto debajo de la línea actual.
- Tecla **O**: para insertar texto por encima de la línea actual.

MODIFICAR TEXTO

Desde el modo comando podemos reemplazar o sustituir parte del texto:

- Tecla **r**: permite sustituir el carácter en el que se encuentra el cursor.
- Tecla **R**: en la parte inferior de la pantalla aparecerá la cadena –REPLACE--, que nos indica que podemos reemplazar el texto hasta que pulsemos la tecla *Esc*. Es decir, R es similar al modo de inserción, con la diferencia de que en lugar de insertar texto lo sobrescribe.
- Teclas **:r <fichero>**: inserta en el fichero que estamos editando el contenido del fichero que indicamos.
- Tecla **~**: cambia de mayúsculas a minúsculas, o viceversa, el carácter en el que se encuentra el cursor (F10 cambia uno, F11 cambia tres, F12 cambia cuatro).

ÓRDENES DE DESPLAZAMIENTO

Además de las flechas del cursor, podemos movernos por el documento desde el modo comando utilizando una serie de órdenes:

- Tecla **h**: un carácter a la izquierda.
- Tecla **j**: un carácter abajo.
- Tecla **k**: un carácter arriba.
- Tecla **l** (ele minúscula): un carácter a la derecha.
- Tecla **e**: al final de la palabra actual.
- Tecla **b**: al inicio de la palabra actual.
- Tecla **w**: al inicio de la palabra siguiente.
- Tecla **0** (cero): al inicio de la línea actual.
- Tecla **$**: al final de la línea actual.
- **/<cadena>**: desplaza el cursor hacia delante hasta que encuentra el texto *cadena*.
- **?<cadena>**: desplaza el cursor hacia atrás hasta que encuentra el texto *cadena*.
- Tecla **H**: va al comienzo del archivo.
- Tecla **G**: va al final del archivo.
- **CTRL+ f**: avanza una pantalla.
- **CTRL+ b**: va una pantalla hacia atrás.

Cada uno de los comandos de movimiento puede estar precedido por un número, de forma que tenemos la posibilidad de movernos a una palabra, línea o carácter arbitrarios.

Además, podemos asociar órdenes de desplazamiento con otras órdenes como por ejemplo borrar.

Ejemplos:

10G: va a la línea 10 del fichero.

dG: borrará todo, desde la posición del cursor hasta el final del fichero.

d$: borrará todo desde la posición del cursor hasta el final de la línea.

3e: moverá el cursor tres palabras hacia delante.

d3b: borrará tres palabras hacia atrás.

d/<cadena>: borra todo desde la posición del cursor hasta que encuentra el texto cadena.

d0: borra todo hasta el inicio de la línea actual.

CORTAR, COPIAR Y PEGAR

Utilizaremos las órdenes **y** (Yank) y **d** (Delete) para copiar y cortar texto respectivamente.

Combinaremos estas dos órdenes con las de desplazamiento para copiar o cortar conjuntos de caracteres, líneas, palabras.

Para pegar el texto que hemos copiado o cortado utilizaremos las órdenes p (para insertar el texto después del cursor) y P (para insertar el texto antes del cursor).

Ejemplos:

y?<cadena>: copiará todo desde la posición del cursor hacia atrás, hasta que encuentre el texto cadena.

d15l: cortará 15 caracteres desde la posición del cursor hacia la derecha.

y$: copiará todo desde la posición del cursor hasta el final del párrafo actual.

GUARDAR Y SALIR

Para salir sin guardar los cambios escribimos **:q!**.

Para salir guardando los cambios escribimos **:wq** o :**ZZ** o **:x**.

Para guardar los cambios sin salir escribimos **:w**.

EDITAR OTROS FICHEROS

Si estamos editando un fichero con *vi*, podemos editar otro escribiendo **:e <fichero>** desde el modo comando. Para poder utilizar esta orden tendremos que indicarle a *vi* si queremos guardar o no los cambios del primer fichero; es decir, utilizaremos **:w** y luego **:e**, o bien **:e!** Directamente si no queremos guardar los cambios. Dejaremos de editar el primero y pasaremos al segundo.

EJECUTAR COMANDOS DEL INTÉRPRETE

Podemos insertar, en el fichero que estamos editando, la salida de un comando. Para ello utilizamos la orden **:r!** y a continuación el comando que queremos ejecutar. Por ejemplo,

:r! ls –i

inserta un listado del directorio actual con números de inodo al final del párrafo actual.

También podemos ejecutar una orden desde *vi* y volver al editor una vez que ésta finalice. Utilizaremos la orden **:!**. Por ejemplo,

:! ls –i

mostrará en pantalla el mismo listado que en el ejemplo anterior, aunque en este caso los resultados no se insertarán en el fichero.

Incluso podemos dejar temporalmente *vi* e iniciar el intérprete de comandos para ejecutar otras órdenes. Para salir del intérprete y regresar a *vi* utilizamos la orden **exit**. Para iniciar el intérprete usamos la orden **:shell**. Por ejemplo, es posible que queramos consultar la página de manual de *vi* y guardarla en un fichero.

11 – ARCHIVAR Y COMPRIMIR FICHEROS

El comando **tar** es utilizado normalmente para archivar o empaquetar ficheros. Su sintaxis general es:

tar <opciones> <ficheros>

El modo en el que se escriben las opciones de *tar* es un poco especial. El guión inicial, por ejemplo, no es necesario.

Las opciones más comunes para *tar* son:

-c: creación de archivadores nuevos.

-x: extracción de archivos de un archivador existente.

-v: muestra los archivos mientras se agregan o se extraen.

-t: muestra el contenido de un archivo tar.

-f: el siguiente argumento es el archivador a crear, del que queremos extraer archivos o mostrar un listado.

-: el archivador a crear es asociado con la salida (creación) o la entrada (extracción) estándar.

Ejemplos:

tar cvf escritorio.tar Desktop empaqueta el contenido de *Desktop* en un archivador nuevo *escritorio.tar*

#tar xvf escritorio.tar Desktop/Floppy.desktop extrae del archivo *escritorio.tar* el fichero indicado

#tar xvf escritorio.tar extrae todo el contenido del archivo *escritorio.tar*

#tar tvf escritorio.tar muestra un listado largo del contenido del archivo *escritorio.tar*

Hay que tener en cuenta, a la hora de extraer el contenido de un archivador, si el archivador se creó conservando el nombre del directorio de origen. Es posible que se sobrescriba el contenido de los ficheros originales.

Ejemplo: Nos situamos en el directorio raíz como root. Si archivamos los ficheros /etc/group y /etc/passwd:

#tar cvf backup.tar /etc/group /etc/passwd

estamos conservando los nombres del directorio al que pertenecen. Por lo tanto, para extraer estos ficheros nos tendremos que situar en el directorio raíz:

#cd /

#tar xvf backup.tar /etc/group /etc/passwd

Sin embargo, si archivamos los ficheros *group* y *passwd* estando en /etc:

#tar cvf /backup.tar group passwd

No guardamos la ruta, por lo que para extraer los ficheros tendremos que situarnos en ella:

#cd /
#cd /etc

#tar xvf /backup.tar group passwd

COMPRIMIR/DESCOMPRIMIR

Para comprimir un fichero utilizamos el comando **gzip** y para descomprimirlo **gunzip**:

gzip/gunzip <opciones> <ficheros>

Las opciones más comunes son:

-1 a –9: grado de compresión, mínimo y máximo respectivamente.

-c: envía el resultado de la compresión a la salida estándar.

Ejemplo: gzip –9 backup.tar (comprime el fichero backup.tar y le añade .gz al final).

gunzip backup.tar.gz (descomprime el fichero backup.tar.gz y obtenemos otro sin .gz).

COMBINAR TAR Y GZIP/GUNZIP

Para archivar un conjunto de ficheros y comprimir el resultado debemos usar en primer lugar el comando tar y a continuación el comando gzip.

Igualmente debemos realizar las operaciones inversas para descomprimir con gunzip, y desempaquetar con tar.

Sin embargo, podemos realizar ambas operaciones en una sola línea de comandos usando el redireccionamiento y tuberías.

Ejemplo:

- Nos situamos en */etc*

  #tar cvf- group passwd |gzip –9c >/backup.tar.gz

  En este ejemplo, tar va a empaquetar los ficheros *group* y passwd. El resultado lo envía mediante "-" a la salida estándar. Mediante la tubería, lo que tar ha enviado a la stdout lo recibe gzip de la stdin. Esto que recibe, el resultado de tar, lo comprime y lo envía a la salida estándar mediante "c". Finalmente, a través del redireccionamiento, enviamos la stdout de gzip al fichero backup.tar.gz.
- Nos situamos en el directorio raíz:

  #gunzip –c backup.tar.gz |tar xvf –

  En este caso, gunzip descomprime *backup.tar.gz* y el resultado lo envía a la stdout (-c). El comando tar hace uso de la entrada estándar para desempaquetar su contenido.

Afortunadamente, la orden tar proporciona la opción **z** para comprimir y descomprimir o desempaquetar.

Ejemplo:

#tar cvfz backup.tar.gz passwd group

#tar xvfz backup.tar.gz

12 – CONTROL DE TAREAS Y PROCESOS

El control de tareas es una utilidad incluida en la shell que permite el control de multitud de comandos o tareas al momento.

Cada vez que se ejecuta un programa, se lanza lo que se conoce con el nombre de *proceso*, que es simplemente el nombre que se le da a un programa que se está ejecutando.

El comando **ps** visualiza la lista de procesos que se están ejecutando actualmente. Con la opción **–e** veremos todos los procesos del sistema.

Ejemplo:

PID	TTY	TIME	CMD
1523	tty1	00:00:00	bash
4757	tty2	00:00:00	ps

Cada proceso tiene un número PID que lo representa (Process ID), y el comando ps lo muestra en la primera columna. La segunda columna representa la consola virtual en la que se está ejecutando cada proceso. La última columna, CMD, contiene el nombre del comando ejecutado.

PRIMER PLANO Y SEGUNDO PLANO

Un proceso puede estar en Primer plano o en Segundo plano. Sólo puede haber un proceso en primer plano al mismo tiempo, que será el que interactúe con nosotros, es decir el que recibirá las entradas del teclado y enviará las salidas al monitor. El proceso en segundo plano no recibirá ninguna señal desde el teclado.

Algunos programas necesitan mucho tiempo para terminar, por ejemplo compilar o realizar copias de seguridad de gran volumen de datos. Será mejor lanzar estos programas en segundo plano para dejar el ordenador en condiciones de ejecutar otro programa.

Además, los procesos pueden ser suspendidos. Un proceso suspendido es aquel que no se está ejecutando actualmente, sino que está temporalmente parado. Después de suspender una tarea podemos indicar a la misma que continúe, en primer o segundo plano según necesitemos y la tarea continuará ejecutándose justo donde se dejó.

Debemos tener en cuenta que suspender un trabajo no es lo mismo que interrumpirlo. Cuando se interrumpe un proceso, generalmente con la combinación de teclas CTRL+ C, el proceso "muere", deja de estar en memoria y utilizar recursos del ordenador.

ENVÍO A SEGUNDO PLANO Y ELIMINACIÓN DE PROCESOS

Una forma de enviar procesos a segundo plano, y así liberar el teclado para poder seguir haciendo otras cosas, es añadir el carácter **&** al final del comando. Al hacerlo, la shell nos muestra en pantalla la siguiente información:

[Número de tarea] PID

El número entre corchetes es el número de tarea del proceso que hemos lanzado, y el siguiente número es el identificador que el sistema le ha asignado. Ambos números pueden utilizarse para referirnos a la tarea.

Para chequear el estado del proceso utilizamos el comando **jobs**.

Para eliminar una tarea utilizamos el comando **kill**. Este comando toma como argumento un número de tarea precedido del símbolo **%**, o bien un número de ID de un proceso.

Una vez que hemos matado la tarea podemos usar nuevamente el comando jobs para comprobarlo. Si lo utilizamos por segunda vez no mostrará nada.

PARADA Y RELANZAMIENTO DE TAREAS

También podemos poner una tarea en segundo plano lanzándola como un proceso normal en primer plano, seguidamente pararla y finalmente relanzarla en segundo plano.

Para suspender una tarea utilizamos la combinación de teclas **CTRL+ Z**.

Para relanzar en primer plano una tarea que fue suspendida utilizamos el comando **fg** (foreground).

Para relanzar en segundo plano una tarea que fue suspendida utilizamos el comando **bg** (background).

También podemos emplear el comando fg para traer a primer plano una tarea que está en segundo plano. Sin embargo, no podremos suspender una tarea que está en segundo plano utilizando CTRL+ Z. Será necesario traerla antes a primer plano.

Si no indicamos lo contrario, los comandos fg y bg actúan sobre el último proceso parado (indicado por + junto al número de tarea cuando utilizamos jobs). Si tenemos varios procesos corriendo a la vez, podemos pasar a primer o segundo plano una tarea específica indicando el número de tarea como argumento de fg o bg precedido del símbolo % (no podremos utilizar el ID de los procesos).

13 – GESTIÓN DE SISTEMAS DE FICHEROS

A diferencia de Windows, Linux no asocia una letra a cada uno de los sistemas de archivos. Linux tiene una estructura de árbol de archivos única, y cada sistema de archivos está montado en algún lugar de la estructura del árbol. Es decir, Linux monta la raíz de su sistema en la partición que contiene al sistema de archivos raíz, y a partir de ahí puede montar otros sistemas de archivos en la estructura de árbol sobre diferentes puntos de montaje.

Debemos tener en cuenta que:

- En principio, solamente el administrador del sistema root podrá montar un sistema de archivos.
- El directorio que actúa como punto de montaje debe existir.
- Es preferible que el punto de montaje esté vacío, puesto que el sistema ocultará todos los ficheros, excepto los del sistema montado, hasta que desmontemos.

ACCESO A UNIDADES DE DISCO

Como ya vimos en el apartado 9, los controladores correspondientes a los dispositivos del sistema se encuentran en el directorio */dev*. Por lo tanto, para acceder a las distintas unidades de disco, montaremos el dispositivo correspondiente del directorio */dev* e indicaremos el punto de montaje. Además, le diremos al sistema qué tipo de sistema de archivos queremos montar.

Para realizar estas operaciones utilizaremos el comando **mount**. Su sintaxis básica es la siguiente:

#mount –t <tipo> <dispositivo> <punto de montaje>

-t tipo: indica el tipo de sistema de archivos que usa la unidad de disco para guardar los ficheros. Los tipos más usuales son:

- iso 9660 para CD ROM.
- vfat para disquete y FAT.
- msdos para particiones de MS DOS.
- ext2 para particiones de Linux.

<*dispositivo*>: indica el dispositivo que se va a montar, y que estará representado por un archivo en directorio /dev.

<*punto de montaje*>: es el directorio en el que se pondrá a disposición del usuario el contenido de la unidad montada. Por regla general, se utiliza el directorio */mnt*, que contiene una serie de subdirectorios relativos a los diferentes sistemas de archivos de que dispone el sistema. De todas formas, el usuario siempre puede crear un directorio vacío con el nombre que él elija para montar las unidades de disco que desee donde desee.

Para montar el disquete o el CD ROM deberán estar insertados en la unidad los soportes correspondientes.

Ejemplos:

<u>Montaje del disquete</u>

#mount –t vfat /dev/fd0 /mnt/floppy

Para ver el contenido del disquete accedemos al directorio */mnt/floppy*.

<u>Montaje del CD ROM</u>

#mount –t iso9660 /dev/cdrom /mnt/cdrom

Para ver el contenido del CD ROM accedemos al directorio /mnt/cdrom

<u>Montaje de la primera partición de Windows</u>

#mount –t vfat /dev/hda1 /mnt/win_c

Cuando el usuario haya dejado de usar la unidad, deberá desmontarla. Para ello utilizamos la orden **umount**. En el caso del CD ROM y el disquete, habrá que desmontar las unidades antes de sacarlos. La sintaxis de umount es:

#umount <punto de montaje>|<dispositivo>

Ejemplos:

#umount /mnt/floppy (o /dev/fd0)

#umount /mnt/cdrom (o /dev/cdrom)

#umount /mnt/win_c (o /dev/hda1)

Si accedemos al escritorio, podemos utilizar iconos de acceso directo al CD ROM y al disquete. Para ello, en el menú contextual del escritorio seleccionamos “Crear nuevo-Dispositivo”. Elegimos la unidad y aparecerá un cuadro de diálogo con tres fichas:

- General: nombre del acceso directo y tipo de icono de montado.

- Permisos: para establecer los permisos del icono.
- Dispositivo: indicaremos el dispositivo de */dev*, el punto de montaje y el tipo de sistema de archivos. También podemos elegir el icono correspondiente al dispositivo desmontado.

Una vez que los tengamos sobre el escritorio, para montarlos y desmontarlos utilizamos su menú contextual. Aunque para montarlos bastará con hacer clic sobre ellos. En principio sólo root podrá hacerlo.

EL FICHERO /etc/fstab

El fichero */etc/fstab* contiene información descriptiva sobre los distintos sistemas de ficheros del sistema. Cada sistema de ficheros ocupa una línea de este fichero, y los campos de cada línea están separados por tabuladores o espacios.

Los comandos fsck, mount y umount actúan sobre él.

Para cada línea encontramos seis campos ordenados que son los siguientes:

- *Dispositivo*: especifica el nombre del dispositivo a montar. Ejemplo: /dev/hda1.
- *Punto de montaje*: especifica el punto de montaje para el sistema de ficheros. Ejemplo: /mnt/win_c.
- *Tipo*: especifica el tipo de sistema de ficheros. Ejemplo: vfat.
- *Opciones*: especifica las opciones de montaje asociadas al sistema de ficheros. Van separados por comas, y algunas de ellas son:
  - *auto/noauto*.- la partición se monta/no se monta en el arranque.
  - *user/nouser*.- los usuarios pueden/no pueden montar y desmontar el sistema de archivos.
  - *ro*.- el sistema de archivos se monta sólo para lectura.
  - *rw*.- el sistema de archivos se monta para lectura y escritura.
  - *exec*.- se pueden ejecutar archivos binarios en el sistema de archivos.
  - *defaults*.- se toman los valores por defecto que son rw, exec, auto, nouser.
- *Frecuencia:* determinamos la frecuencia con la que se deben realizar copias de seguridad del sistema correspondiente por el comando **dump**. El usuario recibirá un aviso. Si este campo está vacío quiere decir que no se realizará copia.
- *Secuencia:* es usado por el comando **fsck** (File System Check, verificación del sistema de archivos) para determinar el orden en que se realizan los chequeos de los sistemas de archivos en el arranque. Si el campo es cero, este sistema de ficheros no se chequeará.

En el fichero /etc/fstab siempre hay varias entradas especiales:

- Una línea para el sistema de archivos raíz.
- Una línea para el sistema de archivos */dev/pts*.
- Una línea para el sistema de archivos */proc*.
- Una línea para la partición *swap*. Estas particiones no son visibles en la estructura árbol, y el campo correspondiente al punto de montaje contiene la palabra clave swap.

Por otra parte, la característica *supermount* aparece incorporada en los núcleos más recientes de Linux. Instalada, esta característica monta o desmonta los dispositivos para unidades de almacenamiento extraíbles (disquete, CD ROM, usb...) automáticamente a medida que se insertan o remueven.

14 – GESTIÓN DE USUARIOS Y GRUPOS

Ya sabemos que Linux es un sistema multiusuario y por lo tanto distingue diferentes usuarios.

Cada usuario recibe una *cuenta* que incluirá toda la información necesaria (nombre de usuario, directorio inicial, etc.).

Además de las cuentas dadas a personas, existen cuentas especiales definidas por el sistema que tienen privilegios especiales. La más importante es la *cuenta raíz*, con el nombre de usuario *root*.

Normalmente, los usuarios normales están restringidos, de forma que los permisos de los ficheros en el sistema están preparados para que no puedan borrar o modificar ficheros en directorios compartidos por todos los usuarios.

Estas restricciones desaparecen para root. El usuario root puede leer, modificar o borrar cualquier fichero en el sistema, cambiar permisos y pertenencias, etc.

Por lo tanto, podemos deducir que la gestión de los usuarios solamente puede realizarla el usuario root.

CONCEPTOS DE GESTIÓN DE USUARIOS

La información que el sistema mantiene acerca de cada usuario es la siguiente:

- *Nombre de usuario:* es un identificador único dado a cada usuario del sistema. Es la cadena de caracteres con la que el usuario se identifica al entrar en el sistema. Se pueden utilizar letras, dígitos y los caracteres _ (guión bajo) y . (punto)..
- *User ID o UID:* es un número único dado a cada usuario del sistema. Su número debe ser mayor que el del último usuario creado en el sistema.
- *Group ID o GID:* número identifica el grupo al que pertenece el usuario. El número ha de ser el mismo para todos los usuarios que formen el grupo. Cada usuario puede pertenecer a uno o más grupos definidos por el administrador del sistema. Aunque la importancia real de las relaciones de grupo es la relativa a los permisos de ficheros.
- *Clave:* el sistema almacena la contraseña del usuario encriptada. El comando passwd nos permitirá asignar y cambiar las claves de los usuarios.
- *Nombre completo:* puede ser el nombre real del usuario, su número de teléfono, su dirección, etc.
- *Directorio inicial:* es el directorio al que accede el usuario al entrar en el sistema. Cada usuario debe tener su propio directorio inicial, normalmente situado bajo */home*. En principio será el único directorio en el que el usuario podrá guardar su información personal, programas, etc. Ejemplo: */home/usu1*.
- *Intérprete de inicio:* es el intérprete de comandos que arranca para el usuario cuando se conecta al sistema. Ejemplos: */bin/bash*, */bin/tcsh*.

El fichero que contiene toda esta información relativa a los usuarios es el fichero */etc/passwd*. Este fichero contiene una línea por cada usuario del sistema, y su estructura es la siguiente:

nombre:clave encriptada:UID:GID:nombre completo:directorio de inicio:intérprete

Ejemplo:

usu1:x:501:501:usu1:/home/usu1:/bin/bash

En el caso de los grupos, la información sobre ellos se encuentra en el fichero */etc/group*. Hay varios grupos definidos en el sistema (root, bin, sys, mail, etc) que se utilizan para permisos de ficheros del sistema. Los usuarios no deben pertenecer a ninguno de estos grupos.

El formato de cada línea del fichero */etc/group* es el siguiente:

nombre del grupo:clave:GID:otros miembros

La clave del grupo no suele utilizarse.

En */etc/passwd* cada usuario tiene un GID. Sin embargo, como los usuarios pueden pertenecer a otros grupos, podemos añadir su nombre de usuario en el campo *otros miembros* (separados unos usuarios de otros por comas) de todos aquellos grupos no definidos por el sistema a los que queremos que pertenezca.

Podemos conocer a qué grupos pertenece un usuario utilizando la orden **groups**.

AÑADIR NUEVOS USUARIOS Y BORRAR USUARIOS

Podemos añadir usuarios al sistema de varias formas. La más engorrosa de todas es hacerlo a mano. Es decir, añadimos al fichero */etc/passwd* la línea correspondiente al usuario. Seguidamente le asignamos una clave con la orden *passwd* y finalmente establecemos el dueño, grupo dueño y permisos para el directorio */home/usuario* que deberemos crear. Además, debemos actualizar */etc/shadow* con **pwconv**.

Useradd/Adduser

Las órdenes **useradd** y **adduser** también nos permiten añadir nuevas cuentas de usuario desde el modo consola.

Podemos crear un usuario con las características por defecto:

#useradd usuario

#adduser usuario

Habremos creado un usuario sin contraseña.

Si queremos crear un usuario a nuestra medida utilizaremos la siguiente sintaxis:

#useradd/adduser –r <nombre usuario> -p <clave> -u <UID> -g <GID> -c <información> -d <direct. Inicial> -s <shell>

Lo único que nos quedará por hacer será crear el directorio */home* del usuario y establecer los permisos pertinentes.

Para borrar un usuario desde el modo consola:

#userdel [-r] usuario

Si utilizamos la opción **–r** también eliminaremos el directorio home del usuario.

Una forma de deshabilitar una cuenta de usuario sin tener que borrarla es escribir **!** en el campo clave del usuario en el fichero */etc/shadow* o */etc/passwd*.

OTRAS ÓRDENES PARA LA GESTIÓN DE USUARIOS Y GRUPOS

chfn: permite cambiar el nombre completo del usuario.

#chfn –f <nuevo nombre> <usuario>

groups: muestra todos los grupos a los que pertenece el usuario.

groupadd: permite añadir un nuevo grupo.

#groupadd [-g GID] [-f] <nombre del grupo>

-f obliga al sistema a informar si se producen errores (por ejemplo cuando el grupo que queremos crear ya existe).

Si no especificamos un GID, el sistema asigna el menor GID que corresponde a este grupo.

groupdel: borra el grupo cuyo nombre indiquemos junto a la orden.

groupmod: permite modificar el GID y el nombre del grupo.

#groupmod [-g <nuevo GID>] [-n <nuevo nombre>] <nombre actual>

who: muestra información sobre los usuarios conectados al sistema.

id: muestra UID y GID del usuario y los grupos a los que pertenece.

#id

#id usuario

16 – GUIONES DEL INTÉRPRETE DE COMANDOS

Los intérpretes de comandos en Linux, además de meros intérpretes, son también poderosos lenguajes de programación. Para la ejecución de guiones, cada intérprete de comandos usa una sintaxis diferente. Nosotros supondremos que los guiones son escritos con la sintaxis del intérprete de comandos Bourne shell.

El funcionamiento de la shell es el siguiente:

1. Lee la entrada desde teclado o desde un fichero.
2. Divide la entrada en palabras y operadores, obteniendo los comandos.
3. Realiza las expansiones correspondientes y las redirecciones de salida.
4. Ejecuta las órdenes.
5. Espera a que terminen las órdenes y devuelve un valor de estado de finalización. El valor de estado 0 (cero) significa finalización sin errores y un valor distinto de cero indica el código de error producido.

Si a menudo utilizamos una serie de comandos, podríamos agruparlos en una sola orden en lugar de teclearlos todos cada vez que queremos ejecutarlos. Así, podríamos agruparlos en un *guión del intérprete de comandos*. Los guiones son simples ficheros de texto que podremos crear con un editor como *vi*.

Cuando creamos un guión, debemos asegurarnos de que tiene permisos de ejecución para poder ejecutarlo, ya que los editores de texto guardan por defecto los ficheros sin permisos de ejecución. Por lo tanto, tendremos que utilizar la orden *chmod* para cambiar los permisos.

Además, si estamos trabajando con un intérprete de comandos distinto al que utilizamos para escribir el guión, debemos indicar en la primera línea cuál es el intérprete que debe ejecutar el guión. Es decir, si escribimos un guión utilizando la sintaxis de Bourne shell, especificaremos la ruta del programa de la siguiente forma:

#!/bin/bash

Si, además, queremos incluir un comentario en el guión para explicar qué tareas realiza, podemos utilizar el carácter **#** al inicio de la línea. El resto de líneas podrán ser comandos o sentencias del lenguaje de programación del intérprete.

Los guiones podrán recibir argumentos que serán reemplazados en la ejecución del guión por los parámetros correspondientes. Los *parámetros posicionales* que podemos utilizar son **$0**, **$1**, **$2**, ..., **$9**, teniendo en cuenta que si nos hacen falta más parámetros siempre disponemos de la orden **shift**.

Ejemplo:

```
$cat >primero
#!/bin/bash
#Vamos a ver la fecha y hora actuales
echo Fecha y hora actuales:
date
#y guardamos los usuarios conectados en este momento
#ordenados alfabéticamente en un fichero pasado como argumento
who |sort >$1
CTRL+ D
```

VARIABLES DE ENTORNO DE BASH

El intérprete de comandos utiliza ciertas variables ya predefinidas y nos permite definir las nuestras. Para definir una variable escribimos el nombre de la variable seguido del signo igual y su valor. Por ejemplo:

#LISTAR=ls

Para hacer referencia al valor de la variable anteponemos el carácter **$** a su nombre:

#$LISTAR (así estamos ejecutando el comando ls).

Podemos ver el valor de todas las variables de entorno definidas en nuestra shell con el comando **set**.

En principio, sólo el intérprete podrá acceder a las variables. Sin embargo, nos permite *exportar* variables al *entorno*, de forma que todas las órdenes que ejecutemos tendrán acceso a las variables de dicho entorno. Sólo será necesario exportar las variables una vez; cualquier cambio efectuado con posterioridad será automáticamente propagado al entorno.

Algunas de las variables que utiliza bash son las siguientes:

HISTFILE	Nombre del archivo histórico de comandos ejecutados.
HISTFILESIZE	Número de órdenes guardadas en HISTFILE.
HOME	Directorio HOME.
HOSTNAME	Nombre de la máquina.
LOGNAME	Nombre del usuario con el que estamos conectados.
PAGER	Contiene el comando usado por *man* para paginar la salida.
PATH	Contiene los caminos donde buscar los programas ejecutables, separados por **:** (dos puntos). Si contiene **.** (un punto) buscará en el directorio de trabajo actual.
PS1	Contiene la composición del prompt principal. PS1='[\u@\h \W]\$' (prompt definido por defecto en nuestra máquina) Parámetros: \u (usuario) \h (nombre de la máquina) \W (directorio actual) \$ (# si se trata de root, $ si no es usuario privilegiado)
PS2	Contiene la composición del prompt secundario. PS2='>'
SHELL	Nombre de la shell que se está ejecutando actualmente.

GUIONES DE INICIALIZACIÓN DEL INTÉRPRETE

A parte de los guiones que podemos crear, existen algunos que son utilizados por el intérprete de comandos para ciertos propósitos. Entre ellos se encuentran los guiones de inicialización, que son guiones ejecutados automáticamente por el intérprete al abrir una sesión.

Los ficheros de inicialización usados por bash son:

- */etc/profile*: es configurado por el administrador del sistema y ejecutado por todos los usuarios de bash en el momento de la presentación al sistema. En este fichero podemos ver las variables de entorno.
- *$HOME/.bash_profile*: ejecutado por una sesión de presentación de bash y configurable por el propio usuario.
- *$HOME/.bashrc*: ejecutado por todas las sesiones bash que no son de presentación.

Por otra parte, el fichero *$HOME/.bash_history* suele ser utilizado como fichero histórico de órdenes, y el fichero $HOME/.bash_logout es ejecutado por el sistema al cerrar la sesión correspondiente.

Para entender completamente la función de estos ficheros es necesario conocer la programación del intérprete de comandos.