

# Sistema operativo multiusuario Unix/Linux. Gestión de archivos, directorios y usuarios

13

En esta Unidad aprenderás a...

- 1 Utilizar uno de los sistemas operativos multiusuario por excelencia: Unix.
- 2 Manejar tanto el sistema operativo base: Unix, como las versiones de libre distribución: Linux.
- 3 Manejar comandos de administración de ficheros y directorios, tanto en modo texto o consola como en modo gráfico.
- 4 Establecer conexiones desde otros equipos con el equipo servidor a través del comando telnet.
- 5 Gestionar usuarios, grupos y servicios de impresión.



## Introducción

En esta Unidad se describen las características más importantes de los sistemas operativos Unix y Linux. En concreto la versión Unix de **SCO Sistema V versión 4 (Open Server)** y la versión Linux **Suse 9.1**, aunque la mayoría de las características son válidas para otras versiones.

Actualmente la mayoría de los fabricantes de hardware y software suelen tener su propia versión de este sistema operativo, a la que denominan de diferentes formas. La

base es la misma, los comandos en muchos casos coinciden y, lo que es más importante, todos estos sistemas operativos se instalan en ordenadores que serán servidores de red.

De hecho, en la actualidad la mayoría de los Servidores de Internet instalan Unix como sistema operativo, debido especialmente a que es muy estable y tiene pocos fallos.

Linux es un sistema operativo de libre distribución. Tiene la misma filosofía que Unix, aunque incorpora otros programas de configuración, además de los básicos.

## 13.1 Características generales de Unix/Linux

El sistema operativo Unix/Linux es uno de los sistemas multiusuario por excelencia. A lo largo de su evolución histórica se le ha llamado de muchas formas y ha tenido multitud de versiones, pero en definitiva siempre ha estado especializado en gestionar software para que pueda ser utilizado simultáneamente por varios usuarios.

En general, el hardware de cualquier sistema multiusuario consta de dos partes fundamentales:

- **Ordenador central.** También llamado servidor, es el encargado de suministrar información a los diferentes usuarios del sistema.
- **Terminales.** Éstos son los puestos con los que los usuarios se comunican con el ordenador central. Pueden ser, a su vez, de dos tipos:
  - *Terminales puros.* Estos terminales son exclusivamente un monitor y un teclado sin unidad central, que están conectados directamente al ordenador central. Solamente pueden ser utilizados en el sistema en el que están conectados, es decir, no son autónomos, ya que no tienen microprocesador propio, ni memoria, ni ningún componente básico de un ordenador.

- *Ordenadores personales en emulación.* Pueden ser ordenadores de la familia 8086, 486-SX, Pentium, etc. Son autónomos, es decir, pueden trabajar de forma independiente, ya que constan de todos los componentes que cualquier ordenador precisa: microprocesador, memoria, monitor, teclado, ROM, etc. Pueden ser utilizados por los usuarios de un sistema multiusuario, ya que se pueden conectar al ordenador central mediante tarjetas de red (también pueden conectarse a través de los puertos serie). Una vez conectados, el equipo ejecuta un programa que *emula* o transforma la señal recibida del ordenador central, para que éste la entienda y pueda funcionar.

Respecto al sistema de almacenamiento, los sistemas como Unix/Linux permiten que más de un disco pueda formar lo que se denomina un sistema de archivos.

Otro componente que forma parte de un sistema multiusuario es la impresora. Normalmente, en sistemas Unix/Linux solamente se utilizan impresoras conectadas al ordenador central, que son gestionadas directamente por éste y utilizables por todos los usuarios.

Estos sistemas operativos constan de estos dos componentes: *núcleo* y *Shell*. Observa algo más sobre estos dos componentes:

- **Shell.** Es el equivalente al intérprete de comandos de DOS (**Command.com**) o el Explorador de Windows. Permite introducir comandos, ejecutar programas, etc. La diferencia respecto a DOS y Windows es que, además, el *Shell* es un lenguaje de programación que permite controlar como se están ejecutando las comandos.
- **Núcleo.** Es la parte del sistema que interactúa con el hardware. Aporta servicios para la gestión de memoria, control de acceso a los periféricos, control y gestión del sistema de archivos, manejo de interrupciones, gestión de procesos de entrada/salida, etcétera.

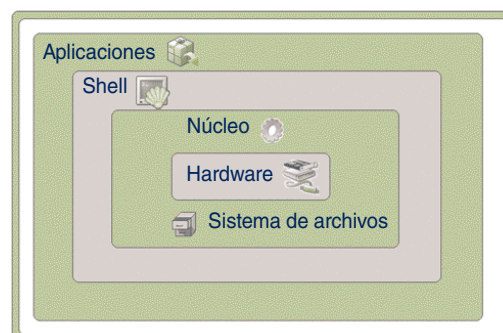


Fig. 13.1. Estructura del sistema operativo Unix/Linux.

## 13.2 Introducción al sistema multiusuario Unix/Linux

### A. Historia y versiones



Los inicios de Unix datan de los años 60, fruto de un proyecto de investigación de los laboratorios BELL. El sistema se llamaba **MULTICS**.

El proyecto estuvo parado durante algunos años, básicamente, por dos problemas: necesitaba mucha memoria y los programas no acababan de funcionar bien con él.

Ya en 1969 se escribe de nuevo un SO basado en MULTICS que supera los problemas de éste, para aplicarlo en máquinas **PDP-7**. Unix es muy parecido a este.

En 1973 se escribe en lenguaje «C», gana sobre todo en portabilidad y mejora su mantenimiento. Eso le da un gran auge.

A partir de aquí, el sistema sufre muchas modificaciones, hasta que en 1982, **AT&T**, lanza la primera versión comercial de Unix. A la vez, otras empresas sacaron «sus» versiones de Unix, lo cual hizo dudar a los usuarios, ya que no se sabía qué partes pertenecían realmente a Unix o si eran alguna de sus variantes.

Unix empezó a fraguarse con más rapidez con la aparición de los ordenadores **PDP-11** a mediados de los años setenta. Como este sistema operativo se ajustaba muy bien a las características y arquitectura de los ordenadores **DEC-PDP-11**, la venta de éstos se incrementó y facilitó el desarrollo del sistema operativo.

Simultáneamente, AT&T envió copias del sistema operativo Unix a muchas universidades del mundo, por lo que la mayoría de los informáticos realizaban sus estudios basándose en este sistema operativo. Fue en la Universidad de California de Berkeley, donde se desarrolló aún más este sistema operativo, mientras que AT&T seguía su propio desarrollo.

Debido a ello, la expansión del Unix se vio algo frenada, ya que no se podía asegurar en ningún caso la compatibilidad de programas en ambas versiones. Para ello se creó un estándar y multitud de librerías, de forma que la característica fundamental del Unix, la portabilidad, fuera máxima.

Con la aparición de los procesadores 8088 de IBM, Unix mejoró en prestaciones e incluso llegaron a hacerse versiones **ligeras** del Unix, como el sistema operativo **XENIX** para el IBM PC.

La evolución de este sistema operativo se puede consultar en Internet, si bien cabe indicar como característica general, que la primera versión de este sistema operativo nace en 1960 con la empresa MULTICS y, tras reformarse prácticamente cada año, llega al sistema de la actualidad en la forma de SYSTEM V RELEASE 4.



La historia de Linux empieza en Finlandia, en 1991, cuando en **Linus B. Torvalds**, estudiante de la Universidad de Helsinki, se le ocurrió comprarse un PC con procesador 386. Después de observar que el MS/DOS no aprovechaba los recursos de la máquina, decidió usar otro sistema operativo de entonces: **Minix** (Minix era un pequeño sistema Unix).

Sin embargo, debido a las limitaciones del Minix, Linus decidió reescribir algunas partes del sistema, añadiéndole mayor funcionalidad. Posteriormente, Linus difundió el código fuente por Internet, de manera gratuita y con el nombre de Linux. La primera difusión de Linux tuvo lugar el mes de agosto de 1991. Se trataba de la versión 0.01, y por el momento, funcionaba bajo Minix.

Esta primera versión era lo que se podría denominar un embrión ni siquiera hubo anuncio oficial. La primera versión «oficial», la 0.02, se hizo pública el 5 de octubre de 1991. En esta nueva versión ya se permitía el uso de algunos programas **GNU** (GNU es un acrónimo recursivo que significa «GNU No es Unix»).

En estas primeras versiones, Linux era bastante limitado, pero el hecho de que se difundiera la fuente por Internet, y totalmente gratis, hizo que cada vez más personas empezaran a colaborar con el proyecto, hasta llegar a los cientos de colaboradores que hay trabajando en la actualidad en los cientos de proyectos GNU.

La primera versión estable de Linux fue la 1.0 y apareció en marzo de 1994. El número de versión asociado al núcleo tiene un sentido muy particular, ya que está ligado a su desarrollo; la evolución de Linux se efectúa en dos fases:

- **Fase de desarrollo:** es el momento en que se añade funcionalidad al núcleo, optimizaciones y demás. En definitiva, es la fase en la que se desarrolla más el núcleo y se caracteriza por su nombre de versión impar: 1.1, 1.3, etcétera.

# 13. Sistema operativo multiusuario Unix/Linux...

## 13.2 Introducción al sistema multiusuario Unix/Linux

Distribución	MANDRAKE	REDHAT	DEBIAN	GENTOO	SUSE	SLACKWARE	LYCORIS	BEEHIVE	TURBO LINUX	CALDERA
										
Procedencia	Francia	EE. UU.	EE. UU.	EE. UU.	Alemania	EE. UU.	EE. UU.	EE. UU.	Japón	EE. UU.
N.º de CD-ROM	3	7	7	1	3	4	3	1	7	6
Versión del núcleo	2.4.18	2.4.18	2.2.20	2.4.19	2.4.18	2.4.18	2.4.18	2.4.18	2.4.18	2.4.13
Instalación	Gráfica	Gráfica	Texto	Texto	Gráfica	Texto	Gráfica	Texto	Gráfica	Gráfica
Gestor por defecto	KDE	Gnome	—	—	KDE	KDE	KDE	KDE	KDE	KDE
Tipo de paquetes	rpm	rpm	deb	scr	rpm	tar.gz	rpm	tar.gz	rpm	rpm

Tabla 13.1. 10 distribuciones de Linux.

- **Fase de estabilización:** se trata de coger el núcleo desarrollado en la fase anterior, y hacer que sea lo más estable posible. Aquí las modificaciones son mínimas; se trata más de retoques y pequeños ajustes. Los núcleos estables tienen número de versión par: 1.0, 1.2, 2.0, 2.4, etcétera.

Actualmente, Linux es un sistema Unix completo, aunque inicialmente se diseñó como un clónico de éste, distribuido libremente para funcionar en máquinas PC con procesadores 386, 486, etc. En la actualidad funciona sobre otras muchas plataformas como los procesadores Alpha, Sparc, Amiga, Atari, las máquinas tipo MIPS y sobre los PowerPC.

Hay que resaltar también que Linux respeta las especificaciones **POSIX**, pero posee también ciertas extensiones de las versiones System V y BSD de Unix. Esto simplifica notablemente la adaptación de programas desarrollados inicialmente para otros sistemas Unix.

El termino POSIX significa *Portable Operating System Interface*. Son unas normas definidas por el IEEE y estandarizadas por el ANSI y el ISO. POSIX permite tener un código fuente transportable.

Observa la Tabla 13.1, con las versiones más utilizadas de Linux en la actualidad.

Esta Unidad y las siguientes se han centrado en la distribución de SUSE por ser una de las más estandarizadas y similares a Unix que existen en el mercado.

Además, es la distribución utilizada actualmente por el Ministerio de Educación para los centros educativos.

### B. Características generales

Como se ha comentado anteriormente, Unix/Linux es un sistema operativo **multiusuario**.

Por otro lado, es un **multitarea**, es decir, el sistema permite que los usuarios estén ejecutando varias aplicaciones simultáneamente utilizando la técnica de tiempo compartido. Para ello se aplican los diferentes algoritmos de planificación como los que viste en la Unidad 2. A continuación se detalla el utilizado habitualmente por Unix/Linux y otros sistemas multiusuario, junto con algunas de las características más importantes de estos sistemas operativos:

- **Algoritmo por prioridades o multinivel.** Es uno de los más complejos y eficaces. Asigna los tiempos de ejecución de la UCP según una lista de prioridades. En cada una de estas listas, el sistema operativo incluirá aquellos procesos a los que se les haya asignado esa prioridad. El tiempo de ejecución del procesador se irá destinando, en primer lugar, de forma secuencial a los procesos de mayor nivel. Terminados éstos, se ejecutarán los procesos del nivel inferior, y así sucesivamente hasta los procesos del nivel más bajo.
- **Memoria virtual.** Esta técnica permite a los usuarios del sistema ejecutar programas, de tal forma que dé la sensación de que toda la memoria RAM es para ellos. Concretamente en Unix/Linux se utiliza la paginación de la memoria. Esta técnica, que ya viste en la Unidad 2, es la que utilizan la mayoría de los sistemas operativos multiusuario. Dividen la memoria en páginas al igual que los programas y de esta forma se realiza el intercambio entre disco y RAM para ejecutar los mismos.
- **Sistema de archivos jerárquico.** Utiliza, de forma similar al DOS, un sistema de archivos en forma de árbol invertido. La diferencia esencial frente a DOS es que el Unix/Linux no gestiona dispositivos (como una disquetera) de forma directa. El Unix/Linux gestiona los dispositivos como si fueran directorios, de tal forma que cuando estemos accediendo al directorio asociado a una disquetera, en realidad lo que estarás haciendo es acceder a la información contenida en el disquete.

## 13. Sistema operativo multiusuario Unix/Linux...

### 13.3 El sistema operativo Unix/Linux a fondo

- **Comunicación con otros ordenadores.** Un sistema Unix/Linux permite no solamente trabajar con él en una máquina, sino también conectar varios ordenadores centrales Unix/Linux entre sí de tal forma que cada usuario tenga acceso a la información contenida en todos ellos. La conexión se realiza a través del conjunto de protocolos y servicios que ofrece TCP/IP. Gracias a él puedes ejecutar programas en máquinas Unix/Linux que estén a varios kilómetros de distancia entre sí; enviar correo electrónico de unos equipos a otros; realizar conversación directa entre dos usuarios, etcétera.
- **Sistemas de seguridad.** Es una de las características más importantes, ya que la información a la que un usuario puede tener acceso puede limitarse de

forma sencilla. De este modo, el administrador del sistema operativo, a través de palabras clave (para archivos empaquetados o comprimidos) o mediante la asignación de derechos a los usuarios, hace que la información contenida en un servidor Unix/Linux esté totalmente protegida de piratas o usuarios no deseados.

- **Interfaz texto/gráfica.** Lo normal es que la interfaz utilizada por Unix/Linux sea de tipo texto. Pero poco a poco se han ido incorporando mejoras que permiten gestionar el sistema operativo en modo gráfico. Fundamentalmente, la aparición de las interfaces **X WINDOWS** ha permitido agilizar y mejorar procesos, especialmente para el administrador.

## 13.3 El sistema operativo Unix/Linux a fondo

Para comprender el funcionamiento del sistema Unix/Linux, es necesario entender su estructura. Este sistema operativo está formado por varios componentes principales. Entre ellos, el **núcleo**, el **Shell**, el **sistema de archivos** y los **comandos**.

### A. El núcleo y el shell

El núcleo es la parte del sistema operativo que sirve para interactuar con el hardware. Proporciona una serie de servicios que pueden ser utilizados por los programas, sin que éstos tengan que preocuparse de cómo se gestiona el hardware.

En general, el **núcleo** es el encargado de gestionar la memoria, mantener el sistema de archivos, del manejo de las interrupciones, manejo de errores, realización de los servicios de entrada/salida, asignación de los recursos de la UCP, gestión de periféricos de entrada/salida, etcétera.

Cada programa se relaciona con la máquina a través del núcleo. Un programa realizará al núcleo las denominadas **llamadas al sistema**. Con estas el programa indicará, por ejemplo, que le abra un archivo, que escriba en otro, que utilice la impresora, que cambie la prioridad de ejecución de otro proceso, etcétera.

El núcleo del sistema operativo Unix/Linux, que recibe el nombre de **KERNEL**, actúa directamente con los elementos físicos del ordenador, y se carga en memoria al arrancar la máquina. Permanece en ella hasta que ésta se apaga. Recordemos que en DOS, el núcleo estaba formado por dos programas MSDOS.SYS y IO.SYS.

El **Shell** es el intérprete de mandatos o de comandos con el que cuenta este sistema operativo. En DOS es el **Command.com** el que se encarga de realizar esta función.

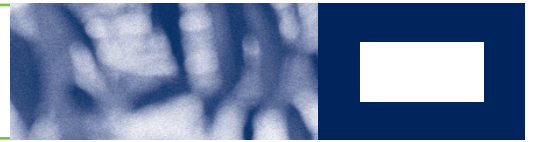
El **Shell** actúa como interfaz de comunicación entre el usuario y el ordenador, y cuando un usuario se conecta con el servidor Unix/Linux, automáticamente se arranca un **Shell** para que pueda trabajar. Cada usuario conectado al servidor tendrá un **Shell** para su uso.

Al contrario que en DOS, en el que el intérprete de comandos es único, en Unix/Linux existen varios. Éstos son los siguientes:

- **Shell Bourne (sh).** Creado por S. Bourne, es el más utilizado en la actualidad. El prompt del sistema queda representado por el símbolo «\$». Este **shell** es el estándar de AT&T y el que se monta en casi todos los sistemas Unix/Linux.
- **C-Shell (csh).** Procedente del sistema BSD, proporciona características tales como control de trabajos, historia de comandos (como el doskey en DOS), capacidades de edición, etc. Ofrece importantes características para los programadores que trabajan en lenguaje C. Su prompt de sistema queda representado con el símbolo «%».
- **Shell job (jsh).** Incorpora algunas características de control al **shell** estándar del sistema.
- **Shell Korn (ksh).** Escrito por David Korn, amplía el **shell** del sistema añadiendo historia de comandos, edición de la línea de ordenes y características ampliadas de programación.

## 13. Sistema operativo multiusuario Unix/Linux...

### 13.3 El sistema operativo Unix/Linux a fondo



- **Bourne Again shell (Bash).** Fue creado para usarlo en el proyecto GNU. BASH, por lo tanto, es un *shell* o intérprete de comandos GNU; éste es compatible con el *shell* **sh**. Además, incorpora algunas características útiles de **ksh** y **csk**, y otras propias, como la edición de línea de comandos, tamaño ilimitado del histórico de comandos, control de trabajos y procesos, funciones y alias, cálculos aritméticos con números enteros, etcétera.

### B. Funciones de *shell*

Como ya has visto anteriormente, la interfaz de usuario que sirve de comunicación entre éste y el sistema recibe el nombre de *shell*.

Cuando un usuario emite una orden, se está relacionando con el *shell*, parte del sistema Unix/Linux a través de la cual se controlan los recursos del sistema operativo, proporcionando características que hacen que el sistema sea potente y flexible.

El *shell* es tanto un intérprete de comandos como un lenguaje de programación interactivo. Sobre él se pueden ejecutar comandos con los que se pueden crear programas denominados *guiones*.

Cuando el usuario se conecta al sistema Unix/Linux, se inicia automáticamente un programa de *shell*. Éste es el denominado *shell de presentación*. Este *shell* se carga de forma automática cuando se accede al fichero **/etc/passwd**. Este archivo contiene la información que el sistema necesita conocer de cada usuario. Es dentro de este archivo o fichero, el situado en último campo de cada línea, donde se encuentra el nombre del *shell* que quieres ejecutar, que normalmente es el *shell* estándar del sistema.

En cuando se inicia el *shell* de presentación, se busca el fichero **.profile** dentro del directorio activo. Este fichero contiene comandos que sirven para personalizar su entorno de trabajo. Asimismo puede compararse al **AUTOEXEC.BAT** del DOS.

Realizada esta operación automáticamente, se muestra en pantalla, normalmente, el símbolo «\$», para que el usuario pueda introducir sus comandos.

En general, todas las órdenes de Unix/Linux son programas que están almacenados en el sistema de archivos. Se escriben de forma similar a como se hace en DOS, ya que su sintaxis es la siguiente:

**\$ mandato [-modificadores] [argumentos]**

### C. Interfaces de usuario

Éstas se definen como la parte del Sistema Unix/Linux que determina cómo interactúa el usuario con él, es decir, de qué forma el usuario introduce comandos o cualquier otra información y cómo el sistema visualiza los mensajes después de procesar tal información.

La interfaz primaria o básica del Unix/Linux es de tipo texto y hasta no hace mucho ha sido la única para el sistema. Actualmente hay interfaces gráficas como el **X Windows**, **Open Look**, **GNOME** o **KDE**.

La interfaz de tipo texto es la que se muestra al cargar el **shell** por defecto o el deseado. Las básicas son las mismas que las explicadas en el punto 13.4.A de esta unidad, aunque hay más.

En cuanto a la interfaz gráfica **X-Windows**, la característica fundamental es que incorpora un *modelo cliente-servidor* para el modo en que las aplicaciones interactúan con los dispositivos terminales. Incorpora también un *protocolo de red* y varias *herramientas software* que pueden ser utilizadas para crear aplicaciones basadas en X Windows.

Un concepto fundamental es la separación de las aplicaciones con respecto al software que maneja la entrada y salida por *Terminal*. Todas las operaciones realizadas mediante la entrada o salida estándar (teclado y monitor) son manejadas por un programa que se dedica exclusivamente a ello (**servidor**). Las aplicaciones (**clientes**) envían al servidor información a visualizar, y el servidor envía a las aplicaciones información referente a la entrada de usuario. Para gestionar este modelo, se utiliza el protocolo de **red X**. Este protocolo es un lenguaje estándar utilizado por las aplicaciones clientes para enviar instrucciones a los servidores X, y por los servidores para enviar la información transformada a los clientes (por ejemplo, el movimiento del ratón).

La interfaz **OPEN LOOK** ha sido diseñada por AT&T y Sun Microsystems como interfaz gráfica estándar para el sistema Unix/Linux.

Esta interfaz permite ejecutar y visualizar varias aplicaciones simultáneamente en ventanas separadas sobre una misma pantalla. En general, todas las operaciones de gestión de archivos se realizan de forma gráfica gracias a la interfaz OPEN LOOK.

La interfaz **KDE 3.2** para Linux SUSE, ahora con un potente gestor de información personal (PIM), incorpora, además de la función de inicio rápido y la reproducción automática de los medios introducidos (CD, DVD), KDE 3.2, un gran número de nuevas prestaciones y programas. Observa algunos de ellos:



## 13. Sistema operativo multiusuario Unix/Linux...

### 13.4 Procedimiento de conexión y desconexión

- **Kontakt.** Este programa equiparable a MS Outlook integra herramientas de correo electrónico, organización, libreta de direcciones y notas.
- **KWallet.** *Password safe* utilizado por numerosas aplicaciones para administrar de forma segura las distintas contraseñas del usuario.
- **Kopete.** Programa de mensajería instantánea.
- **KDevelop.** Dispone de soporte para más de 15 lenguajes de programación, diseño configurable, libre elección de editor, vista separada de clases para C++, código completado automáticamente, depuradores integrados y comprobación de sintaxis mientras se introduce el código.
- **Konqueror.** La nueva versión del navegador Web.
- **Quanta Plus.** Editor HTML con un nuevo componente de editor WYSIWYG.
- **Barra lateral universal.** Barra lateral de navegación para la representación jerárquica de árboles de directorios que puede ser mostrada en el escritorio independientemente del administrador de archivos Konqueror.

La interfaz GNOME 2.4 ha mejorado la usabilidad y se han incluido ayudas de accesibilidad para discapacitados, como, por ejemplo, un lector de pantalla con salida de voz o a una línea braille.

## 13.4 Procedimiento de conexión y desconexión

### A. Conexión de usuarios remotos a un servidor Unix/Linux

En este epígrafe vas a ver cómo puedes conectarte desde cualquier ordenador a otro ordenador con Unix/Linux. En este caso, te podrás conectar como un usuario cualquiera o como el administrador del sistema, es decir, como **root**.

La diferencia es el tipo de operaciones que podrás realizar en el servidor desde el terminal desde el que te conectas.

En primer lugar, es evidente que el ordenador central o servidor tiene que estar conectado. Esto, aunque parece obvio, es importante, especialmente en aquellos casos en los que el servidor no se encuentra en el mismo espacio físico que el usuario. Si es así y el usuario no llega a conectar, no tendrá demasiado claro si el fallo se debe a un problema de su equipo, de la línea de transmisión o sencillamente del servidor.

El inicio del ordenador que tiene instalado el sistema operativo Unix/Linux en el servidor suele ser automático. Arrancado este ordenador, podrás iniciar o no sesión en él físicamente. En este caso, la iniciarás, normalmente, en entorno gráfico.

El primer paso para conectar desde otro ordenador con el servidor Unix/Linux es lanzar desde nuestro ordenador el comando de conexión. Esta orden de conexión solamente pretende hacer que el ordenador del usuario y el ordenador central se entiendan.

El protocolo de comunicación, en la actualidad, suele ejecutarse bajo el entorno Windows o Linux, y se puede

lanzar con el software denominado **TELNET**. *Telnet* es el software que permite la conexión entre terminales mediante el protocolo de comunicaciones TCP/IP. Este software, que en realidad es un servicio más de los que incorpora el propio TCP/IP, es un protocolo de esta familia. Sirve para conectarse a cualquier equipo identificado con una dirección IP determinada.

Observa a continuación el proceso de conexión, teniendo en cuenta que el ordenador desde el que te vas a conectar es un ordenador en emulación y que el sistema operativo instalado es Windows XP.

En primer lugar, ejecutarás *Telnet* desde la opción *Ejecutar* del botón de *Inicio*. Dependiendo de la versión del sistema operativo desde el que te conectes, aparecerá una u otra pantalla. En versiones de Windows 98, aparece una ventana en la que podrás introducir los parámetros de la conexión en entorno gráfico. En versiones Windows 2000/XP, automáticamente aparecerá una ventana de comando, tipo DOS, en la que tendrás que introducir, siempre, los datos de la conexión a realizar.

Si conectamos desde un Terminal con Windows 98, el procedimiento a seguir es el siguiente:

- Ejecutas *Telnet* y en la primera pantalla (Figura 13.2), en el menú *Conectar*, pulsarás la opción *Sistema remoto*, y la primera vez aparecerá lo siguiente (Figura 13.3).
- En la casilla *Nombre del Host*, especificarás la dirección IP que tenga el servidor Unix/Linux; por ejemplo, 192.168.0.2.



Fig. 13.2. Conexión con Telnet.

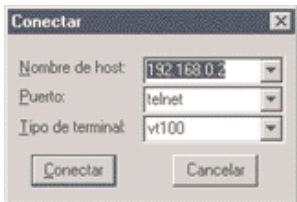


Fig. 13.3. Conectar con un equipo remoto.

## 13. Sistema operativo multiusuario Unix/Linux...

### 13.4 Procedimiento de conexión y desconexión

Las casillas de *Puerto* y *Tipo de terminal* las dejarás como están, ya que, por defecto, Windows interpreta que *Telnet* se utilizará para conectar con un ordenador de estas características. Concretamente, el puerto, es mediante el que el ordenador puede entenderse con el servidor Unix/Linux, y el tipo de terminal implica de qué forma se va hacer la emulación en tu terminal, es decir, cómo tu ordenador visualizará la ventana para trabajar en Unix/Linux.

Cuando realizas la primera conexión de forma efectiva, en sucesivas conexiones no será necesario realizar esta secuencia de conexión, ya que en la siguiente ocasión, cuando pulses *Conectar*, saldrá una lista de direcciones TCP/IP válidas con las que alguna vez has conectado con un ordenador remoto. De esta forma, solamente será necesario hacer clic sobre la dirección IP que quieras.

Si realizas la conexión desde versiones 2000/XP de Windows, lo harás pulsando *Inicio*, *Ejecutar*, y dentro introducirás el comando siguiente: **Telnet 192.168.0.2**.

Si en la línea de ejecución tecleas solamente *Telnet*, a continuación tendrás que introducir la dirección IP tras el símbolo del sistema, que muestra lo siguiente: **Microsoft Telnet> 192.168.0.2**.

También puedes utilizar la herramienta **TermLite**. Esta herramienta la incorpora el fabricante de Unix SCO (Santa Cruz Operating) en los discos que suministra con el producto.

Ejecutado *Telnet* y ejecutado el comando de conexión, el usuario recibirá la siguiente información suministrada por el servidor: **login:** o **Linux login:**, dependiendo de que el sistema con el que estés conectando sea Unix o Linux respectivamente.

Introducirás tu identificación de usuario y a continuación se te pedirá que introduzcas la contraseña o **password**, obligatoria en la mayoría de los casos en todos los sistemas Unix/Linux.

Es importante tener en cuenta que Unix/Linux, al ser un sistema **case sensitive**, diferencia las letras mayúsculas de las minúsculas, es decir, no es lo mismo USER01 que user01.

La contraseña solamente tiene que ser conocida por el usuario. Al teclearla no aparecerá ningún carácter en pantalla, por seguridad, pero después de su introducción se tendrá que pulsar de nuevo **Return**, al igual que cuando introdujiste el nombre de usuario.

Si al introducir el ID del usuario o la *password* te has equivocado, el sistema no te dejará entrar, enviará un mensaje y volverá de nuevo a la situación de partida.

Después de estos mensajes, el sistema carga un *shell* para ese usuario y aparece el símbolo del sistema, que variará dependiendo del tipo de *shell* asignado. Normalmente, el símbolo del sistema es el símbolo «\$» para usuarios y el símbolo «#» para el administrador, aunque esto dependerá siempre del *shell* que se cargue al iniciar sesión.

A partir de este momento podrás trabajar sobre el sistema Unix/Linux con los privilegios que tengas concedidos en el mismo.

### B. Conexión de usuarios sobre el propio servidor Unix/Linux



Encendido el servidor Unix, aparecerá una pantalla en modo gráfico, en la que introducirás, de forma similar, el nombre de usuario y contraseña para entrar al sistema. Pulsarás en el botón *Login*. Tras pulsar sucesivas veces el botón *OK*, llegarás al escritorio principal de Unix. Obsérvalo en la Figura 13.4.

Para cerrar la sesión de trabajo, si estamos conectados en modo texto, el usuario podrá hacerlo de dos formas distintas.

- Teclear **exit** y luego pulsar **Return**.
- Teclear **Ctrl+d**.

Después de esto, el sistema mostrará de nuevo: **login:** y el usuario podrá conectar de nuevo o no.

El cierre de sesión en modo gráfico es seleccionando en el menú *File*, opción *Exit*; posteriormente se confirmará el abandono de la sesión de trabajo. Observa la Figura 13.5.

Para cerrar definitivamente el equipo, ejecutarás una sesión en modo comando y teclearás: **#shutdown -g0**.



En la pantalla inicial de Unix. La diferencia se muestra en la pantalla de conexión, ya que, a diferencia de Unix, en Linux podrás seleccionar algunas opciones antes de entrar al sistema. Podrás seleccionar algún usuario de los que aparecen a la izquierda de la pantalla de **login**, así como el tipo de escritorio, aunque te recomiendo utilizar, por defecto, KDE. Por último, en la opción menú podrás reiniciar el servidor o apagarlo.

Introducidos los datos de conexión de forma adecuada, se mostrará el escritorio de Linux, tal y como puedes ver en la Figura 13.6.

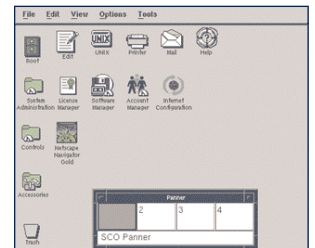


Fig. 13.4. Escritorio de Unix SCO.

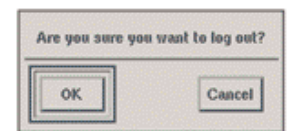


Fig. 13.5. Cerrar sesión de usuario en Unix SCO.

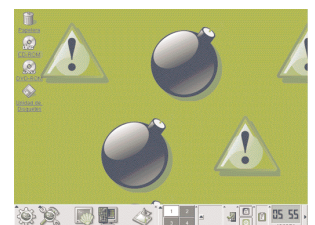





Fig. 13.6. Escritorio del administrador SUSE.



## 13. Sistema operativo multiusuario Unix/Linux...

### 13.4 Procedimiento de conexión y desconexión

Si inicias sesión como un usuario estándar, el escritorio tomará un aspecto diferente, aunque básicamente será el mismo.

Si quieres terminar la sesión de usuario, o cambiar de usuario, sin cerrar el sistema, harás clic en el icono correspondiente , o en el icono *Iniciar aplicación* . Terminar . En cualquier caso, aparecerá la pantalla de la Figura 13.7, en la que podrás indicar el tipo de operación que quieres realizar.

En la figura anterior, puedes observar que las opciones son las de cambiar de usuario, reiniciar el equipo o apagarlo. De este forma, no necesitas, al contrario que en Unix SCO, realizar acciones diferentes para cambiar de usuario o para cerrar el sistema.

#### C. Primera conexión al sistema Unix/Linux

Si el usuario que se conecta lo hace por primera vez, y si el administrador no ha decidido asignarle una palabra clave, no habrá contraseña, o la contraseña será nula.

Si se da este caso, el usuario podrá asignarse él mismo una clave de acceso utilizando el comando **passwd**. En este caso, una vez que al usuario le aparezca el prompt del sistema «\$», y después de introducir el comando **passwd**, aparecerán los siguientes mensajes:

**\$ passwd**

**passwd: changin password for user01**

**Old password:**

**New password:**

**Re-enter new password:**

**Password Changed**

Como se puede apreciar, lo primero que se te solicita es la clave antigua. Es evidente que, si esto no fuera así, cualquier usuario podría cambiar las claves de los demás. Con ello el sistema consigue que sólo el usuario pueda cambiar su propia clave de acceso.

Después de verificar la clave, teclearás la nueva clave dos veces. De esta forma, se comparan las dos claves para verificar que la introducida originalmente es la deseada.

Quando cambias la palabra clave, tienes que seguir una serie de normas. Estas normas suelen ser definidas por el administrador del sistema, y son las siguientes:

- La palabra clave ha de ser diferente.
- Debe tener al menos seis caracteres.
- Al menos dos caracteres tienen que ser alfabéticos.
- Debe tener al menos un carácter numérico o especial.
- Tiene que ser distinta del nombre de cuenta.
- No se puede utilizar como palabra clave los mismos caracteres asignados como nombre de usuario cambiados de orden.
- No se puede cambiar la contraseña de mayúsculas a minúsculas.

Quando veas la gestión de usuarios en modo gráfico, verás cómo se cambian las claves de acceso.

#### D. Primeros conceptos Unix/Linux

En el punto anterior hemos hablado del nombre de la cuenta del usuario. Este nombre de cuenta es un número que el administrador del sistema asigna a cada usuario que da de alta en el sistema.

Quando el administrador da de alta un nuevo usuario, además de asignarle el nombre, le asigna un **número de identificación de usuario (uid)**. Además de esto, a cada usuario se le incluye en un **grupo de trabajo (gid)** que también es un número.

Un grupo de trabajo es un entorno creado por el administrador para que determinados usuarios puedan hacer uso del sistema en las mismas condiciones. Es en este entorno en el que se les asigna a los diferentes usuarios los derechos de utilización del sistema, es decir, a qué directorios tienen acceso y a cuáles no, qué programas pueden ejecutar y cuáles no, etcétera.

La información correspondiente a cada usuario de su **uid** y su **gid** puede ser conocida mediante el tecleo del comando **id**.

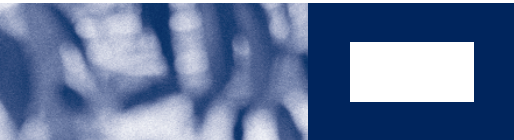
**\$ id uid = 200 (users) gid = 100 (users)  
groups(users)**

Ya sabes cuál es la forma para entrar y salir del sistema, y también qué aspecto toma el **indicador del sistema** o intérprete de comandos.

Quando se inicia el **shell** de presentación de cada usuario, se busca un archivo denominado **.profile** en el di-



Fig. 13.7. Terminar sesión de trabajo en Linux.



rectorio asignado al usuario (HOME), que puede verse en el fichero `/etc/passwd`.

Este fichero, que puede asemejarse al **Autoexec.bat** en DOS, contiene comandos e instrucciones que el *shell* ejecutará y que serán necesarias para ejecutar determinados programas y, en general, para particularizar el entorno de trabajo de cada usuario.

La **presentación en pantalla** de los diferentes *shell* ha sido abordada anteriormente, aunque conviene recordar que lo normal es que al superusuario le aparezca el símbolo `#` como indicador de sistema y que a los usuarios les aparezca un `$`. También has visto cuál sería la interfaz en modo gráfico.

### E. Teclas y caracteres especiales

En general, todos los teclados diseñados para trabajar en Unix/Linux son compatibles con el código **ASCII**.

Estos teclados son parecidos o iguales a los de un ordenador personal, pero algunos de ellos (especialmente los terminales que no son autónomos) pueden incorporar teclas que realizan funciones particulares y específicas en el sistema.

En general, aunque la ubicación de las teclas es susceptible, los teclados tienen la siguiente estructura:

- **Letras mayúsculas y minúsculas.** Teclado QWERTY.
- **Dígitos numéricos.** De 0 al 9.
- **Símbolos especiales.** Son los siguientes: `@ - # & ) _ + = ` [ ] \ : ; " ' < > ? / | . ,`
- **Teclas especiales.** Return, Delete, Backspace y Tab.
- **Barra espaciadora.**
- **Caracteres de control.** Combinación de CTRL más una letra.
- **Teclas de función.** Usadas en tareas de programación.


 Comando	Sintaxis	Función
<b>clear</b>	clear	Borra la pantalla en modo comando.
<b>date</b>	date [-u —utc —universal] [MMDDhhmm[[CC]YY][.ss]]	Presenta en pantalla el día y la hora que tiene el equipo en ese momento. Muestra, de izquierda a derecha, el día de la semana, mes, día del mes, hora, meridiano y año.
<b>cal</b>	cal [-smjy13] [[month] year]	Calendario perpetuo que incorpora el sistema. Permite ver calendarios completos de otros años o el calendario de un mes determinado.
<b>who</b>	who [options] [file   ARG1 ARG2 ]	Muestra por pantalla una línea por cada usuario que en ese momento está conectado al sistema. Muestra de izquierda a derecha el nombre de presentación del usuario, el número de terminal, y fecha y hora de presentación. Una variedad de este comando es <b>who am i</b> , que muestra solamente información referente a el usuario conectado.
<b>finger</b>	finger [-h[Mmops] [user] finger [-l] [user@host]	Presenta información completa de los usuarios conectados a la red. <b>\$ finger user01</b>
<b>write</b>	write user [ttyname]	Permite la comunicación entre usuarios, ya que permite escribir un mensaje directamente en el terminal de otro usuario, siempre y cuando se conozca cuál es el terminal al que está conectado el usuario. Copia en la pantalla de otro usuario lo que se escribe en nuestro terminal. Se finaliza pulsando <b>Ctrl + d</b> . Supón que eres <i>user01</i> y escribes: <b>\$ write user02</b> . Esto significa que quieres dialogar con <i>user02</i> .
<b>talk</b>	talk address [terminal]	Programa mejorado para comunicación de terminal a terminal, de forma que las pantallas de los interlocutores se dividen en dos. La conversación finaliza pulsando <b>Ctrl + d</b> . Si eres <i>user01</i> y quieres dialogar con <i>user02</i> , deberás escribir: <b>\$ talk user02</b> . Este comando significa que quieres dialogar con <i>user02</i> .
<b>mesg</b>	mesg [ n ] [ y ]	Permite aceptar o rechazar mensajes enviados a través de <b>write</b> y <b>talk</b> . <b>\$ mesg n</b> . Evita que podamos recibir mensajes. <b>\$ mesg y</b> . Restaura el permiso para recibir mensajes. <b>\$ mesg</b> . Muestra el estado de este permiso en nuestro terminal.
<b>wall</b>	wall [file]	Sirve para enviar un fichero, o los caracteres escritos, a todos los usuarios conectados. El terminal que recibe el mensaje (por ejemplo, <i>user02</i> ) va precedido del mensaje: <b>Broadcast message from user02</b> .

Tabla 13.2.

## 13.5 Comandos básicos de Unix/Linux

El sistema Unix/Linux tiene a disposición de todos sus usuarios una gran cantidad de programas. Estos programas se ejecutan mediante comandos.

Asimismo tiene una sintaxis de comandos bastante estandarizada, de tal forma que puedan aplicarse a casi todos ellos por igual.

Comando	Opciones	Argumentos
---------	----------	------------

Tabla 13.3.

Generalmente, las comandos en Unix/Linux:

- Se utilizan por sí solas.
- Se utilizan acompañados por argumentos.
- Algunos permiten opciones.

### A. Comandos básicos

Al igual que en DOS, hay comandos que el usuario puede manejar sin provocar problemas en el funcionamiento del sistema, como los que muestra la Tabla 13.2.

### B. Formato de los comandos de Unix/Linux

El formato de las comandos Unix/Linux es bastante parecido al de los comandos DOS. Cada orden se separa de las siguientes, ya que en Unix/Linux se puede introducir más de una orden en la misma línea, mediante un punto y coma (;).

Al igual que en DOS, lo primero que se introduce es el comando, seguido de los argumentos o parámetros del mismo. Cuando introduces parámetros o un argumento (caracteres como tal), se deja un espacio en blanco.

Observa la forma de introducir comandos en Unix/Linux.

- **Comando.** Indica al intérprete de comandos o *shell* la acción a realizar.
- **Opciones.** Es el modificador del comando. Las opciones se escriben a continuación del carácter -. Recuerda que en DOS se introducían tras el carácter /.
- **Argumentos.** Caracteres que se utilizan como entrada del comando. El argumento puede ser un archivo o un directorio.

Como puedes ver, la forma de introducir comandos en Unix/Linux es muy parecida a la forma de hacerlo en DOS. La diferencia fundamental es que aquí sí son importante las mayúsculas y las minúsculas. Normalmente, en Unix/Linux todas las comandos, archivos y directorios se escriben en minúsculas, y sin caracteres especiales. Por lo tanto, no será lo mismo **DATE** que **date**.

### C. Otros comandos básicos

(Véase la Tabla 13.4).


 Comando	Sintaxis	Función
<b>dpasswd</b>	<pre>passwd [-f -g -s -k[-q]] [name] passwd [-D binddn][-n min] [-x max][-w warn][-i inact] user  passwd [-D binddn] {-l -u -d -S[-a] -e -h} name</pre>	<p>Este comando se escribe sin parámetros o modificadores. Permite al usuario cambiar o asignar una contraseña, siempre que el administrador lo permita. Para el cambio de contraseña se tendrá en cuenta lo siguiente:</p> <ul style="list-style-type: none"><li>• La contraseña no es obligatoria.</li><li>• La contraseña puede ser modificada siempre que se desee.</li><li>• Ha de tener al menos seis caracteres (especificación C2).</li><li>• Son identificativos los primeros ocho caracteres.</li><li>• Debe tener al menos dos caracteres alfabéticos y uno numérico. En Linux no es igual, ya que la clave no tiene que cumplir determinadas normas.</li><li>• La contraseña puede caducar.</li><li>• El administrador no puede ver las contraseñas.</li><li>• Las contraseñas quedan registradas en el fichero <b>/etc/shadow</b>.</li></ul>
<b>uname</b>	<pre>uname [-asnrvm]</pre>	<p>Se utiliza para obtener el nombre del sistema en el que se está trabajando. Tiene la siguiente sintaxis: <b>uname -X</b> en Unix o <b>uname -a</b> en Linux.</p> <p>Informa del tipo de sistema operativo, versión, tipo de ordenador desde el que se conecta el usuario, etcétera.</p>
<b>logname</b>	<pre>logname</pre>	<p>Permite mostrar el contenido de la variable <b>logname</b>, que contiene el nombre del usuario conectado al ordenador, es decir, el nombre de usuario con el que has conectado con el sistema Unix/Linux.</p> <p>No tiene formato específico basta con escribir el comando o pulsar <b>Return</b> para ver qué usuario está identificado ante este ordenador.</p>
<b>id</b>	<pre>id [-aZGgru] [user]</pre>	<p>Muestra el número de identificación y el grupo al que pertenece el usuario.</p> <p>Cada vez que se da de alta un usuario en el sistema, se le asigna de forma automática o por decisión del administrador, un número que lo identifica sobre el resto de usuarios. Además, cada usuario se integra en un grupo de usuarios que también estará identificado con un nombre.</p>

Tabla 13.4.



## 13.6 Estructura del sistema Unix/Linux

La estructura en Unix/Linux es jerárquica en forma de árbol invertido. Se parte de un directorio principal **root**, representado por el carácter **"/"** (recuerda que en dos era **"\"**). Este directorio puede contener, al igual que en DOS, otros directorios o archivos que dependan de él.

La diferencia fundamental con DOS es que no existe un directorio raíz por cada unidad lógica de almacenamiento.

Los nombres de archivo y directorio en Unix/Linux siguen reglas parecidas a los nombres en DOS. Concretamente, un nombre de archivo puede ser casi cualquier secuencia de caracteres, y se considera que dos nombres de archivo son iguales si coinciden en los primeros catorce caracteres.

### A. Archivos y directorios

En cuanto a los caracteres para los nombres de archivo, se puede utilizar cualquiera, a excepción del carácter **/**, ya que tiene un significado especial. Es conveniente no utilizar caracteres especiales, acentos, etc. No se pueden incluir espacios en blanco ni es conveniente emplear caracteres como los siguientes en los nombres de archivo:

**! # & ( ) ` " ; | > < @ \$ { } \* ? \ Tab Spacebar  
Backspace + -**

Lo más importante es destacar que en Unix/Linux los nombres de archivo son diferentes en mayúscula y en minúscula. Se recomienda utilizar todos en minúscula.

Los directorios en Unix/Linux son archivos especiales cuya función es la de almacenar archivos u otros directorios. Tienen características especiales y, al igual que en DOS, cada directorio, a excepción del raíz, consta de los directorios **.** y **..**, que indican directorio actual y directorio padre, respectivamente.

Para entender mejor lo relacionado con los archivos, hablemos sobre el File System de Unix/Linux. El sistema de archivos de Unix/Linux es particular, respecto de los convencionales de Microsoft.

Su estructura consta de tres partes fundamentales: superbloque, tabla de inodos y bloques de datos. Concretamente, cada archivo o directorio tiene asociado un número en la tabla de inodos. Este número identifica la ubicación del archivo o directorio dentro de la zona de datos.

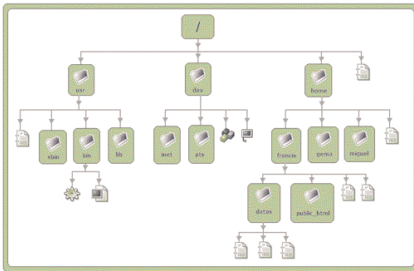
### B. Tipos de archivos en Unix/Linux

En Unix/Linux existen, básicamente, cuatro tipos de archivos. Son los siguientes:

- **Archivos ordinarios.** Contienen la información con la que trabaja cada usuario. Normalmente son archivos que contienen texto, programas escritos por el usuario en lenguaje C, etc. Suelen contener caracteres de tipo ASCII, y pueden ser modificados, creados, borrados, etcétera.
- **Enlaces físicos (Vínculos físicos).** No es específicamente una clase de archivo. Es un segundo nombre asignado a un archivo. Supón que dos usuarios necesitan compartir la información de un mismo archivo. Es evidente que si cada uno de ellos tiene una copia, el problema se soluciona, en cierta medida. Pero las modificaciones que realice un usuario, no podrán ser utilizadas por otro, ya que solamente se modificará la copia de uno de los usuarios.

Si en vez de tener una copia cada usuario, cada uno de ellos utiliza en archivo tipo **enlace** al archivo original, cada vez que uno modifique su archivo lo que estará modificando es en realidad el archivo común. El vínculo es un segundo nombre que hace referencia a un archivo, pero no es en sí el archivo. El vínculo sirve para localizar el fichero en su ubicación actual, pero no es el archivo real. De esta forma, con tener un solo archivo real, éste se podrá utilizar por todos los usuarios que lo necesiten, sin tener que duplicarlo o triplicarlo.

- **Enlaces simbólicos (Vínculos simbólicos).** Se utilizan para asignar más de un nombre a un archivo. No sirven para directorios. Un vínculo simbólico es un archivo que sólo contiene el nombre de otro archivo. Cuando el sistema operativo opera sobre un vínculo simbólico, éste se dirige al archivo al que apunta el vínculo simbólico. A diferencia de los enlaces físicos, que existen dentro de la estructura de archivos, los enlaces simbólicos solamente hacen referencia al nombre de otro archivo. Se utiliza el parámetro **-s**.
- **Directorios.** Son archivos especiales que contienen referencias a otros archivos. Cuentan con información sobre archivos ordinarios, subdirectorios, vínculos, vínculos simbólicos, etcétera.



**Fig. 13.8.** Estructura de directorios en Unix/Linux.

- **Archivos especiales.** Suelen representar dispositivos físicos como unidades de almacenamiento, impresoras, terminales, etc. Unix/Linux trata los archivos especiales como archivos ordinarios. De esta forma, un usuario puede abrir un archivo vinculado a una unidad de disquete, modificarlo, etc. Con ello consigue leer del disquete, escribir en el disquete, etc. Unix/Linux transforma las operaciones básicas y ordenes básicas de archivos ordinarios o regulares en comandos que interactúan con el hardware a través de los archivos especiales.

etc., archivos. Al igual que en DOS, se utilizarán rutas o caminos para realizar las operaciones con los archivos. Concretamente, en Unix no existe el **prompt** del sistema para que te indique la posición actual dentro de la estructura jerárquica. Linux sí lo hace. De esta forma, cuando un usuario quiera saber dónde se encuentra, ejecutará el comando **pwd**, y se mostrará por pantalla la ruta absoluta en la que está situado. Observa el siguiente ejemplo: **#pwd home/org/compras/material.**

## D. Metacaracteres

En muchas expresiones y comandos de Unix/Linux se utiliza un conjunto de caracteres con significado especial para búsquedas.

Recuerda los caracteres **comodín** en DOS. Servían para sustituir determinados caracteres cuando hacíamos referencia a nombres de archivo o directorio.

En Unix/Linux es lo mismo, pero se denominan **metacaracteres** en vez de caracteres comodín. Los metacaracteres son los que se muestran en la Tabla 13.5.

## E. Rutas en Unix/Linux

Al igual que en DOS, dentro de una rama diferente de la estructura jerárquica pueden existir directorios y archivos con el mismo nombre. Esta característica la determina la ruta o camino en el que está situado un directorio o archivo.

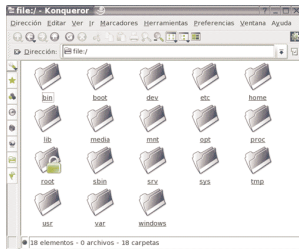
Evidentemente, no es lo mismo un archivo situado en la ruta **/home/org/ventas**, llamado *docs*, que otro con el mismo nombre situado en **/home/org/compras**.

Los nombres completos de los **caminos absolutos** van desde el directorio raíz hasta el propio archivo o directorio.

En Unix/Linux también existen los **caminos relativos**. Su utilización es igual que en DOS. Como has comentado anteriormente, también existen los directorios `.` y `..`

El directorio . hace referencia al directorio activo, que en Unix/Linux se denomina *directorio actual*. El directorio .. hace referencia al *directorio padre* que existe en todos los directorios, a excepción del directorio raíz.

Te recomiendo repasar las rutas en DOS, ya que en Unix/Linux se manejan de la misma forma. Esto tiene su explicación, y es que **el DOS adoptó esta forma de trabajar del Unix**. Es por lo que si se entiende el manejo de rutas y caminos en DOS, en Unix/Linux no tiene que suponer ningún problema.






**Fig. 13.9.** Estructura de directorios en forma gráfica en Linux.

### C. Estructura jerárquica de archivos y directorios

En DOS existe una limitación del número de archivos y directorios que se pueden crear dentro de un directorio. Esta limitación responde a la cantidad de caracteres con la que luego referenciarás el archivo y que no tiene que exceder de 63.

En Unix/Linux no hay límite (conocido). Observa en la Figura 13.8 un ejemplo de una estructura típica de directorios en el sistema Unix/Linux.

Para ver la estructura de archivos y directorios en entorno gráfico, tendremos que pulsar el icono  en Unix y  en Linux, introduciendo el símbolo del directorio raíz en el cuadro de *Dirección* o pulsando directamente en el escritorio el dispositivo que referencia el disco duro, unidad de CD-ROM, etc. Es evidente que previamente han tenido que ser montados, opción que verás más adelante; aunque por defecto, en entorno **root** estarán montadas normalmente en Linux las unidades de CD-ROM y la disquetera, y en entorno usuario en la versión 9.1 bastará con pulsar el icono *My Computer* , que hace parecer las funciones al de *Mi PC* en Windows.

Abriendo este objeto, aparecerá, de forma similar a una ventana de Windows, el contenido de toda la estructura de directorios que tenga tu sistema. Observa su aspecto en la Figura 13.9.

Al igual que en DOS, siempre que estés trabajando sobre la estructura jerárquica de archivos y directorios, el usuario estará situado en una unidad activa y en un directorio activo. En Unix/Linux el concepto de unidad activa no existe como tal, ya que el usuario siempre estará situado en el espacio físico en el que está instalado Unix/Linux. Sin embargo, sí puede estar en un directorio o en otro.

Como en DOS, esta posición determinará la forma de trabajar del usuario a la hora de copiar, borrar, modificar,

Carácter	Acción
?	Al igual que en DOS, sustituye a un solo carácter.
*	Sustituye a cualquier carácter o conjunto de caracteres.
[ ]	Cuando se utilizan como parte de nombres de archivo o directorio, representan un solo carácter de los incluidos entre los corchetes, que se sustituirán por el carácter en el nombre de archivo o directorio en la posición en la que estén estos corchetes. Pueden incluir rangos separados por un guión.
!	Permite negar o excluir caracteres.

**Tabla 13.5.**

## 13.7. Directorios en Unix/Linux

### A. Comandos de administración de directorios

Comando	Sintaxis	Opciones
ls	ls [opciones] [archivo]	[-CFRacdilqrut1]

Tabla 13.6.

Orden que lista por la salida estándar los archivos contenidos en el directorio en el que estamos situados. Los archivos listados son archivos ordinarios, enlaces, o bien subdirectorios del propio directorio.

Al igual que en DOS, puedes obtener un listado de un directorio que no sea el actual. Para ello basta con poner tras del comando la ruta en la que se encuentra el directorio del que quieres obtener el listado. Ejemplo: **ls /home/org/ventas**.

Ejecuta **man ls** para ver los parámetros y su utilización.

La Tabla 13.7 muestra algunos ejemplos con el uso de metacaracteres.

Esta orden es prácticamente igual que su equivalente en DOS.

Para el manejo de esta orden, has de tener muy en cuenta lo que significan dos de los archivos que componen cada directorio, a excepción del raíz.

El fichero **.** hace referencia al directorio actual.

El fichero **..** hace referencia al directorio padre.

Para cambiar de directorio, puedes utilizar las rutas absolutas o relativas, al igual que en DOS. Recuerda que en Unix/Linux los directorios se representan con el símbolo **/**.

- Si ejecutas **cd** sin ningún parámetro ni ruta específica, retornarás automáticamente al directorio **HOME** del usuario.

Comando «ls» con la expresión:	Muestra los archivos y directorios que:
*[ab]*	Contengan el carácter <b>a</b> y el <b>b</b> .
archi[A-P]	Empiecen por <b>archi</b> y que, a continuación, tengan cualquier carácter comprendido entre <b>A</b> (mayúscula) y <b>P</b> (mayúscula).
texto1[1-4]	Se llamen <b>texto1</b> y el siguiente carácter sea un número comprendido entre <b>1</b> y <b>4</b> , ambos inclusive.
[1-3]???t*	Su nombre empieza por un número comprendido entre <b>1</b> y <b>3</b> , seguido de tres caracteres cualesquiera, una <b>t</b> y el resto que se quiera.
[123]???t*	Igual que el anterior.
[dD]ocs	Empiecen por <b>D</b> (mayúscula) o <b>d</b> (minúscula) y el resto del nombre sea <b>ocs</b> .
[;0-9]	No tengan un número como primer carácter. Sólo se mostrarán los archivos que empiecen por una letra.
texto[;123]	Se llamen <b>texto</b> y que el siguiente carácter no sea un <b>1</b> , <b>2</b> o <b>3</b> .
le[ae]me	Se llamen <b>leame</b> o <b>leeme</b> .
[a-z]asa.txt	Empiecen por cualquier letra en minúscula y las tres siguientes sean <b>asa</b> , y la extensión sea <b>.txt</b> : <b>casa.txt</b> , <b>tasa.txt</b> , <b>rasa.txt</b> .
[A-Z]asa.txt	Igual que el ejemplo anterior, pero la primera letra en mayúsculas: <b>Casa.txt</b> , <b>Tasa.txt</b> , <b>Rasa.txt</b> .
[a-zA-Z]asa.txt	Los dos ejemplos anteriores en conjunto: <b>Casa.txt</b> , <b>Tasa.txt</b> , <b>Rasa.txt</b> , <b>casa.txt</b> , <b>tasa.txt</b> , <b>rasa.txt</b> .

Tabla 13.7.

Comando	Sintaxis
cd	Cd [directorio]

Tabla 13.8.

Esta orden se utiliza indicando tras **cd** la ruta a la que quieres moverte.

Para movernos por la estructura de directorios mediante la interfaz gráfica, seguiremos los mismos pasos que conforme lo hacemos en Windows, respetando las opciones que ofrece Unix/Linux.

Comando	Sintaxis
pwd	pwd

Tabla 13.9.

Esta orden indica cuál es el directorio en el que estamos situados actualmente.

Ya sabes que en DOS, gracias al comando **PROMPT \$P\$G**, sabes en todo momento cuál es tu directorio activo. En muchas versiones de Unix/Linux esto no es posible, ya que estemos donde estemos de la ruta de directorios, solamente te aparecerá el símbolo del sistema, **\$** o **#**.



## 13. Sistema operativo multiusuario Unix/Linux...

### 13.7 Directorios en Unix/Linux

En la actualidad algunas versiones de Unix/Linux, y sobre todo de Linux, sí te indican en qué directorio te encuentras, dependiendo del tipo de *shell* cargada.

Esta orden tiene su equivalente en DOS. Es el comando **CD** sin más.

El uso del comando **pwd** no tiene complicación. Basta con poner **pwd** tras el símbolo del sistema y te aparecerá el camino del directorio en el que estamos situados.

Comando	Sintaxis
<b>mkdir</b>	mkdir [directorio]

Tabla 13.10.

Se trata de un comando que sirve para crear nuevos directorios.

Es equivalente al comando **MD** de DOS, pero tiene más potencia, ya que, entre otras cosas, permite crear dentro de un directorio, varios a la vez, mientras que en DOS hay que hacerlo de uno en uno.



Has de considerar que para el uso de esta orden debes tener en cuenta las rutas o caminos.

En primer lugar, si ejecutas el comando **mkdir** seguido del nombre de directorio que quieres crear, éste se creará sobre el directorio actual como un subdirectorio de éste.

Por otro lado, al igual que en DOS, puedes crear directorios desde el directorio actual en otros diferentes a éste. Basta con indicar, tras el comando, la ruta en la que quieres crear el nuevo directorio.

También, como has dicho antes, permite la creación de varios directorios simultáneamente, ejecutando el comando una sola vez. Para ello basta con indicar, tras el comando, los nombres de los subdirectorios que quieres crear, separados por espacios en blanco:

Como puedes ver, el manejo es el mismo que en DOS.

-  Si utilizamos la interfaz gráfica en Unix, puedes crear subdirectorios dentro de otros utilizando una técnica similar a la utilizada en Windows. Abriremos el directorio sobre el que quieres crear uno nuevo y a continuación pulsaremos el botón *File, New Directory*. Y se te pedirá el nombre del nuevo directorio y pulsaremos **Return**.
-  En Linux puedes realizar la misma operación situándote en el directorio en el que quieres crear otro

nuevo. Pulsarás con el botón derecho del ratón sobre cualquier zona de la ventana, que no sea un icono, y seleccionarás *Crear nuevo, directorio*. Introduces, el nombre en la casilla de texto que se muestra a tal efecto y pulsa **Return**.

Comando	Sintaxis
<b>rmdir</b>	rmdir [directorio]

Tabla 13.10.

Se trata de un comando que sirve para eliminar directorios.

El funcionamiento en cuanto a características es muy similar al comando anterior.



Permite eliminar directorios, teniendo en cuenta una característica fundamental: el directorio a ser eliminado tiene que estar en principio vacío, al igual que en DOS.

Pero esta orden permite eliminar directorios que tengan contenido, es decir, directorios que contengan archivos u otros directorios.

En DOS no es posible utilizar el comando **RD** para eliminar directorios que no estén vacíos. En DOS hay que utilizar otra orden. Éste, como ya sabes, es el comando **DELTREE**.

En Unix (no en la versión SCO), puedes utilizar el comando **rmdir** seguido del parámetro **-r**. En Linux lo haremos con el parámetro **-p**. Con éste, lo que haces es eliminar el directorio en cuestión y todo lo que éste contiene, archivos y directorios, de una sola vez.

Recuerda que, al igual que en DOS, para eliminar un directorio tienes que estar situado por encima de él en la estructura, es decir, no puedes eliminar un directorio en el que estás situado. Tendrás que subir un nivel (**cd..**) y posteriormente proceder a su eliminación.

-  Para borrar un directorio en entorno gráfico, te situaremos sobre el icono al que hace referencia, pulsaremos el botón derecho del ratón, y pulsaremos *Discard*. Si queremos eliminar un conjunto de directorios, seleccionaremos varios de ellos, teniendo pulsada la tecla **Ctrl**.
-  En Linux la operación se realiza de la misma forma, si bien una vez seleccionado el directorio o directorios a eliminar, pulsaremos el botón derecho del ratón, y pulsaremos *Eliminar*, o *Enviarlo a la papelera*, de la que posteriormente podrás recuperarlos.

B. Consideraciones acerca de la administración de directorios

En Unix/Linux es importante tener en cuenta que cada vez que se da de alta un usuario en el sistema, el propio sistema operativo, a través de las indicaciones del administrador o superusuario, asigna un directorio de trabajo para ese nuevo usuario. Este directorio es de uso particular, a menos que el administrador indique lo contrario, para ese nuevo usuario.

A estos directorios se les denominan **HOME**. Pero HOME no es el nombre propiamente dicho del directorio, sino una forma de llamar en Unix/Linux al directorio de trabajo de cada usuario.

El lugar en el que se crean estos directorios de trabajo lo elige el superusuario o administrador del sistema. Normalmente, estos directorios se crean encima del directorio **/usr** seguido del nombre o **login** con el que usuario se identifica ante el sistema.

Por ejemplo, el administrador del sistema crea un usuario llamado **contable**. Pues bien, lo normal es que su directorio HOME, es decir, el directorio en el que trabajará, estará en la ruta **/usr** y se llame **contable**. En definitiva, el directorio de trabajo de este nuevo usuario estará ubicado en la ruta **/usr/contable**.

Conviene indicar que esto es lo que suele ocurrir, pero el administrador puede crear el directorio de trabajo para cada usuario en el lugar de la estructura que quiera y con el nombre que quiera. Simplemente, se elige la ruta **/usr** por comodidad y por estructuración lógica del propio sistema.

Por otro lado, cuando damos de baja a un usuario en el sistema, ocurre que el directorio de trabajo que se le asignó en su momento no se elimina. Es evidente que aunque a un usuario se le dé de baja, éste puede tener en su directorio de trabajo programas, archivos o información que no tiene necesariamente que ser eliminada.

En Unix, si deseamos eliminar el directorio de trabajo asignado a ese usuario, tendremos que borrarlo manualmente ejecutando la siguiente orden: **\$rmdir -r /usr/contable**.

En Linux sería del siguiente modo: **\$rmdir -p /usr/contable**.

Se incluye este parámetro, ya que lo normal es que el directorio tenga algún tipo de información.

El uso de las órdenes de manejo de directorios en DOS es muy similar al de las del Unix/Linux. Recuerda que el DOS nació del Unix/Linux como sistema operativo para ordenadores personales. Asimismo debes recordar que en Unix/Linux los nombres asignados a archivos y directorios varían si se ponen en mayúsculas o minúsculas.

13.8 Archivos en Unix/Linux

A continuación vas a ver qué operaciones puedes realizar sobre archivos de Unix/Linux, es decir, cómo crearlos, borrarlos, modificarlos, etcétera.



Comando	Sintaxis	Opciones
cp	<ul style="list-style-type: none"><li>cp [opciones] [archivo] ruta</li><li>cp [opciones] [archivo] [directorio]</li></ul>	[-fipRr]

Tabla 13.13.

Con esta orden se puede visualizar en pantalla el contenido de un archivo. Esta orden es equivalente al comando **TYPE** de DOS.

Con esta orden se pueden visualizar archivos que están en el directorio activo, o archivos que se encuentran en otra ubicación. Para ello es suficiente con indicar la ruta o camino en el que se encuentra el fichero a visualizar. Hay que tener en cuenta, que esta orden se suele utili-

zar con archivos de texto, ya que cualquier otro tipo de archivo contiene caracteres que, aunque se visualicen, no podrás entender.

-  Para ver el contenido de un archivo en entorno gráfico, pulsaremos dos veces sobre el icono que lo representa o pulsaremos el botón derecho del ratón en *Edit*.
-  En Linux, se procede de igual forma, con la posibilidad añadida de elegir el tipo de aplicación con el que se quiere visualizar el archivo, es decir, algo similar a Windows con la opción *Abrir con*, o de hacer doble clic sobre el mismo si éste fue creado con alguno de los editores existentes, como **Kwrite**.

Comando	Sintaxis	Opciones
cat	cat [opciones] [archivo]	[-AbeEnstTuv]

Tabla 13.12.

## 13. Sistema operativo multiusuario Unix/Linux...

### 13.8 Archivos en Unix/Linux

Esta orden, que sirve para copiar archivos, es muy similar al comando **COPY** de DOS.

Ejemplo: **\$cp doc1 doc2**

Como se puede ver, se utiliza casi igual que el comando **COPY**. En este caso, estamos copiando el archivo *doc1* en el mismo directorio activo con el nombre *doc2*.


Has de tener en cuenta que si en el directorio activo existiera un subdirectorio llamado *doc2*, el archivo *doc1* se copiaría en el subdirectorio *doc2* con el mismo nombre.

Al igual que el resto de órdenes de manejo de archivos, para copiar un archivo de un sitio a otro, antes de los nombres de archivo en cuestión, puedes incluir la ruta en la que se encuentran éstos.

Ejemplo: **\$cp /usr/contable/doc1 /dev/doc2**

Con esta orden, lo que estamos haciendo es copiar el archivo *doc1*, que se encuentra en el directorio **/usr/contable**, en el directorio **/dev** con el nombre *doc2*.


Si en esta orden no indicásemos el nombre destino de fichero, es decir, no pusiéramos *doc2*, el archivo se copiaría en el directorio **/dev** con el nombre del archivo original, igual que ocurre en DOS.

-  Para copiar archivos en entorno gráfico, seleccionarás el archivo (o los archivos) y pulsas el botón derecho del ratón y elegirás la opción *Copy to...*

A continuación aparecerá una ventana en la que se te solicitará el destino de los archivos.

En este caso solamente te queda indicar la ruta absoluta en la que quieres copiar el archivo o archivos seleccionados.

Si quieres realizar una copia del archivo sobre el mismo directorio, selecciona el archivo a copiar, pulsas el botón derecho y la opción *Duplicate*. Se te solicitará el nombre con el que quieres copiar el archivo, y la operación finaliza.

-  En Linux, el procedimiento es muy similar al descrito anteriormente.


 Comando	Sintaxis	Opciones
<b>mv</b>	<ul style="list-style-type: none"><li>• mv [opciones] [origen] [destino]</li><li>• mv [opciones] [origen] [destino]</li></ul>	<b>[ -fi ]</b>

Tabla 13.16.

Con esta orden puedes eliminar archivos. Es equivalente al comando **DELETE** o **DEL** de DOS.



El uso de esta orden no tiene mayor complicación. Basta con especificar tras el comando, el nombre del archivo a eliminar, y si fuese necesario, la ruta o ubicación del mismo dentro de la estructura de directorios.

Ejemplo: **\$rm /usr/contable/doc1**

Con esta orden estamos eliminando el archivo *doc1* que no esté situado en el directorio activo.

Para eliminar un archivo o conjunto de archivos en entorno gráfico, los seleccionaremos y pulsaremos el botón derecho. A continuación elegimos la opción *Discard*. Los archivos eliminados, al igual que en Windows, pasan a la papelera de Unix (**Trash/Papelera**), cuyo funcionamiento es parecido a la de Windows. En Linux disponemos de la opción de *Eliminar* o *Mover a la papelera*.

La papelera de Unix/Linux solamente funciona en entorno gráfico, no en entorno texto.

-  Observa la papelera de Unix (véase la Figura 13.10).
-  Y en Linux (Figura 13.11).
- Las opciones son equivalentes a las de Windows, ya que puede restaurar archivos eliminados, eliminarlos definitivamente, etcétera.

 Comando	Sintaxis	Opciones
<b>rm</b>	rm [opciones] [archivo]	<b>[ -fipRr ]</b>

Tabla 13.23.

Orden equivalente a **MOVE** en DOS. Con esta orden puedes cambiar el fichero de un directorio a otro. Ten en cuenta que esta orden no copia el archivo, sino que lo mueve de un sitio a otro. Solamente existirá una copia del mismo, a diferencia del comando **cp**, que sí lo duplica.

Ejemplo: **\$mv /usr/contable/doc1 /dev/doc2**

Esta orden pasa el archivo *doc1*, situado en la ruta **/usr/contable**, a un nuevo directorio **/dev** con el nombre *doc2*.

Si no se especifica nombre de destino, el archivo pasa al directorio con el mismo nombre.

Al igual que el comando **cp**, si *doc2* fuese un nombre de directorio, el archivo *doc1* pasaría al directorio **/dev/doc2** con el nombre de *doc1*.

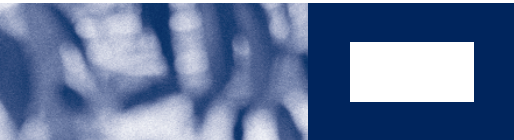


Fig. 13.10. Papelera en Unix SCO.



Fig. 13.11. Papelera en Linux SUSE.







Si lo que quieres es cambiar el nombre de un archivo dentro del directorio activo, has de tener en cuenta que es obligatorio incluir el nuevo nombre, es decir, el nombre de destino. Solamente puedes omitir el nombre de archivo destino cuando lo mueves a una ruta diferente del directorio activo.

Ejemplo: **\$mv doc1 doc2**

En este caso, el archivo *doc1* pasa a llamarse *doc2*. Evidentemente, *doc1* desaparece.

Además, debes tener en cuenta que si *doc2* fuese un directorio, *doc1* pasaría al directorio *doc2* con el nombre *doc1*.

-  La operación en entorno gráfico se realiza arrastrando el icono correspondiente de un directorio a otro, igual que en Windows, o pulsando el botón derecho sobre el icono o iconos seleccionados y eligiendo la opción *Move to*. A continuación se introduce la nueva ruta absoluta a la que irá el fichero o ficheros seleccionados.
-  De igual forma a como se hace en Windows, mediane las opciones de *Cortar* y *Pegar*.


 Comando	Sintaxis	Opciones
<b>ln</b>	<ul style="list-style-type: none"><li>• ln [opciones] [origen] [destino]</li><li>• ln [opciones] [origen] [directorio]</li></ul>	[-f]

Tabla 13.16.

Con esta orden consigues algo que en DOS no es posible hacer.

Lo que haces gracias al uso del comando es asignar a un archivo existente un segundo nombre. Es decir, no es como el comando **mv** ni el comando **cp**. Es una especie de mezcla, ya que el archivo al que le aplicamos esta orden aparecerá en realidad dos veces, es decir, el nombre original y el nuevo que le acabamos de asignar.

Lo característico de esta orden es que el contenido del archivo solamente existe una vez. Concretamente, es el archivo original el que dispone de la información. El nuevo archivo creado solamente es un **enlace** hacia el archivo original.

Es lo que en algunas ocasiones en informática denominamos **alias**. El nuevo archivo existe como tal, pero el contenido real del archivo está en el archivo primero, ya que el nuevo solamente lo referencia.

Si editamos el nuevo archivo, para el usuario que lo edita estará trabajando con ese archivo, pero en realidad lo estará haciendo con el contenido del archivo original al que le aplicamos el comando **ln**.

Ejemplo: **\$ln doc1 doc2**

El archivo *doc1* existirá con dos nombres, lo que ocurre es que *doc2* no es realmente el archivo que contiene la información, sino *doc1*.

El uso de esta orden es importante cuando quieres que varios usuarios tengan acceso al contenido de un archivo y que todos ellos puedan modificarlo, imprimirlo, etc. Con esto se consigue que la información real del archivo no se duplique o triplique, ya que lo que haces es asignar a cada usuario un nombre de archivo que en realidad es un enlace al archivo original.

PSe trata de un comando parecido al filtro **MORE** de DOS.

Se utiliza para visualizar el contenido del archivo, similar al comando **cat**. La gran diferencia es que si el archivo ocupa más de una pantalla, con el comando **cat** no lo podrás visualizar desde el principio, es decir, las líneas de archivo se te mostrarán por pantalla, pero solamente se quedarán las últimas.


 Comando	Sintaxis	Opciones
<b>more</b>	more [opciones] [-num] [+ / modelo] [+ numlinea] [archivo]	[-dlfpctu]

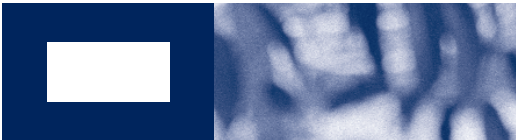
Tabla 13.17.

Con **more** haces lo mismo, pero visualizando el contenido del archivo pantalla a pantalla.

Al final de cada pantalla, puedes pulsar la barra espaciadora para ver otra nueva pantalla con el contenido del archivo. Si quieres salir de la visualización del archivo antes de que realmente se haya completado la operación, en vez de pulsar la barra espaciadora, pulsa la combinación de teclas **CTRL + D**.

Ejemplo: **\$more doc1**

Te muestra el contenido del archivo *doc1*, pantalla a pantalla. Es evidente que si el archivo se encuentra en un directorio distinto al directorio activo, podrás incluir la ruta en la que éste se encuentra sin ningún problema.



### 13. Sistema operativo multiusuario Unix/Linux...

#### 13.8 Archivos en Unix/Linux

Para ver la diferencia con el filtro MORE del DOS, te indico un ejemplo de cómo se produce en DOS la salida paginada de un archivo.

C:\>TYPE DOC1MORE

En realidad, el resultado de esta orden es el mismo que la equivalente en Unix/Linux. Lo que sí que tienes que tener claro que en DOS tienes que utilizar el comando propia de visualización de archivos y pasarla por el filtro MORE para obtener tal resultado.

En Unix/Linux, el comando **more** es una orden propia o un filtro, igual que en DOS.


 Comando	Sintaxis	Opciones
pg	pg [-número [-p cadena] [+ numlinea] [+/- modelo] [archivo]	

Tabla 13.18.


 Comando	Sintaxis	Opciones
less	less [opciones] [-b espacio] [-h líneas] [-j línea] [-k archivoClave] [-K conjuntoDeCaracteres] [-{oO} archivoDeRegistro] [-p modelo] [-P símboloDe Comandos] [-t indicador] [-T archivoDeIndicadores] [-x ficha,...] [-y líneas] [-z líneas] [archivo]	[[-][+] aABcCdeEFgGiIJLmMnNq QrRsSuUVwWX]

Tabla 13.19.

Se trata de una orden similar a la anterior, ya que en realidad hace lo mismo. La diferencia entre **pg** y **more** es que **pg** solamente se utiliza con este fin, y **more** lo puedes utilizar, al igual que en DOS, como filtro.

Ejemplo: **\$pg doc1**, en Unix, o **\$less doc1**, en Linux.

Hemos de insistir nuevamente en que si el archivo no se encuentra en el directorio actual, se puede indicar la ruta en la que está.


 Comando	Sintaxis	Opciones
sort	sort [opciones] [archivo]	[-bdfgiMnrckm osStTuz]

Tabla 13.20.

Como el filtro **sort** en DOS, esta orden lo que hace es ordenar alfabéticamente las líneas que contiene un archivo.

La ordenación es de menor a mayor y se realiza por el primer campo que contiene el archivo. Debes tener en

cuenta que hay diferencia entre caracteres en mayúscula y minúscula.

Evidentemente, existen parámetros como **-r** para realizar la ordenación en orden inverso. Otro parámetro es poner **+n**, es decir, un número entero positivo, lo que indica que la ordenación se realizará por el campo que ocupa la posición que acabamos de señalar.

Es conveniente, en esta orden, recurrir a la ayuda de Unix/Linux para ver la potencia que tiene y la cantidad de parámetros para realizar salidas de archivos ordenadas de múltiples formas.

Ejemplo: **\$sort -r +2 doc1**

Esta orden visualiza por pantalla el contenido ordenado del fichero *doc2* en orden descendente, tomando el segundo campo como referencia para la ordenación.


 Comando	Sintaxis	Opciones
tail	tail [opciones] [archivo]	[-cFngsv] [--mas] [--pid]

Tabla 13.21.

En este caso, la orden muestra por pantalla las últimas líneas de un archivo concreto.

Ejemplo: **\$tail doc1**

Esta orden muestra las últimas 10 líneas del archivo *doc1* por pantalla. Si quieres mostrar un número determinado de líneas, incluirás el parámetro **-n**; indica el número de líneas a mostrar.

Si incluyes el parámetro **+n**, estarás indicando al comando que te muestre, a partir de la línea n, todas hasta el final.

Ejemplo: **\$tail -8 doc1**

Te muestra por pantalla las últimas 8 líneas del archivo *doc1* situado en el directorio activo.


 Comando	Sintaxis	Opciones
nl	nl [opciones] [archivo]	[-bdfhiInpsvwl]

Tabla 13.22.

Esta orden tiene utilidad para los programadores de Unix/Linux. En realidad, lo que hace es mostrar por pantalla las líneas de un archivo numeradas secuencialmente.

Ejemplo: **\$nl doc1**


 Comando	Sintaxis	Opciones
<b>wc</b>	wc [opciones] [archivo]	[-cmlLw]

Tabla 13.23.

Esta orden suele utilizarse conjuntamente con las tuberías (caracteres **pipe** del DOS) y con los redireccionamientos de entrada y salida para contabilizar las líneas de un archivo concreto.

Ejemplo: **\$ wc-doc1**

Como se aprecia en el comando, lo que haces es que **wc** tome como datos de entrada el resultado del comando **cat**; por lo tanto, el número de líneas.


 Comando	Sintaxis	Opciones
<b>grep</b>	<ul style="list-style-type: none"><li>grep [opciones] PATTERN [archivo]</li><li>grep [opciones] [-e PATTERN   -f archivo] [archivo]</li></ul>	[-AaBdCbDEeF PfGHHiILmno qRsuUvVxXyZ]

Tabla 13.24.

Orden equivalente al comando **FIND** de DOS. Gracias a ella puedes buscar palabras y cadenas de caracteres en un archivo especificado.



Ejemplo: **\$grep 'pedro' 'pablo' doc1**

Es importante tener en cuenta que los nombres de las cadenas a buscar se introducen entre comillas simples o dobles.



Con esto lo que consigues es que se te muestren por pantalla todas aquellas líneas que contengan el nombre *pedro pablo*.

Si ejecutas lo siguiente: **\$grep pedro pablo doc1**, se buscara la palabra *pedro* en los archivos *pablo* y *doc1*.

Recuerda la diferencia entre mayúsculas y minúsculas.

-  Por ejemplo, para localizar una cadena de caracteres dentro de un archivo, seleccionaremos la opción *Search, Find*, o el icono .

En esta ventana introduciremos la cadena de caracteres a buscar y las opciones adecuadas. Conviene insistir en que todo esto se maneja de forma similar a como se hace con cualquier editor de textos, a excepción de **vi** en entorno texto.

-  En Linux basta con pulsar en el icono , o *Editar, Buscar* en la línea de menús. En cualquiera de los dos casos, introduces la cadena de caracteres a buscar dentro del documento y pulsas *Aceptar*.

## 13.9 Permisos y derechos en Unix/Linux

Ya viste en DOS y Windows lo que eran los atributos de los archivos y directorios.

En DOS, al ser un sistema monousuario, monotarea y monoprogramación, los atributos se reducían a cuatro. Es decir, la seguridad en un sistema DOS es casi nula, ya que se supone que será siempre el mismo usuario el que utilizará un ordenador concreto y que ningún otro usuario tendrá acceso a la información almacenada en éste.

Esto obedece fundamentalmente a que el sistema DOS no está diseñado para trabajar en red, de tal forma que los usuarios no comparten información ni recursos hardware de ningún tipo. No existe un ordenador principal que sea servidor de archivos, por lo que la seguridad de la información almacenada en cada equipo no es prioritaria.

Pero en Unix/Linux, al ser un sistema diseñado fundamentalmente para trabajo en red, la seguridad de la información que almacenemos en los ordenadores centrales o servidores es fundamental, ya que muchos usuarios

tendrán o podrán tener acceso a parte de los recursos software y hardware que están gestionados en estos ordenadores.

Concretamente en Unix/Linux, los permisos o derechos que los usuarios pueden tener sobre determinados archivos contenidos en él o en los ordenadores principales se establece en tres niveles claramente diferenciados. Estos tres niveles son los siguientes:

- Permisos del **propietario**.
- Permisos del **grupo**.
- Permisos del **resto** de usuarios.

### A. Usuarios del sistema Unix/Linux

Para tener claros estos conceptos, has de recordar que en los sistemas en red siempre existe la figura del **administrador** o **superusuario**.

## 13. Sistema operativo multiusuario Unix/Linux...

### 13.9 Permisos y derechos en Unix/Linux

Este administrador es el encargado de crear usuarios, dar de baja a usuarios y, fundamentalmente, de establecer los privilegios que cada uno de ellos tendrá en el sistema.

Estos privilegios se establecen tanto para el directorio HOME de cada usuario como para los directorios y archivos a los que el administrador decida que el usuario pueda acceder.

El **propietario** es aquel usuario que genera o crea un archivo dentro de su directorio de trabajo en un directorio sobre el que tenga derechos. Cada usuario tiene la potestad de crear, por defecto, los archivos que quiera dentro de su directorio de trabajo. En principio, él y solamente él será el que tenga acceso a la información contenida en los archivos y directorios que hay en su directorio HOME.

Además, lo normal es que cada usuario pertenezca a un **grupo** de trabajo. De esta forma, cuando se gestiona el grupo, se gestionan todos los usuarios que pertenecen a éste. Es decir, es más fácil integrar varios usuarios en un grupo al que se le conceden determinados privilegios en el sistema, que asignar los privilegios de forma independiente a cada usuario.

Por último, también los privilegios de archivos contenidos en cualquier directorio de la estructura, pueden tenerlos otros usuarios que no pertenezcan al grupo de trabajo en el que está integrado el archivo en cuestión. Es decir, a los usuarios que no pertenecen al grupo de trabajo en el que está el archivo, pero que pertenecen a otros grupos de trabajo, se les denomina **resto** de usuarios del sistema.

Evidentemente, un usuario que no pertenezca a nuestro grupo, será considerado **resto** para nuestro grupo. Lógicamente, nosotros seremos considerados **resto** para su grupo.

Recuerda que, en cada sistema de red, existirá un superusuario o administrador del sistema que tendrá privilegios sobre todos los archivos directorios, dispositivos, etc., que existan en el sistema. Éste será el que dé o quite privilegios según las necesidades de la organización en la que estemos integrados.

Por ejemplo, en Internet, es sabido que los usuarios de la red tienen acceso a la mayoría de las páginas Web. Pero claramente vemos que sólo puedes leerlas, es decir, visualizarlas. Ningún usuario puede modificar o borrar una página así como así. Solamente el que la diseña decide quién puede realizar estas operaciones.

## B. Tipos de permisos en Unix/Linux

Antes de indicar cómo se establecen los permisos en Unix/Linux, tienes que saber cómo se pueden diferenciar los diferentes tipos de archivos que el sistema puede contener.

Cada archivo en Unix/Linux queda identificado por 10 caracteres. De estos 10 caracteres, el primero por la izquierda hace referencia al tipo de archivo. El resto, es decir, los 9 siguientes, de izquierda a derecha y en bloques de 3, hacen referencia a los permisos que se le conceden, respectivamente, al propietario, al grupo y al resto.

El primer carácter de los archivos puede ser el siguiente:

Permiso	Identifica
-	Sin permiso.
r	Permiso de lectura.
w	Permiso de escritura.
x	Permiso de ejecución.

Tabla 13.26.

Estos tipos de archivos son los más estandarizados en determinados sistemas. Hay versiones de Unix/Linux que no incluyen estos archivos, y por contra incluyen otros. Se ha de tener en cuenta que, básicamente, para conocer la gestión de permisos es preciso centrarse en archivos ordinarios y directorios.

Los siguientes nueve caracteres son los permisos que se les concede a los usuarios del sistema. Cada tres caracteres, se referencian los permisos de propietario, grupo y resto de usuarios.

Los caracteres que definen estos permisos son los siguientes:

Permiso	Identifica
-	Archivo.
d	Directorio.
b	Archivo de bloques especiales.
c	Archivo de caracteres especiales.
l	Archivo de vínculo o enlace.
p	Archivo especial de cauce.

Tabla 13.25.

Que los permisos se asignen a archivos ordinarios o a directorios hace que su significado no sea exactamente el mismo.



## 13. Sistema operativo multiusuario Unix/Linux...

### 13.9 Permisos y derechos en Unix/Linux

#### C. Permisos para archivos

- **Lectura:** permite, fundamentalmente, visualizar el contenido del archivo con órdenes como `ls`, `cat`, `pg`, `more` y `cp`.
- **Escritura:** permite modificar el contenido del archivo. El archivo se puede editar, por ejemplo, con `VI`, y puede modificarse su contenido sin ningún problema.
- **Ejecución:** permite ejecutar el archivo como si de un programa ejecutable se tratase. Estos permisos se suelen asignar a archivos *SHELL*, es decir, a archivos que realizan funciones propias del sistema operativo, como copias de seguridad, análisis de la integridad del sistema, etcétera.

#### D. Permisos para directorios

- **Lectura:** Permite saber qué archivos y directorios contiene el directorio que tiene este permiso. Concretamente, podrás utilizar órdenes como `ls`.
- **Escritura:** permite crear archivos en el directorio, bien sean archivos ordinarios o nuevos directorios. Se pueden borrar directorios, copiar archivos en el directorio, mover, cambiar el nombre, etcétera.
- **Ejecución:** permite situarse sobre el directorio para poder examinar su contenido, copiar archivos de él. Si además se dispone de los permisos de escritura y lectura, se podrán realizar todas las operaciones posibles sobre archivos y directorios.

Si no se dispone del permiso de ejecución, aunque utilicemos el comando `cd` para situarnos en el directorio, esta acción será denegada. Permite delimitar el uso de un directorio como parte de una ruta. Si el permiso de ejecución de un directorio está desactivado, se podrá ver su contenido (si se cuenta con permiso de lectura), pero no se podrá acceder a ninguno de los objetos contenidos en él, pues para ello este directorio es parte del camino necesario para resolver la ubicación de sus objetos.



#### E. Gestión de permisos en Unix/Linux

Para poder examinar los permisos y derechos, se pueden utilizar diferentes comandos, pero en primer lugar tienes que saber qué indica cada uno de ellos.

La forma más gráfica y clara de ver los permisos de cada fichero es utilizar el comando `ls -l`. Con su ejecución se

mostrará toda la información de los archivos del directorio, a excepción de los ocultos, con sus permisos. Si quieres visualizar incluso los ocultos, ejecutarás el comando `ls -la`.

A continuación se muestran los privilegios de un archivo en entorno gráfico en Linux (véase la Figura 13.12).

-  Observa cómo se muestran los permisos. Para ello, selecciona el archivo o directorio y *Propiedades* (véase la Figura 13.13).
-  En Linux, basta con seleccionar las *Propiedades* del archivo o directorio y *Permisos*. Se mostrarán entonces los permisos del archivo o directorio seleccionado (Figura 13.14).

Hay que tener en cuenta que cuando damos de alta un usuario en el sistema, le concedemos de forma automática unos privilegios. Estos privilegios, por supuesto, no serán totales, es decir, los usuarios no dispondrán, normalmente, de los mismos permisos y derechos del superusuario.

Cuando creas el usuario, el sistema genera por defecto los privilegios del usuario para manejo de archivos y para manejo de directorios. Evidentemente, éstos pueden ser modificados por el administrador, pero el sistema genera unos privilegios más o menos válidos para la mayoría de las operaciones que cada usuario realizará sobre su directorio, sus archivos y sobre los directorios y archivos del resto de usuarios.

Los derechos, privilegios o autorizaciones que el sistema genera por defecto varían según las versiones de Unix/Linux que utilices. En general, son los siguientes:

- Para archivos: **-rw- r-- r--**
- Para directorios: **-rwx rwx rwx**

Conviene insistir en que no son los mismos en todas las versiones de Unix/Linux. Son configurables por el administrador.

Estos privilegios te permiten crear archivos, copiar archivos, borrar archivos, crear nuevos directorios, etc. Afectan a los objetos que se generen en las ubicaciones en las que tienes derechos para ello.

La Tabla 13.27 muestra los comandos que el sistema tiene para poder modificar los privilegios y derechos sobre archivos y directorios.

A los 10 caracteres que acompañan a cada archivo y directorio en Unix/Linux, se le suele denominar **máscara**.

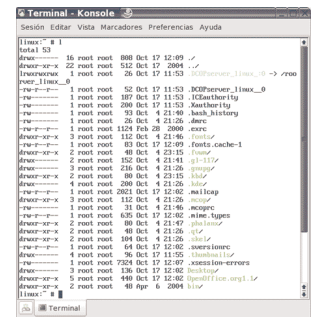


Fig. 13.12. Listado de archivos en Linux.

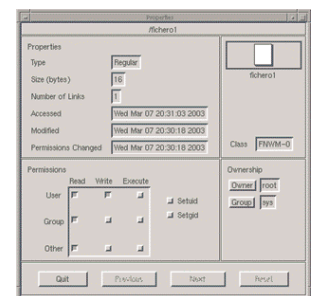


Fig. 13.13. Propiedades de un archivo de Unix.

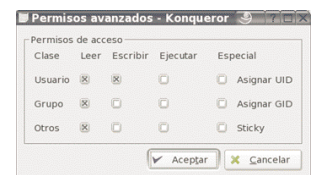


Fig. 13.14. Permisos de archivo en Linux.

## 13. Sistema operativo multiusuario Unix/Linux...

### 13.9 Permisos y derechos en Unix/Linux

Comando	Sintaxis	Opciones
<b>Chmod</b> (sintaxis no numérica)	chmod [opciones] [modo] [archivo]	[-R]

Tabla 13.27.

La orden **chmod** permite modificar esta máscara para que se puedan realizar más o menos operaciones sobre archivos o directorios. Esta orden es fácil de utilizar si tienes claro que los bloques de 3 caracteres a partir del segundo de la máscara corresponden, respectivamente, a su propietario, al grupo y al resto de usuarios (véase la Tabla 13.27).

La orden **ATTRIB** de DOS es la única que tiene algo de semejanza con la gestión de privilegios, aunque en DOS solamente puedes otorgar a un archivo o directorio unos «privilegios» muy escasos.

Con esta orden DOS se añaden privilegios con el carácter + y se eliminan los privilegios o atributos con el carácter -. Pues bien, en Unix/Linux se hace de la misma forma, aunque con una pequeña modificación, véase la Tabla 13.28.

Con el comando **chmod** puedes quitar o eliminar derechos a cada tipo de usuarios. Para ello tienes que saber cómo hace referencia a cada usuario, véase la Tabla 13.29.

Si no se especifica el tipo de usuario al que le quieres quitar, poner o asignar privilegios, lo que harás será realizar la operación a todos los usuarios simultáneamente.

Carácter	Acción
-	Elimina derechos.
+	Agrega derechos.
=	Asigna permisos especificados.

Tabla 13.28.

Carácter	Actúa sobre
u	Propietario.
g	Grupo al que pertenece el usuario.
o	Resto de usuarios.

Tabla 13.29.

Comando	Resultado
<b>\$chmod g+x doc1</b>	Concede privilegios de ejecución al grupo al que pertenece el archivo llamado doc1.
<b>\$chmod rwx doc1</b>	Se asignan los privilegios de lectura, escritura y ejecución a todos los usuarios para el archivo doc1.
<b>\$chmod go-wx doc1</b>	Se quitan los privilegios de escritura y ejecución al grupo y al resto de usuarios del archivo doc1.
<b>\$chmod =x doc1</b>	Asigna a todos los usuarios el permiso de ejecución. Este comando se podría escribir <b>\$chmod ugo+x doc1</b> , como es evidente.
<b>\$chmod = doc1</b>	Quita todos los privilegios a todos los usuarios del archivo doc1.

Tabla 13.30.

Comando	Sintaxis	Opciones
<b>Chmod</b> (sintaxis numérica)	chmod [opciones] [modo] [archivo]	[-R]

Tabla 13.31.

La sintaxis de esta orden es muy similar al comando **ATTRIB** de DOS. Se especifica el comando, seguido del tipo de usuario sobre el que quieres actuar, el carácter +, - o =, tipo de permiso y archivo o directorio (véase la Tabla 13.30).

Ejemplos:

Hay otra forma de utilizar el comando **chmod** que, para muchos usuarios, resulta más cómoda, aunque a priori sea algo más compleja de entender.

Para ello has de tener en cuenta que cada uno de los tres caracteres que representan los privilegios para cada tipo de usuarios se puede representar mediante la combinación de tres dígitos en octal.

Recuerda que cuando viste los sistemas de numeración, los caracteres binarios tenían correspondencia directa con dígitos octales y hexadecimales directamente.

Así que en base 8 los guarismos utilizados son desde el 0 al 7, ambos incluidos. Ten presente que un dígito octal lo puedes representar con sus correspondientes dígitos en binario.

Recuerda que: **2 elevado a cero = 1**, **2 elevado a uno = 2**, y **2 elevado a dos = 4**.

Situados posicionalmente los dígitos binarios según el exponente de menor a mayor, de derecha a izquierda y empezando por el exponente cero, puedes obtener la correspondencia con los dígitos octales.

Recuerda la Tabla 13.32.

Dígito octal	Dígitos binarios		
	2 <sup>2</sup>	2 <sup>1</sup>	2 <sup>0</sup>
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Tabla 13.32.

Así, por ejemplo, el dígito 3 en octal tiene una correspondencia con el binario en la combinación 010. Es decir, los bits uno multiplican a la potencia a la que corresponden posicionalmente, y los bits cero, no.

$$0 * 2^0 + 1 * 2^1 + 0 * 2^2 = 3 \text{ (en octal)}$$

## 13. Sistema operativo multiusuario Unix/Linux...

### 13.9 Permisos y derechos en Unix/Linux

Cabría pensar qué tiene que ver esto con los privilegios. Pues bien sencillo. Imagina por un momento que los dígitos binarios no tienen correspondencia con las potencias de 2. Supón ahora que la correspondencia es posicional con los privilegios de los archivos. Para ello, considera la siguiente tabla (véase la Tabla 13.33).

Observa la correspondencia entre el dígito octal, con sus caracteres binarios, y los privilegios que representan.

Al igual que en los sistemas de numeración, cuando el dígito binario es un 1, indica que esa potencia entra a formar parte del dígito octal, y cuando es un 0, indica que no entra a formar parte.

De forma análoga, supón el número 5 en octal. Éste se corresponde posicionalmente con los bits 101. Numéricamente hablando, este número sería la suma de las potencias  $1 * 2^2 + 0 * 2^1 + 1 * 2^0 = 5$ .

Si utilizas el mismo número, puedes apreciar que el 5 se correspondería con la siguiente combinación de privilegios, es decir,  $1 * r + 0 * w + 1 * x = r - x$ .

En cuanto a los privilegios de Unix/Linux, sabes que los puedes otorgar a tres tipos diferentes de usuarios: al propietario, al grupo y al resto. Pues nada más sencillo que utilizar un dígito octal para cada uno de ellos. Es decir, si utilizamos tres dígitos octales, posicionalmente el primero servirá para asignar privilegios al propietario; el segundo, para asignar privilegios al grupo, y el tercero, al resto de usuarios.

Considera la siguiente cifra: 750. Si la descomponemos en bloques de 3 bits cada una, sería: 111 101 000. Si a continuación haces corresponder cada bit 1 con asignación y cada bit 0 con eliminación de privilegios, obtendríamos el siguiente resultado: `rxw r-x ---`

El propietario tendría todos los privilegios el grupo de usuarios, lectura y ejecución, y el resto de usuarios, nada.

Supón que quieres asignar al archivo `doc1` la siguiente máscara: `rx- r-- rxw`

Claramente puedes ver que el resultado intermedio sería el siguiente: 110 100 111, que pasado a octal se correspondería con 647.

Por lo tanto, si ejecutas el comando `chmod` de la siguiente forma: `$chmod 647 doc1`, estarás asignando los privilegios de lectura y escritura al propietario, de lectura al grupo y todos los privilegios al resto de usuarios.

En definitiva, el que asigna privilegios deberá tener en cuenta únicamente cuáles de ellos quiere asignar, y realizar un pequeño análisis de su equivalencia con el número en octal y el correspondiente desglose en binario. Nada más.

La forma de asignar o modificar privilegios a archivos y directorios en entorno gráfico es visualizar las *Propiedades* del archivo o directorio y activar o no las casillas correspondientes a los permisos de propietario, grupo u otros.


 Comando	Sintaxis	Opciones
<code>umask</code>	<code>umask [máscara]</code>	<code>[-lpPvVsSfqux]</code>

Tabla 13.34.

Con esta orden puedes cambiar la máscara de los privilegios que, por defecto, se asigna a un usuario para la creación de archivos y directorios.

Esta orden es inversa a **chmod**. Concretamente, lo que hace no es asignar privilegios, sino restringirlos.

Esta orden solamente tiene formato numérico, no como **chmod**, que también lo tiene no numérico.

Su uso es sencillo: después del comando se especifica un número en octal de tres dígitos, que indica qué privilegios se quitan y a quién. Parecido a **chmod**.

A diferencia de **chmod**, en donde puedes especificar un archivo o directorio concreto, con **umask** lo que haces es restringir, en general, los privilegios para la creación de archivos y directorios. Es decir, esta orden no puede aplicarse a un archivo o directorio en particular; se aplica, en general, para un usuario concreto, aunque en casos determinados se puede aplicar específicamente a un archivo o directorio.

En esta orden has de tener en cuenta cuáles son los privilegios que por defecto asigna el sistema. Ya viste anteriormente que los privilegios por defecto eran:

Para archivos: **-r w - rw -**

Para directorios: **-rwx rwx rwx**

y sus equivalentes numéricos:

Para archivos: **666**

Para directorios: **777**

Dígito octal	Privilegios		
	r	w	x
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Tabla 13.33.

En resumen, lo que hace esta orden es restar de la máscara, por defecto, el valor que especifiquemos tras ella.

Ejemplo: **\$umask 022**

El resultado sería, para archivos, restar 022 a 666 obteniendo como resultado la máscara deseada: 644, que se traduce en **rw- r- - r- -**.

Es decir, es lo mismo que poner: **\$chmod 644 o \$chmod g+rw u+r o+r**

Esta orden, al igual que el fichero **AUTOEXEC.BAT** en DOS, se suele incluir en el fichero **.profile** de Unix/Linux (habitualmente en **/etc/profile**), del que más adelante verás que es equivalente al **AUTOEXEC.BAT** de DOS. De esta forma consigues que cada usuario solamente pueda hacer lo que el administrador del sistema desea de forma automática.

La orden **umask** sin parámetros muestra cuál es el valor de la máscara a restar de los valores por defecto para gestión de archivos y directorios.

Con el comando **umask** solamente se utilizan tres dígitos. Puede utilizarse un cuarto (el primero), con el que determinados si la máscara la asignamos al grupo o a los usuarios.

Este comando se ejecuta en entorno gráfico, fundamentalmente cuando damos de alta a un usuario o grupo, y definimos la máscara por defecto.


 <b>Comando</b>	Sintaxis	Opciones
<b>chown</b>	chown [opciones] user [:grupo] archivo	[-R]

Tabla 13.35.

Como ya sabes, cada archivo tiene un propietario y suele coincidir con la persona que lo creó. El propietario suele tener la mayoría de los privilegios sobre ese archivo para manipularlo a su gusto. Pero el resto de usuarios no disponen de tantos derechos, como es evidente.

El propietario del archivo lo será mientras no se indique lo contrario. Si, por ejemplo, copiamos un archivo sobre el directorio HOME de otro usuario, el propietario no cambia, sigue siendo el creador del archivo, aunque este archivo se haya copiado en varios sitios.

Con el comando **chown** se puede cambiar el propietario de un archivo, siempre y cuando tengas esa necesidad.

La operación de cambio de propietario solamente la podrá hacer el administrador del sistema o el usuario que lo creó, es decir, su propietario actual.

Supón que en nuestro sistema tienes un archivo que pertenece al usuario **contable**. Si visualizas sus propiedades con el comando **ls -l**, se te mostrará algo como lo siguiente:

**-rw-r--r-- 1 contable grupo 1000 Aug 12 12:00 doc1**

Supón que en el sistema existen más usuarios, por ejemplo el usuario **nomina**, y quieres asignar a este usuario la propiedad del archivo *doc1*; ejecutarías la siguiente orden:

**\$chown nomina doc1**

lo que indica que el archivo *doc1* pasa a pertenecer con todos los privilegios disponibles al usuario nomina, que desde ahora es su propietario. Contable se queda sin los privilegios que tenía cuando creó el fichero.

En algunas versiones, el comando **chown** incluye el parámetro **-R**, que permite cambiar en modo recursivo (a la vez) las propiedades de todos los archivos de un directorio:

**\$chown -R nomina direct1**

Esta orden da la propiedad de todos los archivos contenidos en el directorio *direct1* al usuario nomina.

Quando se transfiere la propiedad de un archivo a otro usuario, es conveniente copiar o mover el archivo al directorio HOME del nuevo usuario, para que así el nuevo propietario pueda tener los privilegios sobre ese archivo.


 <b>Comando</b>	Sintaxis	Opciones
<b>chgrp</b>	chgrp [opciones] archivo de grupos	[-R]

Tabla 13.36.

Es una orden muy similar a la anterior. La diferencia es que sirve para cambiar el grupo asociado a un archivo.

La sintaxis es igual que en el comando **chown**.

Supón el mismo archivo de antes y sus características:

**-rw-r--r-- 1 contable grupo 1000 Aug 12 12:00 doc1**

y la ejecución de la siguiente orden: **\$chgrp grupo1 doc1**.

El resultado es que el archivo *doc1* pasa a pertenecer al grupo1 y deja de ser de grupo.

También has de tener en cuenta que un usuario puede pertenecer a varios grupos. Lo que es evidente es que en una sesión de trabajo estará identificado ante uno de ellos.



# 13. Sistema operativo multiusuario Unix/Linux...

## 13.9 Permisos y derechos en Unix/Linux


 Comando	Sintaxis	Opciones
<code>newgrp</code>	<code>newgrp [-l] [grupo]</code>	<code>[-luv]</code>

Tabla 13.37.

Con esta orden lo que se puede hacer es cambiar de grupo de trabajo en una sesión de trabajo.

Sabes que cuando un usuario se identifica ante el sistema, éste tiene definidas unas características concretas: permisos de los que dispone, grupo al que pertenece, directorio **HOME**, etcétera.

En algunos casos puede resultar interesante o necesario que un usuario pertenezca temporalmente a otro grupo de trabajo.

Es más fácil, entonces, integrar al usuario en el nuevo grupo al que pertenece que crear otro usuario que pertenezca al nuevo grupo. Para ello se utiliza el comando **newgrp**.

Su uso es sencillo. Desde el símbolo del sistema se especifica el comando y el nuevo grupo al que quieres que pertenezca el usuario.

### \$newgrp grupo1

Desde entonces el usuario pertenece al **grupo1**.

Un usuario puede estar asignado a más de un grupo. Cuando se conecta al sistema (abre una sesión), figura como perteneciente a uno de ellos (tiene un *gid* que lo representa a la hora de otorgar derechos). Si el usuario desea cambiar su *gid* (el grupo en el que está actuando), utilizará el comando **newgrp**, y su *gid* cambiará.


 Comando	Sintaxis	Opciones
<code>su</code>	<code>su [opciones [-] [user [arg]]</code>	<code>[-lcfmps]</code>

Tabla 13.38.

Con esta orden puedes cambiar el identificador de usuario con el que no has conectado al sistema.

Supón que tienes un terminal conectado a un servidor Unix/Linux. En primer lugar, llega el usuario contable

y se identifica ante el sistema para realizar su sesión de trabajo:

**login:** contable

**password:** \*\*\*\*\*

Si el usuario contable ha llegado al final de su sesión de trabajo, y el terminal lo necesita el usuario nomina.

La operación se podría hacer de dos formas: saliendo del sistema con: **\$exit**, volviendo a realizar una nueva conexión o identificación con el sistema:

**login:** nomina

**password:** \*\*\*\*\*

o bien utilizando el comando **su**.

La diferencia es clara. Si el usuario contable tenía lanzado algún proceso, o alguien esta utilizando sus archivos, todo ello desaparecerá de memoria, se perderá el tiempo de conexión, la hora a la que se conecto, etc., ya que se ha realizado una desconexión física al poner **exit** en el símbolo del sistema.

Para ello se usa el comando **su**. Con é, simplemente, dentro de la misma sesión de trabajo, puedes cambiar de usuario sin necesidad de salir y volver a entrar en el sistema.

Supón que quieres cambiar del usuario contable al usuario nomina; la operación sería la siguiente: **\$su nomina**.

A partir de este momento, nomina será el nuevo usuario ante el sistema.

La orden **su**, sin ningún argumento, permite que te identifiquemos como superusuario del sistema, siempre y cuando conozcamos su clave de acceso.

Siempre que ejecutas el comando **su** estás iniciando una nueva sesión sin cerrar la anterior, es decir, es como si tuviésemos dos sesiones abiertas pero solamente una activa. Cada vez que ejecutas el comando **su**, puedes volver a la anterior sesión tecleando **exit**.

Con el comando **su** solamente cambias tu identificación ante el sistema, pero no el directorio de trabajo.

Esta orden solamente se puede utilizar con los grupos a los que pertenezca el usuario y que se definieron al crear el mismo.

## 13.10. Impresión de archivos

Todas las versiones de Unix/Linux contienen una serie de programas y utilidades para la impresión de archivos.

Estos programas se aglutinan en el denominado **sistema lp**, que dispone de la mayoría de las utilidades para lanzar trabajos de impresión, cancelarlos, controlar el spool de impresora, asignar prioridades de impresión, etcétera.

La diferencia en cuanto a la impresión de archivos, con respecto a DOS, radica en que, al disponer de un spool de impresión, una vez lanzado el trabajo a imprimir, el usuario tiene de nuevo control sobre el sistema. El trabajo pasa al spool, que es el que lo gestiona. Se pueden enviar varios trabajos seguidos, sin necesidad de esperar a que finalice la impresión de cada uno de ellos. Basta con esperar a que el spool los gestione.

Con el parámetro **-m** el spool de impresión lanza un aviso al usuario cuando el trabajo se ha finalizado con éxito.

Comando	Sintaxis	Opciones
lp	<ul style="list-style-type: none"><li>lp [-E] [-c] [-d destino] [-h servidor] [-m] [-n num-copias] [-o opción] [-q prioridad] [-s] [-t título] [-H manejo] [-P listadoDePáginas] [archivos]</li><li>lp [-E] [-c] [-h servidor] [-i idDeTarea] [-n num-copias] [-o opción] [-q prioridad] [-t título] [-H manejo] [-P listadoDePáginas] cancelar [-a] [-h servidor] [-u nombreDesuario] [id ] [destino] [destino]</li></ul>	[-EcdhimnoqstuHP]

Tabla 13.39.

Esta orden es la que se utiliza para imprimir archivos.

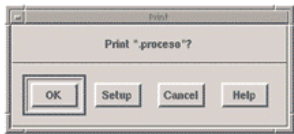


Fig. 13.15. Imprimir en Unix.

Es evidente que para poder utilizar la impresora, tienes que haber instalado previamente la misma y haber configurado el spool. Recuerda que el spool de impresoras se utiliza en la mayoría de los sistemas multiusuario.

Por defecto, cuando imprimimos un archivo: **\$lp doc1**, el sistema no pregunta en qué impresora quieres imprimir. Si tienes más de una conectada, el archivo se imprime en la **impresora por defecto**.

Lo que sí se puede hacer es imprimir por otra que no sea la que tienes instalada por defecto. Para ello es necesario utilizar el parámetro **-d** seguido del nombre de la impresora en la que quieres imprimir.

**\$lp -d Epson doc1**

De esta forma, el documento o archivo doc1 se imprime por una impresora que no es la predeterminada, ya que se imprime en la impresora que denominamos *Epson*.

Si, para un usuario concreto, quieres cambiar siempre la impresora por defecto, ya que, por ejemplo, necesitas imprimir en color, puedes atacar al fichero *.profile* indicando cuál será la impresora predeterminada para ese usuario. Se incluirá una orden similar a ésta:

**LPDEST=Epson export LPDEST**

Ya verás más adelante lo que significa esta línea.

**\$lp -m -d Epson doc1**

Has de tener en cuenta que cuando envías un trabajo a imprimir, el spool sabe quién envía el trabajo y qué archivo tiene que imprimir. Pero desde que envías el comando hasta que el trabajo se imprime pueden transcurrir minutos e incluso horas.

Si en este intervalo de tiempo el archivo que mandamos a imprimir es modificado, lo que se imprimirá será el nuevo archivo, y no el que nosotros en principio lanzamos.

Para ello existe el parámetro **-c**, que envía una copia del archivo al spool de impresión de tal forma que tarde lo que tarde en imprimirse el archivo, lo que se imprimirá será lo que nosotros queramos, y aunque modifiquemos posteriormente el archivo, las modificaciones no se imprimirán.

Esto tiene el inconveniente de cargar al sistema con más trabajo, usar mas espacio en disco, más memoria, etc. En definitiva, se pierde algo de rendimiento en el sistema, pero se gana en seguridad y fiabilidad.

- En entorno gráfico bastará con seleccionar el archivo a imprimir, abrirlo, y una vez abierto seleccionar la opción *File, Print*. Aparecerá la siguiente pantalla y confirmarás la impresión (véase la Figura 13.15).

Si haces clic en el botón *Setup*, podrás modificar las opciones de impresión.

Si quieres imprimir por la impresora determinada, será suficiente con pulsar el icono:

- En Linux, se hace de forma similar, y al seleccionar la opción de *Imprimir* se abrirá una ventana parecida a las que se muestran en Windows (véase la Figura 13.16).

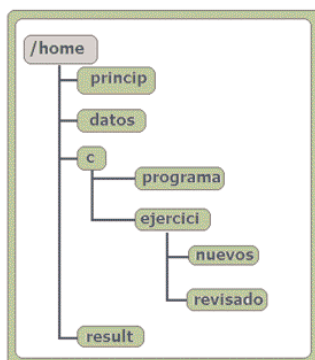
Para imprimir por la impresora predeterminada, pulsarás el icono



Fig. 13.16. Imprimir en Linux.

## Actividades

- 1 Ejecuta una sesión TELNET para conectarse al sistema.
- 2 Entra en el sistema con el nombre de usuario asignado por el administrador.
- 3 Averigua qué usuarios están identificados ante el sistema.
- 4 Envía un mensaje a uno de los usuarios.
- 5 Conversa con el usuario al que se le ha enviado el mensaje anterior.
- 6 Desactiva la opción de recibir mensajes. Vuelve a activarla.
- 7 Envía un mensaje a todos los usuarios identificados ante el sistema.
- 8 Cambia la clave de acceso al sistema del usuario con el que nos hemos conectado.
- 9 Comprueba que el cambio es correcto.
- 10 Identifica el sistema en el que estamos trabajando.
- 11 Realiza las siguientes cuestiones:
  - a) Sitúate en tu directorio personal de trabajo (**home**).
  - b) Crea la siguiente estructura de directorios.



- c) Crea los directorios princip, datos, c y result desde el directorio home.
- d) Cambia al directorio princip.
- e) Utilizando la trayectoria absoluta, crea los directorios programa y ejercici. Sigues situado en princip.
- f) Sitúate en el directorio ejercici utilizando la trayectoria relativa y crea los demás directorios; examina la estructura de directorios.

- g) Situado en el directorio programa, borra el directorio ejercici. Utiliza para todo este punto la trayectoria absoluta.
- h) Sitúate en el directorio home, y crea dos archivos: c.dat y c.bak.
- i) En el directorio home, crea el fichero **texto**, cuyo contenido sea la sintaxis del comando cp.
- j) En el directorio programa, crea tres ficheros: *texto.txt*, *texto.bak* y *texto1.bas*. El contenido de cada uno de ellos respectivamente, será el de la sintaxis de las órdenes **cd**, **md** y **rd**.
- k) Copia el contenido del directorio programa al directorio result, situado en el directorio datos, los archivos cuyo nombre empieza por tex y cuyos dos primeros caracteres de la extensión sea ba.
- l) Renombra en result el fichero texto1.bas por texto1.bak.
- m) Situado en el directorio home, cambia el nombre del directorio datos por el de datos.dat.
- n) Sitúate en el directorio home. Renombra el fichero texto.txt del directorio programa como texto.bas. Realiza esta operación con la orden correspondiente y, posteriormente, copia en el mismo directorio el fichero texto.bas con el nombre texto.nue, utilizando para ello el redireccionamiento y no la orden cp.
- o) Cambia en el directorio result la extensión de todos los archivos por la extensión dat.
- p) Visualiza por pantalla el contenido del fichero texto.dat del directorio result, pero paginado.
- q) Mueve el fichero texto.dat del directorio result al directorio datos.dat con el nombre texto1.dat.
- r) Borra todos los archivos del directorio programa cuya primera letra de la extensión sea b.
- s) Crea en result tres ficheros: fich1, fich2 y fich3. El contenido de estos ficheros puede ser cualquier cosa.
- t) Borra desde el directorio *programa* todo el directorio *result*.

- 12 Muestra por pantalla las últimas dos líneas de este fichero.
- 13 Ordena el fichero por marcas y visualízalo.
- 14 Ordena el fichero por modelos y visualízalo.
- 15 Busca en el fichero anterior aquellas líneas en el que el color sea BLANCO.
- 17 Asigna a este fichero todos los privilegios posibles. Utiliza la sintaxis numérica.
- 18 Elimina todos los privilegios de grupo y resto de usuarios.
- 19 Asigna todos los privilegios con la sintaxis no numérica.