

MANUAL PRÁCTICO
DE
LINUX
CON EJERCICIOS

ÍNDICE

Capítulo 1. Conceptos básicos sobre Linux.....	4
1.1 Qué es Linux.....	4
1.2 La importancia de trabajar en modo consola.....	5
1.3 Diferencias entre Linux y Unix.....	6
1.4 Diferencias entre Linux y Windows.....	6
1.5 Usar Linux sin instalar nada.....	6
1.6 Qué es una distribución de Linux.....	7
1.7 Cómo empezó todo. Breve historia de Linux.....	8
1.8 Tux, la mascota de Linux.....	10
Resumen del capítulo 1.....	10
Ejercicios del capítulo 1.....	11
Capítulo 2. Ficheros y directorios (parte I).....	12
2.1 Entrada al sistema (login).....	12
2.2 Estructura de directorios.....	13
2.3 contenido de un directorio. Cambio de directorio (ls, cd, pwd).....	14
2.4 Visualización de ficheros (cat, more, less, head, tail).....	17
2.5 Edición de ficheros (touch, vi, ee, mcedit).....	18
Resumen del capítulo 2.....	21
Ejercicios del capítulo 2.....	22
Capítulo 3. Ficheros y directorios (parte II).....	24
3.1 Caracteres comodín.....	24
3.2 Copia y borrado de ficheros (cp, mv, rm).....	25
3.3 Copia y borrado de directorios (cp, mv, rmdir).....	26
Resumen del capítulo 3.....	28
Ejercicios del capítulo 3.....	29
Capítulo 4. Grupos, usuarios y permisos.....	31
4.1 ¿Por qué existen grupos, usuarios y permisos?.....	31
4.2 ¿Qué es el superusuario?.....	32
4.3 Permisos.....	32
4.4 ¿Quiénes somos? (whoami, groups).....	33
4.5 Gestión de grupos (groupadd, groupdel, groupmod).....	34
4.6 Gestión de usuarios (adduser, userdel, usermod).....	34
4.7 Cambio de grupo y de dueño (chown, chgrp).....	35
4.8 Cambio de privilegios (chmod).....	36
Resumen del capítulo 4.....	38
Ejercicios del capítulo 4.....	39
Soluciones a los ejercicios.....	41
Capítulo 1.....	41
Capítulo 2.....	41
Capítulo 3.....	42
Capítulo 4.....	44
Apéndices	
GNU Free Documentation License.....	45
Licencia de Documentación Libre de GNU.....	50

¿A QUIÉN VA DIRIGIDO ESTE MANUAL?

Este manual es un texto PRÁCTICO dirigido a todos aquellos que quieren aprender a manejar Linux desde el terminal, utilizando comandos.

Está indicado para estudiantes de Informática de cualquier nivel: Ingenierías y Licenciaturas en carreras técnicas y/o científicas, Ciclos Formativos de Grado Superior, Ciclos Formativos de Grado Medio y Bachillerato. No obstante, puede ser de utilidad para cualquiera que simplemente tenga curiosidad o quiera aprender a realizar ciertas tareas con Linux más allá del entorno gráfico.

CAPÍTULO

1

CONCEPTOS BÁSICOS SOBRE LINUX

1.1 ¿QUÉ ES LINUX?

Para entender qué es Linux debemos saber antes qué es un **sistema operativo** (a veces se utilizan simplemente las siglas S.O.). Podemos dar una definición sencilla de este concepto:

“Un sistema operativo es un programa que permite al usuario interactuar con el ordenador y sus componentes (monitor, disco duro, impresora, etc) y que facilita la realización de tareas básicas como copiar o mover ficheros de un sitio a otro, editar archivos de texto, establecer una conexión a internet o hacer copias de seguridad”.

El sistema operativo es el primer programa que se ejecuta al encender el ordenador.

A un nivel superior tenemos los programas que permiten al usuario realizar tareas específicas. Estos programas se denominan **aplicaciones de usuario**, o simplemente aplicaciones. Podemos encontrar muchos ejemplos en el trabajo diario con el ordenador: programas de gestión contable como ContaPlus, procesadores de texto como OpenOffice.org Writer o MS Word, programas de retoque fotográfico como The Gimp o Photoshop.

GNU/Linux (comúnmente Linux a secas) es uno más de los muchos S.O. que existen en la actualidad que tiene una serie de características que lo hacen especial:

- **Libre:** Se puede descargar de internet, se puede copiar y distribuir sin que por ello se incurra en ningún tipo de delito. La licencia que establece los términos de uso, copia y distribución se denomina Licencia GNU (www.gnu.org)
- **Hecho por voluntarios:** Linux no se creó para obtener beneficios con él sino para satisfacer una serie de necesidades a la hora de trabajar con el ordenador. Hoy día sigue funcionando así. Cuando alguien necesita un determinado programa, simplemente lo crea y lo pone al servicio de la comunidad para que lo use y para que cada cual lo mejore y lo adapte a sus propias necesidades.
- **Multiusuario:** Varios usuarios pueden conectarse y usar el mismo ordenador a la vez.
- **Multitarea:** Pueden funcionar varios programas al mismo tiempo en la misma máquina.
- **Multiplataforma:** Hay versiones de Linux para gran cantidad de plataformas: todos los PCs basados en procesadores Intel o AMD, ordenadores Digital/Compaq con procesadores Alpha, ordenadores Apple, ultraportátiles como el Asus Eee e incluso dispositivos móviles como el Sharp Zaurus.
- **Estable:** Linux es un sistema operativo muy maduro, probado durante mucho tiempo. Hay muchos servidores que llevan funcionando bajo Linux de forma ininterumpida muchos años sin un solo cuelgue.
- **Eficiente:** Linux aprovecha bien los recursos hardware. Incluso los viejos Pentium pueden funcionar bien con Linux y servir para alguna tarea.
- **Hay miles de programas libres:** Hay una gran cantidad de programas, desde procesadores de texto hasta programas de dibujo pasando por todo tipo de servidores, totalmente libres y gratuitos que se pueden descargar e instalar desde el propio entorno de Linux.

1.2 LA IMPORTANCIA DE TRABAJAR EN MODO CONSOLA

Hay muchas formas de llamarlo, podemos hablar de “trabajar en un terminal”, “mediante líneas de comando”, “trabajar en la consola”, incluso hay quien prefiere hablar de “trabajar en modo texto”. Todas estas denominaciones se refieren a un modo de trabajo en el que para realizar una determinada tarea, se deben teclear comandos.

Por ejemplo, si desde una ventana de terminal tecleamos el comando **date**, se mostrará como resultado la fecha y la hora actual del sistema. Se podría haber averiguado la fecha haciendo clic con el ratón sobre el reloj del sistema, en la esquina superior derecha de la pantalla, pero claro, eso es válido en el caso de que el reloj del sistema esté en ese lugar y de que permita mostrar un calendario. En definitiva, en un terminal, escribiendo **date** se obtiene la hora y la fecha, ya se trate de un ordenador doméstico o un servidor de una agencia espacial independientemente de la versión de Linux utilizada. Sin embargo, realizar una tarea en un entorno gráfico puede ser muy diferente incluso en ordenadores similares con versiones similares de Linux ya que estos entornos son muy personalizables y a veces no aparecen los mismos menús ni están instalados los mismos programas.

1.3 DIFERENCIAS ENTRE LINUX Y UNIX?

A nivel de línea de comandos, apenas hay diferencias. Se pueden usar los mismos comandos indistintamente en distribuciones Linux o en sistemas Unix.

A nivel gráfico tampoco hay mucha diferencia ya que ambos sistemas utilizan X-Windows.

La principal diferencia es que Linux es libre mientras que la mayoría de Unix no (a veces cuestan mucho dinero).

Otra diferencia importante es el hecho de que hay versiones de Linux para prácticamente cualquier plataforma, mientras que los diferentes Unix suelen estar enfocados a un hardware muy específico.

1.4 DIFERENCIAS ENTRE LINUX Y WINDOWS

La principal diferencia, una vez más, es que Linux es libre y, en la mayoría de los casos, gratis mientras que Windows es un software propietario y cuesta dinero. De igual manera, la gran mayoría de aplicaciones para Linux son libres y gratuitas mientras que las aplicaciones para Windows no lo son, aunque a veces es fácil encontrar programas de evaluación (shareware) para Windows que permiten probar un determinado programa durante un tiempo de forma gratuita.

1.5 USAR LINUX SIN INSTALAR NADA

Se puede probar Linux aún teniendo otro sistema operativo instalado en el ordenador, sin instalar nada en el disco duro. De hecho, prácticamente todos los ejercicios de este libro se podrían hacer sin instalar Linux.

Se puede usar Linux sin tener que instalarlo mediante algunos de los siguientes métodos:

- **Live-CD:** Un live-CD es un disco que permite arrancar Linux y ejecutar programas desde el mismo disco. Casi todos los discos de instalación de Linux funcionan también como live-CD lo que da la oportunidad de probar el funcionamiento primero, e instalarlo en nuestro ordenador si nos gusta después.
- **Lápiz de memoria (*pen drive*):** Hay versiones de Linux especialmente ligeras en cuanto a requerimientos (necesitan poca RAM y poca CPU para funcionar) que se pueden arrancar desde un lápiz de memoria). El lector puede encontrar más información en <http://www.pendrivelinux.com/> así como instrucciones detalladas sobre la instalación y personalización.
- **Telnet:** Se trata de una aplicación que permite a un usuario conectarse a otro ordenador y trabajar en él. Si ese ordenador al que se conecta el usuario tiene Linux, entonces trabajará bajo Linux, aunque en su máquina local tenga otro sistema operativo.

1.6 ¿QUÉ ES UNA DISTRIBUCIÓN DE LINUX?

Una distribución de Linux es el sistema operativo propiamente dicho, lo que se suele denominar el núcleo (*kernel* en inglés) junto con un programa de instalación y un conjunto de aplicaciones, normalmente de propósito general.

Las primeras distribuciones eran difíciles de instalar pero actualmente apenas hay que introducir el CD e ir haciendo clic en “siguiente”.

A continuación se listan algunas de las distribuciones de Linux:

- **Ubuntu:** Está enfocada, sobre todo, a ordenadores de escritorio aunque también proporciona soporte para servidores. Está basada en Debian y sus principales características son la facilidad de uso e instalación. Se publica una versión cada 6 meses, una en abril y otra en octubre de cada año.
Ubuntu está patrocinado por Canonical, una empresa privada fundada y financiada por el empresario sudafricano Mark Shuttleworth.
El eslogan de Ubuntu es toda una declaración de intenciones: “Linux para seres humanos”. Resume una de sus metas principales: hacer de Linux un sistema operativo más accesible y fácil de usar.
- **openSUSE:** Se trata de una distribución auspiciada por las empresas Novell y AMD. El proyecto openSUSE tiene como objetivo hacer una distribución muy fácil de conseguir, tanto mediante descargas de internet como a través de puntos de venta físicos y, sobre todo, muy fácil de utilizar.
- **Mint:** Está basada en Ubuntu y su meta es ofrecer un sistema “listo para funcionar” y que incluya plugins para el navegador, codecs para ver video, programas para reproducir DVD, Java, etc. de tal forma que el usuario se ahorre instalar y configurar estos componentes. Linux Mint es compatible con el software de Ubuntu.
- **Fedora:** Proviene de otra distribución llamada Red Hat que incluía tanto software libre como software propietario. El objetivo del proyecto Fedora es construir un sistema operativo completo, de propósito general, basado exclusivamente en software libre.
- **Debian:** Fue Ian Murdock, en 1993, quién inició el proyecto Debian e inicialmente estaba patrocinado por la Free Software Foundation. Es quizás la distribución que mejor ha sabido mantener a lo largo del tiempo la filosofía del proyecto inicial de GNU/Linux.
Por su estabilidad y rendimiento, se utiliza con frecuencia en servidores cuya misión es crítica.
- **Mandriva:** Es la sucesora de la popular Mandrake y, como ella, incluye KDE como entorno gráfico. Hay todo un abanico de versiones de Mandriva: una totalmente gratuita, otra para arrancar desde un lápiz de memoria, otra con múltiples aplicaciones para servidores... La empresa del mismo nombre encargada de esta distribución ofrece también servicio técnico.
- **MEPIS:** La primera versión fue creada por Warren Woodford en el 2002 y está basada en Debian.
Tiene KDE como entorno de escritorio y está dirigida especialmente a los usuarios que utilizan el ordenador como estación de trabajo, para tareas ofimáticas y administrativas más que para su uso como servidor.
Tiene como punto fuerte un buen programa de instalación con posibilidad de reparticionar NTFS y recuperar el sistema operativo.
- **Sabayon:** Es una distribución basada en Gentoo, creada y mantenida por el italiano Fabio Erculiani.

Hace especial hincapié en el apartado multimedia, la aceleración gráfica y la posibilidad de ejecutar programas de Windows con el emulador Wine. Vienen incluidos en esta distribución algunos juegos 3D como Battle of Wesnoth, Nexuiz y Warsaw.

- **Slackware:** Creada por Patrick Volkerding, se trata de una de las distribuciones con más solera. Intenta mantener la tradición de ser una distribución fácil de usar y estable. Incluye servidores listos para usar en cuanto se instala el sistema: web, ftp y correo electrónico.

1.7 CÓMO COMENZÓ TODO. BREVE HISTORIA DE LINUX

La primera versión de Linux fue creada por un estudiante finés llamado Linus Torvalds.

Linux se matriculó en la Universidad de Helsinki en 1988 donde estudió Informática. Después de comprarse un PC, concretamente un 386, empezó a usar Minix, un sistema operativo creado por Andrew Tannenbaum para fines educativos. Linus no estaba demasiado contento con este sistema. Se lamentaba de la inestabilidad del emulador de terminal, que utilizaba para conectarse a los ordenadores de la universidad. Linus decidió hacer él mismo el programa emulador de terminal, independiente de Minix. Éstos fueron los primeros pasos que se dieron en la creación de Linux.

Linus terminó pronto su programa de emulación de terminal y pensó que estaría bien crear otros programas, por ejemplo para transferir ficheros de un sitio a otro.

En agosto de 1991, Linus mandó un correo electrónico, que ya es histórico, a USENET (una red de discusión parecida a los foros actuales), diciendo que estaba trabajando en este proyecto.

A continuación se muestra el mensaje original en inglés y su correspondiente traducción al castellano:

```
From: torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)
Newsgroups: comp.os.minix
Subject: What would you like to see most in minix?
Summary: small poll for my new operating system
Message-ID:
<1991Aug25.205708.9541@klaava.Helsinki.FI> Date: 25
Aug 91 20:57:08 GMT
Organization: University of Helsinki
```

I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things).

I've currently ported bash(1.08) and gcc(1.40), and things seem to work. This implies that I'll get something practical within a few months, and I'd like to know what features most people would want. Any suggestions are welcome, but I won't promise I'll implement them :-)

Linus (torvalds@kruuna.helsinki.fi)

PS. Yes - it's free of any minix code, and it has a multi-threaded fs. It is NOT portable (uses 386 task switching etc), and it probably never will support anything other than AT-harddisks, as that's all I have.

Traducción al castellano:

Hola a todos los que usan minix. Estoy haciendo un sistema operativo (gratis) para clónicos AT 386(486) (sólo como hobby, no será grande ni profesional como gnu). Se ha estado cocinando desde abril y parece que empieza a estar listo. Me gustaría recibir opiniones sobre lo que a la gente le gusta/disgusta de MINIX, ya que mi SO se parece a él en algunos aspectos (el mismo diseño físico del sistema de ficheros debido a razones prácticas). Actualmente, he incluido el bash(1.08) y el gcc(1.40), y la cosa parece que funciona. Esto quiere decir que tendré algo funcional en unos pocos meses, y me gustaría saber qué características querría la mayoría de la gente que tuviera. Cualquier sugerencia es bienvenida, pero no prometo que sea implementada :-). Linus (torvalds@kruuna.helsinki.fi)
PD: Sí - está libre de cualquier código de minix, y tiene un fs multi-hilo. NO es portable (usa el task switching del 386, etc.), y, probablemente, nunca soportará discos duros distintos a los AT, es todo lo que tengo :-).

Linus publicó la primera versión de Linux, la 0.01, en septiembre de 1991. El resto ya es historia...

1.8 TUX. LA MASCOTA DE LINUX

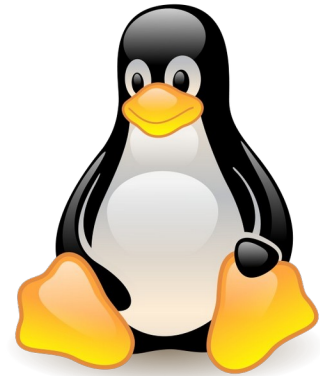
La mascota del sistema operativo Linux es un pingüino llamado Tux.

Hay diferentes versiones sobre el origen del término. La más aceptada es la que afirma que viene del término inglés “**tuxedo**”, que quiere decir esmoquin, y es lo primero que se le viene a la cabeza a mucha gente cuando ve a un pingüino.

Aunque hay quien dice que podría venir también de Torvalds **Unix**.

La mascota fue elegida por el propio Torvalds inspirándose en una foto que encontró en internet.

Tux es el protagonista de muchos de los juegos hechos para Linux como “Tux Racer”, “Tux on the Run”, “Super Tuxedo T. Penguin: A Quest for Herring”, “Chromium B.S.U.” o “Pingus”.



RESUMEN DEL CAPÍTULO 1

- Un **sistema operativo** es un programa que permite al usuario interactuar con el ordenador y sus componentes hardware y que facilita la realización de tareas básicas.
- Trabajar mediante comandos, en una ventana de terminal, permite realizar tareas de forma similar en cualquier versión de Linux o Unix.
- **Linux** es un sistema operativo que se caracteriza por ser libre y, en la mayoría de los casos también gratuito. Está hecho por voluntarios. Es multiusuario, multitarea y multiplataforma. Es muy estable y aprovecha bien los recursos de que dispone la máquina. La mayoría de los programas disponibles para Linux son también libres.
- La principal diferencia entre Linux y **Unix** radica en que Linux es libre y multiplataforma mientras que Unix suele ser comercial y muy orientado al hardware. **Windows** también es un sistema operativo comercial y las aplicaciones para este SO también suelen ser comerciales.
- Se puede usar Linux sin tener que instalar nada en el ordenador mediante alguno de estos métodos: live-CD, lápiz de memoria, telnet.
- Una **distribución** consta del sistema operativo propiamente dicho más el programa de instalación y una selección de aplicaciones. Algunas de las distribuciones más importantes son Ubuntu, openSUSE, Mint, Fedora, Debian y Mandriva.
- La primera versión de Linux fue creada por Linus Torvalds en 1991 con el fin de mejorar MINIX, un sistema operativo tipo UNIX utilizado en la universidad.
- La mascota de Linux es un pingüino al que se ha bautizado con el nombre de **Tux**.

EJERCICIOS DEL CAPÍTULO 1

En ocasiones, la respuesta a los ejercicios no se puede completar únicamente con el material teórico que se proporciona en este capítulo y el alumno debe, por tanto, buscar en otras fuentes complementarias como Internet.

Las soluciones a los ejercicios se encuentran al final del libro.

Los ejercicios están clasificados según su nivel de dificultad:

 **Facil.** El concepto viene explicado en el capítulo.

 **Dificultad media.** Es necesario relacionar conceptos y/o buscar información en Internet.

 **Difícil.** Hace falta una investigación concienzuda



-  1. Un sistema operativo es a) un programa que permite al usuario realizar tareas específicas, b) un procesador de textos, c) un programa que permite al usuario interactuar con el ordenador y sus componentes o d) ninguna de las respuestas anteriores es correcta.
-  2. Una distribución de Linux es a) el núcleo del SO, junto con un programa de instalación y una selección de aplicaciones, b) el núcleo del SO, junto con un entorno gráfico y una selección de aplicaciones o c) las dos respuestas anteriores son correctas.
-  3. Cita a) alguna distribución de Linux buena en el apartado gráfico y en juegos 3D y b) alguna distribución de Linux con servidores preinstalados.
-  4. a) ¿Cuáles son las diez distribuciones de Linux más populares del último mes? b) y del último año?
-  5. a) ¿Quién creó la primera versión de Linux?, b) ¿con qué objetivo?
-  6. Para ejecutar comandos de Linux en una máquina remota mediante telnet hace falta instalar antes Linux en la máquina local. a) Verdadero b) Falso.
-  7. a) ¿La primera versión de Linux se creó para un tipo de máquina concreta o podía funcionar con cualquier hardware? b) Sucede lo mismo hoy día.
-  8. La mascota de Linux se llama a) Tuxedo, b) Tux, c) Pingu o d) todas las anteriores son correctas.
-  9. ¿Quién fue el artista encargado de dibujar la primera versión de la mascota de Linux?
-  10. Citar tres distribuciones de Linux ligeras (con pocas necesidades en cuanto a recursos hardware) que se puedan instalar en un lápiz de memoria.

CAPÍTULO

2

FICHEROS

Y

DIRECTORIOS (PARTE I)

2.1 ENTRADA AL SISTEMA (LOGIN)

Para poder usar Linux, lo primero que hay que hacer es identificarse con un **nombre de usuario** y una **contraseña**.

El nombre de usuario no puede contener caracteres especiales como signos de puntuación (, ; :), la barra invertida (/), etc. La clave debe ser suficientemente larga y difícil de adivinar. No es buena idea utilizar como clave el nombre, apellidos, el número de teléfono, el número de la tarjeta de crédito o un nombre de mascota. Si la clave que utiliza un usuario es corta o fácil de adivinar corre el riesgo de que alguien entre en su sistema y borre o modifique información importante.

```
Ubuntu 8.04.2 ubuntu-desktop tty1
```

```
ubuntu-desktop login: luisjose
```

```
Password:
```

```
Linux ubuntu-desktop 2.6.24-23-generic #1 SMP Mon Jan 26 00:13:11 UTC 2009 i686
```

```
The programs included with the Ubuntu system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.
```

```
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by  
applicable law.
```

```
To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
Last login: Thu Mar  5 16:27:09 2009
luisjose@ubuntu-desktop:~$
```

La contraseña no aparece por pantalla mientras se teclea. Hay que tener cuidado con las mayúsculas y las minúsculas, si el sistema dice que la clave no es correcta puede que esté activada la tecla “BlqMayús”. Una vez introducidos el nombre de usuario y la clave, si el proceso de login se lleva a cabo correctamente, el sistema muestra el prompt con el formato:

```
nombre_de_usuario@nombre_de_la_máquina:~$
```

En este caso, el nombre de usuario es “luisjose”, el nombre de la máquina es “ubuntu-desktop” y aparece un carácter “\$” que indica que el usuario conectado es un usuario “normal”. Cuando un usuario tiene privilegios de root (super-usuario) aparece el carácter “#” como se verá más adelante.

¡Linux ya está listo para ejecutar comandos! El lector puede probar con el comando “date”, visto como ejemplo en el capítulo anterior.

```
luisjose@ubuntu-desktop:~$ date
Thu Mar  5 16:55:13 GMT 2009
luisjose@ubuntu-desktop:~$
```

2.2 ESTRUCTURA DE DIRECTORIOS.

Imagine el lector por un momento un montón de papeles amontonados en la mesa de una oficina: recibos del teléfono, facturas a clientes, notas tomadas en una reunión, factura de una reparación del coche. Buscar un documento entre todos estos papeles puede ser una pesadilla si están todos mezclados.

La solución a este desorden es muy fácil: utilizar **carpetas**.

Etiquetando carpetas y metiendo cada papel en su carpeta correspondiente, todo queda perfectamente ordenado. Puede haber una carpeta donde almacenar los pedidos a los proveedores, otra donde almacenar todas las notas tomadas en reuniones, otra para las facturas...

Es más, puede haber **subcarpetas** dentro de algunas carpetas. Por ejemplo, la carpeta etiquetada como “Facturas” puede contener, a su vez, subcarpetas etiquetadas como “Teléfono”, “Electricidad”, “Coche...”

En un ordenador, el almacenamiento de información se lleva a cabo de la misma manera. Trabajando en el entorno gráfico se habla de carpetas y trabajando con comandos en un terminal, se habla de **directorios**, pero conceptualmente son exactamente lo mismo.

A continuación se muestra una tabla con los directorios más importantes de un sistema Linux:

/ directorio raíz	/bin	Contiene programas ejecutables básicos para el sistema.
	/boot	Contiene los ficheros necesarios para el arranque del sistema.
	/dev	Contiene los ficheros correspondientes a los dispositivos: sonido, impresora, disco duro, lector de cd/dvd, video, etc.
	/etc	Contiene ficheros y directorios de configuración.
	/home	Contiene los directorios de trabajo de los usuarios. Cada usuario tiene su propio directorio en el sistema dentro de /home/.
	/lib	Contiene las librerías compartidas y los módulos del kernel
	/media	Dentro de este directorio se montan los dispositivos como el CD-ROM, memorias USB, discos duros portátiles, etc
	/opt	Directorio reservado para instalar aplicaciones.
	/sbin	Contiene los ficheros binarios ejecutables del sistema operativo.
	/srv	Contiene datos de los servicios proporcionado por el sistema.
	/tmp	Directorio de archivos temporales.
	/usr	Aquí se encuentran la mayoría de los archivos del sistema, aplicaciones, librerías, manuales, juegos... Es un espacio compartido por todos los usuarios.
	/var	Contiene archivos administrativos y datos que cambian con frecuencia: registro de errores, bases de datos, colas de impresión, etc.
	/root	Directorio de trabajo del administrador del sistema (usuario root).
	/proc	Aquí se almacenan datos del kernel e información sobre procesos.

2.3 VIZUALIZACIÓN, CREACIÓN Y CAMBIO DE DIRECTORIO (pwd, ls, cd, mkdir)

2.3.1 pwd

El comando **pwd** muestra cuál es el directorio de trabajo actual, en otras palabras, le dice al usuario dónde se encuentra dentro de la estructura de directorios del sistema. Es muy útil cuando estamos perdidos.

```
luisjose@ubuntu-desktop:~$ pwd
/home/luisjose
```

2.3.2 ls

El comando **ls** muestra el contenido del directorio actual. Por defecto, los archivos ocultos no se muestran. Éste es seguramente el comando que más se utiliza.

```
luisjose@ubuntu-desktop:~$ ls
Desktop Documents Examples Music Pictures Public Templates Videos
```

Se pueden añadir opciones a `ls`, por ejemplo

```
ls -a
```

muestra todos los archivos, incluyendo los ocultos (cuyo nombre comienza por un punto),

```
ls -l
```

muestra un listado detallado, con la última fecha de modificación de cada archivo, el tamaño, etc.,

```
ls -h
```

muestra el tamaño de los ficheros en bytes, Kb, Mb, etc.

Todas las opciones disponibles, tanto para `ls` como para el resto de comandos se pueden consultar mediante las páginas del manual, con el comando `man` seguido del comando del que se quiere obtener información:

```
luisjose@ubuntu-desktop:~$ man ls
```

Esto dará información detallada sobre el comando `ls`. Para salir del manual basta pulsar la letra “q”.

2.3.3 cd

El comando `cd` (change dir) permite cambiar de directorio. Si se utiliza tal cual, sin ningún tipo de argumento, cambia al directorio de trabajo personal. Si se utiliza seguido de una **ruta**, cambia al directorio que se indica.

```
luisjose@ubuntu-desktop:~$ pwd
/home/luisjose
luisjose@ubuntu-desktop:~$ cd /etc
luisjose@ubuntu-desktop:/etc$ pwd
/etc
```

En este caso, el usuario estaba en su directorio de trabajo, y ha “saltado” al directorio `/etc`. El lector puede teclear el comando `ls` si tiene curiosidad por ver lo que hay dentro.

Las rutas pueden ser **absolutas** o **relativas**. Una ruta es absoluta cuando comienza por el carácter “/” y relativa cuando comienza por cualquier otro carácter.

En el ejemplo anterior se ha usado una ruta absoluta, esto es, `/etc`. Veamos cómo cambiar a otros directorios utilizando otras rutas absolutas:

```
luisjose@ubuntu-desktop:/$ cd /usr/local/
luisjose@ubuntu-desktop:/usr/local$ ls
bin etc games include lib man sbin share src
luisjose@ubuntu-desktop:/usr/local$ cd /var/spool/
luisjose@ubuntu-desktop:/var/spool$ ls
anacron cron cups cups-pdf mail openoffice
```

Una ruta relativa es algo así como una ruta parcial. La ruta que se aplica es la concatenación de la ruta actual y de la ruta relativa. Veamos un ejemplo:

```
luisjose@ubuntu-desktop:/var/spool$ cd
luisjose@ubuntu-desktop:~$ pwd
/home/luisjose
luisjose@ubuntu-desktop:~$ cd Music
luisjose@ubuntu-desktop:~/Music$ pwd
/home/luisjose/Music
```

Recordemos que el comando `cd` sin argumentos, nos lleva al directorio de trabajo personal.

En este caso,
`cd Music`
 sería equivalente a
`cd /home/luisjose/Music`
 ya que se suma la ruta actual (`/home/luisjose`) a la ruta relativa indicada (`Music`)

Las rutas, tanto las absolutas como las relativas se pueden utilizar en la mayoría de comandos. No son algo específico que se utilice sólo con `cd`.

Podemos, por ejemplo, utilizar rutas como argumentos del comando `ls`.

```
luisjose@ubuntu-desktop:~/Music$ ls /boot/grub/
default          installed-version  minix_stage1_5    xfs_stage1_5
device.map       jfs_stage1_5      reiserfs_stage1_5
e2fs_stage1_5    menu.lst          stage1
fat_stage1_5     menu.lst~         stage2
```

Dos puntos (`..`) hacen referencia al directorio que hay justo a un nivel superior.

```
luisjose@ubuntu-desktop:~/Music$ ls ..
Desktop Documents Examples Music Pictures Public Templates Videos
```

`ls ..` muestra el contenido del directorio `/home/luisjose` que es el directorio que hay justo a un nivel superior de `/home/luisjose/Music`

```
luisjose@ubuntu-desktop:~/Music$ cd ..
luisjose@ubuntu-desktop:~$ pwd
/home/luisjose
```

`cd ..` sube un nivel en la estructura de directorios

2.3.4 mkdir

Se pueden crear directorios con el comando `mkdir`. Por ejemplo, para crear una estructura de carpetas donde un estudiante guardará información sobre sus asignaturas según el siguiente esquema:

/home/luisjose	/Documentos			
	/Escritorio			
	/Imágenes			
	/Música			
	/matematicas	/curso_01	/algebra	/exámenes_antiguos /apuntes
			/analisis	
			/fisica	/libros_de_ejercicios /videos
			/informatica	/compiladores_pascal
	/Video			

tendría que hacer lo siguiente:

```
~$ mkdir matematicas
~$ cd matematicas/
~/matematicas$ mkdir curso_01
~/matematicas$ cd curso_01/
~/matematicas/curso_01$ mkdir algebra analisis fisica informatica
~/matematicas/curso_01$ ls
algebra analisis fisica informatica
~/matematicas/curso_01$ cd algebra/
~/matematicas/curso_01/algebra$ mkdir examenes_antiguos apuntes
~/matematicas/curso_01/algebra$ cd ..
~/matematicas/curso_01$ cd fisica
~/matematicas/curso_01/fisica$ mkdir libros_de_ejercicios
~/matematicas/curso_01/fisica$ mkdir videos
~/matematicas/curso_01/fisica$ cd ..
~/matematicas/curso_01$ cd informatica/
~/matematicas/curso_01/informatica$ mkdir compiladores_pascal
```

Nótese que ya no se muestra en el ejemplo el prompt completo, con el nombre de usuario y el nombre de la máquina. Se seguirá en el libro esta norma a partir de ahora.

2.4 VISUALIZACIÓN DE FICHEROS (cat, more, less, head, tail)

Los comandos `cat`, `more` y `less` sirven para mostrar el contenido de ficheros de texto. La diferencia radica en cómo se muestra el contenido. A todos estos comandos hay que pasarles como argumento el fichero que se quiere mostrar. Se puede indicar una ruta, en caso de que el fichero que se quiere mostrar no esté en el directorio actual.

El comando `cat` muestra por pantalla el contenido de un fichero y, cuando termina, el usuario está otra vez de vuelta en la línea de comandos.

Por ejemplo,

```
~$ cat /var/log/dmesg
```

muestra el contenido del fichero `dmesg` que está dentro del directorio `/var/log`. Si el lector ha probado a hacerlo él mismo, se habrá dado cuenta de que es imposible ver todo el contenido de este fichero, porque ha pasado por pantalla muy rápido. Por eso `cat` se suele utilizar para visualizar el contenido de archivos pequeños.

El comando `more` hace lo mismo que `cat`, a diferencia de que muestra el fichero pantalla a pantalla, es decir, llena de texto la pantalla y se espera a que el usuario pulse la tecla <espacio> para pasar a la siguiente:

```
~$ more /var/log/dmesg
```

El comando `less` es el más versátil de los tres, ya que permite moverse hacia delante y hacia atrás dentro del fichero, utilizando los cursores o las teclas de “AvPág” y “RePág”:

```
:~$ less /var/log/dmesg
```

En cualquier momento se puede interrumpir la visualización y volver al símbolo del sistema pulsando la letra “q”.

Los comandos `head` y `tail` permiten mostrar de forma parcial el contenido de un fichero. Como su nombre indica, `head` muestra las primeras líneas del fichero (la cabecera) y `tail` muestra las últimas líneas (la cola).

Veamos algunos ejemplos:

```
~$ head /boot/grub/menu.lst
# menu.lst - See: grub(8), info grub, update-grub(8)
#             grub-install(8), grub-floppy(8),
#             grub-md5-crypt, /usr/share/doc/grub
#             and /usr/share/doc/grub-doc/.

## default num
# Set the default entry to the entry number NUM. Numbering starts from 0, and
# the entry number 0 is the default if the command is not used.
#
# You can specify 'saved' instead of a number. In this case, the default entry

~$ tail /boot/grub/menu.lst
root          (hd0,0)
kernel        /boot/vmlinuz-2.6.24-19-generic   root=UUID=409e68a1-6123-476f-abf7-
042854b68f3c ro single
initrd        /boot/initrd.img-2.6.24-19-generic

title         Ubuntu 8.04.2, memtest86+
root          (hd0,0)
kernel        /boot/memtest86+.bin
quiet

### END DEBIAN AUTOMAGIC KERNELS LIST
```

Por defecto, tanto `head` como `tail` muestran 10 líneas, pero eso se puede cambiar con la opción `-n`.

```
~$ tail -n4 /boot/grub/menu.lst
kernel        /boot/memtest86+.bin
quiet

### END DEBIAN AUTOMAGIC KERNELS LIST
```

En este caso se han mostrado solamente 4 líneas.

2.5 EDICIÓN DE FICHEROS (`touch`, `vi`, `ee`, `mcedit`)

El comando `touch` permite crear un fichero vacío. Con cualquier editor de texto se puede crear un fichero vacío pero con `touch` es especialmente cómodo y rápido.

```
~$ ls
Desktop Documents Examples Music Pictures Public Templates Videos
~$ touch prueba.txt
~$ ls
Desktop Examples Pictures Public Videos
Documents Music prueba.txt Templates
~$ cat prueba.txt
~$
```

Se puede ver en el ejemplo cómo se ha creado el archivo `prueba.txt` pero al visualizar su contenido con `cat`, no aparece nada en pantalla, por tanto está vacío.

El programa `ee` es un editor muy rudimentario pero al mismo tiempo efectivo. Podemos editar el archivo anterior y escribir alguna frase:

```
~$ ee prueba.txt
```

Presionando la tecla `ESC`, el usuario puede salir al menú principal y guardar el fichero. Podemos comprobar ahora cuál es el contenido del fichero:

```
~$ cat prueba.txt
Hola, aquí estoy aprendiendo Linux.
```

Otro editor muy simple es `nano`. Se deja al lector curioso probar su funcionamiento y compararlo con `ee`. En el hipotético caso de no estar instalado alguno de estos editores, su instalación es muy sencilla, basta con teclear `sudo apt-get install` seguido del nombre del programa que queremos instalar. Por ejemplo, si queremos instalar `ee`:

```
~$ sudo apt-get install ee
```

El programa `mcedit` es un editor algo más sofisticado que `ee` o `nano` (al menos en apariencia) y es una parte de `mc` (Midnight Commander), un programa muy al estilo del famoso Norton Commander de MS- DOS . Vamos a modificar el archivo `prueba.txt` creado anteriormente. Antes de eso, instalaremos `mc`, ya que no está instalado por defecto:

```
~$ sudo apt-get install mc
~$ mcedit prueba.txt
```

Con la tecla `F2` guardamos los cambios y con dos pulsaciones de `ESC` (o con la tecla `F10`) salimos del programa.

Comprobamos ahora que todo se ha grabado bien:

```
~$ cat prueba.txt
Hola, aquí estoy aprendiendo Linux.
Me encanta, se pueden hacer muchas cosas.
```

Hemos dejado para el final al editor de Linux por excelencia, se trata de `vi`. A primera vista es el más difícil de utilizar, lo cual es cierto, y parece que tiene menos opciones, pero muy al contrario se trata de un programa muy potente. Cualquier “linuxero” que se precie debe saber manejar bien este programa. Añadiremos una línea más al fichero `prueba.txt`. Para ello, seguiremos los siguientes pasos:

```
~$ vi prueba.txt
```

- Pulsar la letra “i” para entrar en modo “edición”.
- Escribiremos el texto.
- Pulsar la tecla `ESC` para salir del modo “edición”.
- Teclear “:” + “w” + `INTRO` para grabar los cambios.
- Teclear “:” + “q” + `INTRO` para salir del programa.

Comprobamos una vez más que todo está bien grabado:

```
~$ cat prueba.txt
Hola, aquí estoy aprendiendo Linux.
Me encanta, se pueden hacer muchas cosas.
¡Pronto dominaré el editor Vi!
```

Es más que recomendable realizar el tutorial llamado `vimtutor`.

RESUMEN DEL CAPÍTULO 2

- Todo usuario necesita un **nombre** y una **contraseña** para entrar en el sistema.
- La información se almacena físicamente en **directorios** y **subdirectorios** (carpetas y subcarpetas).
- Hay una serie de directorios predefinidos como `/bin`, `/dev`, `/home`, `/etc`, `/var`, etc. para todos los sistemas Linux.
- Hay **rutas absolutas**, que comienzan por el carácter `/`, y que definen una ruta efectiva completa y **rutas relativas**, que no comienzan por el carácter `/`, y cuya ruta efectiva sería la concatenación del directorio actual con esa misma ruta relativa.
- Los comandos vistos en este capítulo son los siguientes:

Comando	Acción	Ejemplo
pwd	muestra el directorio actual	<code>pwd</code>
ls	lista ficheros y directorios	<code>ls -l</code>
cd	cambia de directorio	<code>cd mp3/wim_mertens</code>
mkdir	crea uno o varios directorios	<code>mkdir cartas facturas</code>
cat	visualiza un fichero	<code>cat /var/log/dmesg</code>
more	visualiza un fichero pantalla a pantalla	<code>more /var/log/dmesg</code>
less	visualiza un fichero pantalla a pantalla y permite retroceder	<code>less /var/log/dmesg</code>
head	visualiza las primeras filas de un fichero	<code>head -n5 /var/log/dmesg</code>
tail	visualiza las últimas filas de un fichero	<code>tail /var/log/dmesg</code>
touch	crea un fichero vacío	<code>touch listado.txt</code>
ee	editor de textos muy simple	<code>ee listado.txt</code>
mcedit	editor de textos que forma parte de Midnight Commander	<code>mcedit listado.txt</code>
vi	editor de textos muy potente	<code>vi listado.txt</code>
apt-get	instala y desinstala programas	<code>apt-get install mc</code>
man	muestra ayuda sobre un determinado comando	<code>man ls</code>

EJERCICIOS DEL CAPÍTULO 2

En ocasiones, la respuesta a los ejercicios no se puede completar únicamente con el material teórico que se proporciona en este capítulo y el alumno debe, por tanto, buscar en otras fuentes complementarias. En los ejercicios de este capítulo se recomienda consultar las páginas *man*.

Las soluciones a los ejercicios se encuentran al final del libro.

Los ejercicios están clasificados según su nivel de dificultad:

 **Fácil.** El concepto viene explicado en el capítulo.

 **Dificultad media.** Es necesario relacionar conceptos y/o buscar información en Internet.

 **Difícil.** Hace falta una investigación concienzuda



1.  ¿En qué directorio se encuentran los ficheros de configuración del sistema?
2.  Para entrar en un sistema Linux hace falta a) nombre de usuario, contraseña y dirección IP, b) nombre de usuario y contraseña o c) únicamente una contraseña..
3.  Muestra el contenido del directorio actual.
4.  Muestra el contenido del directorio que está justo a un nivel superior.
5.   ¿En qué día de la semana naciste?, utiliza la instrucción `cal` para averiguarlo.
6.  Muestra los archivos del directorio `/bin`
7.  Suponiendo que te encuentras en tu directorio personal (`/home/nombre`), muestra un listado del contenido de `/usr/bin` a) con una sola línea de comando, b) moviéndote paso a paso por los directorios y c) con dos líneas de comandos.
8.   Muestra todos los archivos que hay en `/etc` y todos los que hay dentro de cada subdirectorio, de forma recursiva (con un solo comando).
9.   Muestra todos los archivos del directorio `/usr/X11R6/bin` ordenados por tamaño (de mayor a menor). Sólo debe aparecer el nombre de cada fichero, sin ninguna otra información adicional.
10.   Muestra todos los archivos del directorio `/etc` ordenados por tamaño (de mayor a menor) junto con el resto de características, es decir, permisos, tamaño, fechas de la última modificación, etc. El tamaño de cada fichero debe aparecer en un formato “legible”, o sea, expresado en Kb, Mb, etc.
11.    Muestra todos los archivos del directorio `/bin` ordenados por tamaño (de menor a mayor). Sólo debe aparecer el tamaño y el nombre de cada fichero, sin ninguna otra información adicional. El tamaño de cada fichero debe aparecer en un formato “legible”, o sea, expresado en Kb, Mb, etc.
12.  Muestra el contenido del directorio raíz utilizando como argumento de `ls` una ruta absoluta.

13. 🐧🐧 Muestra el contenido del directorio raíz utilizando como argumento de `ls` una ruta relativa. Suponemos que el directorio actual es `/home/elena/documentos`.

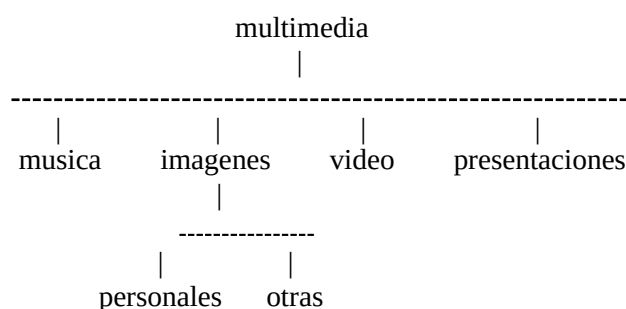
14. 🐧 Crea el directorio `gastos` dentro del directorio personal.

15. 🐧 ¿Qué sucede si se intenta crear un directorio dentro de `/etc`?

16. 🐧 Muestra el contenido del fichero `/etc/fstab`

17. 🐧 Muestra las 10 primeras líneas del fichero `/etc/bash.bashrc`

18. 🐧 Crea la siguiente estructura de directorios dentro del directorio de trabajo personal:



19. 🐧 Crea un fichero vacío dentro del directorio `musica`, con nombre `estilos_favoritos.txt`

20. 🐧 Utiliza tu editor preferido para abrir el fichero `estilos_favoritos.txt` e introduce los estilos de música que más te gusten. Guarda los cambios y sal.

21. 🐧 Muestra todo el contenido de `estilos_favoritos.txt`

22. 🐧 Muestra las 3 primeras líneas de `estilos_favoritos.txt`

23. 🐧 Muestra la última línea de `estilos_favoritos.txt`

24. 🐧🐧🐧 Muestra todo el contenido del fichero `estilos_favoritos.txt` excepto la primera línea. Se supone que no sabemos de antemano el número de líneas del fichero.

CAPÍTULO 3

FICHEROS

Y DIRECTORIOS (PARTE II)

3.1 CARACTERES COMODÍN

En muchas ocasiones es necesario realizar acciones sobre muchos archivos o directorios al mismo tiempo. Por ejemplo:

```
$ cat docu1 docu2 docu3 docu4 docu5 docu6
```

Se pueden crear patrones usando **símbolos comodín** para no tener que escribir todos y cada uno de los ficheros.

Para mostrar cada uno de los ficheros que comienzan por docu seguido de un número del uno al seis se puede utilizar un patrón:

```
$ cat fich[1-6]
```

Si se quiere mostrar simplemente el contenido de todos los ficheros que comienzan por fich se puede hacer:

```
$ cat fich*
```

donde el carácter “*” representa cualquier combinación de caracteres, incluso la cadena vacía. Si existe un fichero con nombre `fich` a secas en el directorio actual, también se mostrará.

El carácter “*” se puede colocar en cualquier lugar. Por ejemplo, para mostrar todos los ficheros que empiezan por la letra `a` y terminan por la letra `s` dentro del directorio `/usr/bin`:

```
$ ls /usr/bin/a*s
```

El símbolo “?” representa un carácter cualquiera. De esta forma, la siguiente sentencia muestra todos los ficheros del directorio `/usr/bin` cuyo nombre comienza por `g`, sigue cualquier carácter, a continuación sigue una `o` y termina con cualquier cadena de caracteres incluida la cadena vacía:

```
$ ls /usr/bin/g?o*
```

Ya se ha visto al principio del capítulo un ejemplo del uso de los corchetes. Los corchetes se utilizan de una forma parecida al carácter “?” aunque, a diferencia de éste, permiten especificar un poco más. Por ejemplo `[adfg]` significa cualquiera de los caracteres `a`, `d`, `f` o `g`. `[Hh]ola` es un patrón que encaja tanto con `Hola` como con `hola`. `[a-z]*` representa cualquier cadena de caracteres que comienza con una letra minúsculas.

3.2 COPIA Y BORRADO DE FICHEROS (**cp**, **mv**, **rm**)

3.2.1 **cp**

El comando **cp** sirve para copiar ficheros. Se puede copiar un único fichero o muchos. Se pueden copiar tanto ficheros como directorios. Por supuesto, se pueden utilizar los símbolos comodín.

En el proceso de copia intervienen tres factores: lo que se copia, la ruta de origen y la ruta de destino. No está de más recordar que las rutas pueden ser tanto absolutas como relativas. La ruta de origen se especifica junto con lo que se quiere copiar. Veamos un ejemplo:

```
$ cp /etc/hosts /home/alumno/pruebas/
```

La sentencia anterior copia el fichero `hosts`, que se encuentra en el directorio `/etc` al directorio `/home/alumno/pruebas/`.

Si no se especifica ningún directorio origen, se toma por defecto el directorio actual. Por ejemplo:

```
$ cp *.odt textos/
```

copia todos los archivos con la extensión `odt` del directorio actual al directorio `textos`.

Cuando se quiere especificar como directorio destino el directorio actual se utiliza el carácter “.”

Por ejemplo:

```
$ cp /usr/bin/g* .
```

copia todos los ficheros del directorio `/usr/bin` que comienzan por la letra `g` al directorio actual.

3.2.2 **mv**

El comando **mv** sirve para dos cosas, para mover y para cambiar de nombre. Se puede hacer cualquiera de las dos cosas por separado o las dos cosas al mismo tiempo. Por ejemplo:

```
$ mv mi_texto.txt carta.txt
```

le cambia el nombre a `mi_texto.txt` y pasa a llamarse `carta.txt`.

En cambio

```
$ mv carta.txt Documentos/
```

mueve `carta.txt` al directorio `Documentos`.

Se pueden hacer las dos cosas a la vez, mover y cambiar el nombre:

```
~$ cd Documentos/  
~/Documentos$ mkdir correspondencia  
~/Documentos$ mv carta.txt correspondencia/carta01.txt
```

En este caso, el fichero `carta.txt` se ha movido al directorio `~/Documentos/correspondencia` y además se le ha cambiado el nombre a `carta01.txt`

3.2.3 rm

El comando **rm** se utiliza para borrar ficheros. Es importante destacar que estos ficheros no se envían a una papelera así que NO SE PUEDEN RECUPERAR UNA VEZ BORRADOS.

Ejemplo:

```
$ rm *.txt
```

Esta sentencia borra todos los archivos con la extensión `txt` del directorio actual.

3.3 COPIA Y BORRADO DE DIRECTORIOS (**cp**, **mv**, **rm**)

De la misma manera que se copian, se borran o se mueven ficheros, se puede hacer lo mismo con los directorios. Hay que tener en cuenta que un directorio puede contener muchos ficheros y, además, otros directorios que, a su vez, pueden contener más ficheros y directorios. Por tanto, si se quiere copiar un fichero completo, con todo lo que tiene dentro, hay que indicarlo con la opción `-R`. A esto último se suele llamar “copiar de forma recursiva”.

Ejemplo:

```
~$ mkdir multimedia2  
~$ cp multimedia/* multimedia2  
cp: se omite el directorio «multimedia/imagenes»  
cp: se omite el directorio «multimedia/musica»  
cp: se omite el directorio «multimedia/presentaciones»  
cp: se omite el directorio «multimedia/video»  
~$ ls multimedia2  
~$
```

Se ha hecho una copia del contenido del directorio `multimedia` al directorio `multimedia2` pero no se ha copiado ningún archivo ¿qué ha pasado? Sencillamente no se ha hecho una copia recursiva (con la opción `-R`). Se ha intentado copiar únicamente justo dentro del directorio `multimedia` pero no a un nivel inferior. Como a ese nivel no había ningún fichero, no se ha copiado nada.

Vamos a intentarlo ahora de forma recursiva:

```
~$ cp -R multimedia/* multimedia2
~$ ls -R multimedia2
multimedia2:
imagenes musica presentaciones video

multimedia2/imagenes:
otras personales

multimedia2/imagenes/otras:

multimedia2/imagenes/personales:

multimedia2/musica:
estilos_favoritos.txt

multimedia2/presentaciones:

multimedia2/video:
```

Como se puede comprobar, se han copiado tanto la estructura de directorios como los contenidos de cada uno de ellos.

El comando `mv` funciona de forma análoga a `cp`, pero mueve en lugar de copiar. Cuando se trata de renombrar, funciona exactamente igual que con los ficheros.

Ejemplo:

```
~$ mv multimedia2 multimedia_copia
```

Esto le cambia el nombre al directorio `multimedia2` y pasa a llamarse `multimedia_copia`. El lector puede comprobar que el contenido de ese directorio permanece intacto.

Con `rm` se pueden borrar directorios.

```
~$ rm multimedia_copia/
rm: no se puede borrar «multimedia_copia/»: Es un directorio
```

Se obtiene un error, ¿qué sucede? Si el lector es perspicaz, sabrá cómo solucionar este problema...
...En efecto, hay que borrar el contenido de forma recursiva:

```
~$ rm -Rf multimedia_copia/
```

Además de la opción `-R`, se ha incluido la opción `-f` que hace que no se nos pida confirmación por cada elemento que se quiere borrar.

RESUMEN DEL CAPÍTULO 3

- Utilización de los símbolos comodín:

<i>Ejemplos</i>	<i>Significado</i>
*	Cualquier cadena de caracteres.
f	Cadena de caracteres que contienen una f.
z*	Cadena de caracteres que empieza por z y le sigue cualquier cosa.
a?	Una cadena formada por dos caracteres, el primero una a y el segundo, cualquier carácter.
[Dd]ocumento	Puede ser Documento o documento.
A[a-z][0-6]	Una cadena formada por la A mayúscula seguida de cualquier letra minúscula, seguida a su vez de un dígito del 0 al 6.

- Los comandos vistos en este capítulo son los siguientes:

<i>Comando</i>	<i>Acción</i>	<i>Ejemplo</i>
cp	copia archivos o directorios	cp *.txt correspondencia/
mv	mueve o renombra archivos o directorios	mv palabras.txt texto.txt
rm	borra archivos o directorios	rm -R cosas/basurilla
rmdir	borra directorios	rmdir viejo

EJERCICIOS DEL CAPÍTULO 3

En ocasiones, la respuesta a los ejercicios no se puede completar únicamente con el material teórico que se proporciona en este capítulo y el alumno debe, por tanto, buscar en otras fuentes complementarias. En los ejercicios de este capítulo se recomienda consultar las páginas [man](#).

Las soluciones a los ejercicios se encuentran al final del libro.

Los ejercicios están clasificados según su nivel de dificultad:

 **Facil.** El concepto viene explicado en el capítulo.

 **Dificultad media.** Es necesario relacionar conceptos y/o buscar información en Internet.

 **Difícil.** Hace falta una investigación concienzuda



1.  Muestra todos los archivos del directorio actual que son imágenes `jpg`.
2.  Muestra todos los archivos del directorio `/usr/bin` que empiecen por la letra `j`.
3.  Muestra los archivos que empiecen por `k` y tengan una `a` en la tercera posición, dentro del directorio `/usr/bin`.
4.  Muestra los archivos del directorio `/bin` que terminen en `n`.
5.  Muestra todos los archivos que hay en `/etc` y todos los que hay dentro de cada subdirectorio, de forma recursiva.
6.  Crea un directorio en tu directorio de trabajo con nombre `prueba`. Copia el archivo `gzip` del directorio `/bin` al directorio `prueba`. Crea un duplicado de `gzip` con nombre `gzip2` dentro de `prueba`.
7.  Cambia el nombre de `prueba` a `prueba2`. Crea `prueba3` en el mismo nivel que `prueba2` y mueve todos los ficheros de `prueba2` a `prueba3`. Borra `prueba2`.
8.  Crea un fichero vacío con nombre `“*?Hola caracola?*”`. ¿Se puede? En caso de que se pudiera, ¿sería recomendable poner nombres así? Razona la respuesta.
9.  Crea un directorio con nombre `multimedia_pruebas` y copia en él todo el contenido del directorio `multimedia`. A continuación crea en `multimedia/video/` dos ficheros, uno con nombre `peliculas.txt` y otro con nombre `actores.txt`. Edita el fichero `peliculas.txt` e introduce el nombre de tu película favorita. A continuación, crea en `multimedia_pruebas/video/` otro fichero que también tenga por nombre `peliculas.txt`, edítalo y esta vez escribe el nombre de tus cinco películas favoritas. Ahora haz una copia de todo el contenido de `multimedia` en `multimedia_prueba` de tal forma que sólo se copien los contenidos nuevos, es decir, si hay coincidencia en el nombre de un archivo se respetará el que se haya modificado más recientemente. Para comprobar que se ha hecho todo correctamente, basta mirar si en `multimedia_prueba/video` está el archivo vacío `actores.txt` y además el archivo `peliculas.txt` debe contener 5 películas y no 1.

10.  Borra el directorio multimedia/imagenes/otras. El sistema debe pedir al usuario que confirme el borrado.
11.  Mueve el archivo peliculas.txt, que está dentro de multimedia/video, al directorio que está justo a un nivel superior. Ahora el archivo debe llamarse mis_peliculas.txt en lugar de películas.

CAPÍTULO

4

GRUPOS, USUARIOS Y PERMISOS

4.1 ¿POR QUÉ EXISTEN GRUPOS, USUARIOS Y PERMISOS?

Vimos en un capítulo anterior que los ficheros deben estar organizados en directorios (carpetas) con el fin de tenerlos ordenados y poder localizarlos convenientemente.

Volvamos a nuestro ejemplo de la oficina. Cada papel está en su sitio, hay carpetas y subcarpetas y todo está organizado. Ahora bien, el contable deberá tener acceso por ejemplo a las carpetas donde se encuentran las facturas y los recibos pero no tienen por qué tener acceso a la información sobre desarrollo de productos o marketing. En un sistema Linux, las carpetas y los archivos funcionan de esta manera. Por ejemplo, los archivos de configuración que se encuentran en el directorio `/etc` sólo pueden ser modificados por el administrador del sistema. Esto previene que cualquier usuario pueda cambiar información crítica y estropear algo.

4.2 ¿QUÉ ES EL SUPERUSUARIO?

El superusuario, administrador del sistema o simplemente el `root`, es un usuario especial que tiene privilegios para cambiar la configuración, borrar y crear ficheros en cualquier directorio, crear nuevos grupos y usuarios, etc.

IMPORTANTE: ES PELIGROSO TRABAJAR COMO SUPERUSUARIO, SE PUEDE DANAR EL SISTEMA DE FORMA IRREVERSIBLE. EL LECTOR DEBE ESTAR

```
$ touch /etc/prueba.txt
touch: no se puede efectuar `touch' sobre «/etc/prueba.txt»: Permiso denegado
$ sudo touch /etc/prueba.txt
$ ls /etc/pru*
/etc/prueba.txt
```

Hemos intentado primero crear el fichero `prueba.txt` en el directorio `/etc` como usuario normal y acto seguido hemos obtenido un error de “Permiso denegado”, lo que quiere decir que un usuario sin privilegios no puede hacer eso. A continuación lo hemos intentado como administrador, para ello hemos usado el comando `SUDO`, tras lo que se nos ha preguntado la clave del administrador. Esta vez sí lo hemos conseguido. No tendría mucho sentido que el sistema no preguntase por la clave, ya que en ese caso cualquiera podría ejecutar comandos como administrador con el peligro que ello supone.

4.3 PERMISOS

La información sobre grupos, usuarios y permisos se puede obtener mediante el comando `ls` junto con la opción `-l`. Vamos a ver los permisos que tiene establecidos el fichero `whatis` que se encuentra en el directorio `/usr/bin`.

```
$ ls -l /usr/bin/whatis
-rwxr-xr-x 1 root root 87792 2008-03-12 14:24 /usr/bin/whatis
```

En la primera columna aparecen los **permisos**, en la tercera se indica el **usuario** (en este caso es el administrador del sistema) y en la cuarta columna aparece el nombre del **grupo** (que en este caso coincide con el de usuario).

Vamos a ver qué significan exactamente los caracteres de la primera columna:

-	r	w	x	r	-	x	r	-	x
Tipo de fichero.	Permisos para el dueño del fichero.			Permisos para el grupo al que pertenece el fichero.			Permisos para el resto de usuarios		

r	Permiso de lectura .
w	Permiso de escritura .
x	Permiso de ejecución .

El tipo de fichero se indica en la siguiente tabla:

<i>Tipo de fichero</i>	
l	Enlace simbólico.
c	Dispositivo especial de caracteres.
b	Dispositivo especial de bloques.
p	FIFO (estructura de datos).
s	Socket (comunicaciones).
-	Ninguno de los anteriores. Puede ser un fichero de texto, un binario, etc.

En el caso que nos ocupa tenemos un carácter “-” como tipo de fichero, porque se trata de un binario (un programa). El dueño del fichero tiene los permisos `rwX`, lo que quiere decir que puede leer, escribir y ejecutar el fichero. Que tiene permiso para escribir significa que puede borrarlo, cambiarle el nombre o editarlo. Tanto el grupo como el resto de usuarios tienen los permisos `r-X`, lo que significa que pueden utilizarlo (pueden leerlo y ejecutarlo) pero no lo pueden modificar.

4.4 ¿QUIÉNES SOMOS? (whoami, groups)

Antes de empezar a crear usuarios, crear grupos y cambiar permisos, debemos saber quiénes somos y a qué grupo o grupos pertenecemos. Aunque, en principio, entremos en el sistema como un determinado usuario, podemos utilizar `SU` para ejecutar comandos como otro usuario distinto, siempre y cuando sepamos la contraseña de ese otro usuario.

```
$ whoami
luisjose
$ su alumno
Contraseña:
$ whoami
alumno
```

Para volver a ser el usuario original basta con utilizar `exit`.

```
$ whoami
alumno
$ exit
exit
$ whoami
luisjose
```

Con el comando `groups` se puede ver a qué grupo pertenecemos.

```
luisjose@luisjose-xps1330:~$ groups
luisjose adm dialout cdrom floppy audio dip video plugdev scanner lpadmin admin
netdev powerdev sambashare
```

Se pueden especificar uno o más usuarios detrás de `groups`. Eso nos dirá a qué grupos pertenece cada uno de ellos.

```
luisjose@luisjose-xps1330:~$ groups alumno root
alumno : alumno
root : root
```

4.5 GESTIÓN DE GRUPOS (`groupadd`, `groupdel`, `groupmod`)

Los comandos `groupadd`, `groupdel` y `groupmod` permiten crear, borrar y modificar grupos respectivamente.

Vamos a crear los grupos `oficina_malaga`, `oficina_jaen` y `oficina_madrid`

```
$ groupadd oficina_malaga
groupadd: incapaz de bloquear el fichero de grupos
$ sudo groupadd oficina_malaga
$ sudo groupadd oficina_jaen
$ sudo groupadd oficina_madrid
```

Vemos que si intentamos crear un grupo como usuario sin privilegios obtenemos un error. Para manejar grupos y usuarios es necesario ejecutar los comandos con privilegios de administrador, por tanto deberemos teclear `SUDO` antes del comando en cuestión.

Hemos escrito mal el nombre del segundo grupo, ¡que no cunda el pánico!, este problema se puede solventar con `groupmod`.

```
$ sudo groupmod -n oficina_madrid oficina_madrid
```

La directiva de la empresa ha decidido cerrar la oficina de Jaén para ahorrar costes y pasar los recursos a la oficina de Málaga, así que no hará falta el grupo `oficina_jaen`. Lo podemos borrar con `groupdel`.

```
$ sudo groupdel oficina_jaen
```

4.6 GESTIÓN DE USUARIOS (`adduser`, `userdel`, `usermod`)

La gestión de usuarios, al igual que la de grupos, exige que los comandos se ejecuten con los privilegios del administrador del sistema. Se puede escribir `SUDO` antes de cada comando, o se puede hacer lo siguiente:

```
$ sudo bash
```

Note el lector que el prompt ha cambiado. Ahora se muestra un carácter “#” en lugar de un “\$”. A partir de ahora, todos los comandos se ejecutarán con privilegios de administrador del sistema. Hay que acordarse de volver al usuario inicial mediante `exit`.

Es necesario dar de alta a dos usuarios para el grupo `oficina_malaga` y uno para `oficina_madrid`. Habrá un cuarto usuario que estará yendo y viniendo de una oficina a otra, por tanto se le dará de alta en las dos.

```
# adduser pedro --ingroup oficina_malaga
# adduser ana --ingroup oficina_malaga
# adduser berta --ingroup oficina_madrid
# adduser laura --ingroup oficina_malaga
# adduser laura oficina_madrid
```

Hemos matado dos pájaros de un tiro. Hemos creado los usuarios y al mismo tiempo los hemos incluido dentro de los grupos correspondientes. Estos dos pasos se pueden hacer de forma independiente. El usuario `laura` pertenece a dos grupos. En primer lugar se ha creado el usuario y al mismo tiempo se ha añadido al grupo `oficina_malaga` con la opción `-ingroup`. Para añadir un usuario existente a un grupo, se utiliza `adduser` sin opciones.

```
# groups ana berta laura
ana : oficina_malaga
berta : oficina_madrid
laura : oficina_malaga oficina_madrid
```

Es importante destacar que se ha utilizado **adduser** y no **useradd**. Este último se utiliza al crear los usuarios, se nos han pedido las claves, no obstante estas claves se pueden cambiar con el comando `passwd`.

```
# passwd pedro
# passwd ana
# passwd laura
```

Recuerde el lector salir del modo `root` con el comando `exit` cuando no tenga que hacer tareas que requieran privilegios de administrador.

```
# exit
```

De ahora en adelante, simplemente se indicará con el carácter “\$” que se trabaja como usuario sin privilegios y con el carácter “#” que se trabaja como `root`.

Cabe señalar que para cada usuario, se crea por defecto un directorio dentro de `/home`. Cuando un usuario se conecta al sistema, “aterriza” en ese directorio. Es lo que hemos denominado anteriormente como el directorio de trabajo.

```
$ ls /home/
alumno ana berta ftp laura luisjose pedro
```

4.7 CAMBIO DE GRUPO Y DE DUEÑO (chown, chgrp)

Imaginemos que el fichero `informe.txt` ha sido creado por el usuario `pedro`. Por defecto, el dueño de un archivo es el usuario que lo crea, en este caso `pedro`. El grupo del usuario `pedro`, como hemos visto antes es `oficina_malaga`.

```
$ su pedro
$ cd
$ pwd
/home/pedro
$ touch informe.txt
$ ls -l
-rw-r--r-- 1 pedro oficina_malaga 0 2009-03-19 12:46 informe.txt
```

Todo esto se puede cambiar. Moveremos el fichero al directorio de trabajo del usuario `laura` y le cambiaremos el dueño.

```
# mv informe.txt /home/laura/
# cd /home/laura/
# chown laura informe.txt
# ls -l
-rw-r--r-- 1 laura oficina_malaga 0 2009-03-19 12:46 informe.txt
```

Ahora el fichero tiene al usuario `laura` como propietario.

Tanto `chown` como `chgrp` se pueden usar con la opción `-R` para cambiar el dueño o el grupo en un directorio completo, de forma recursiva.

4.8 CAMBIO DE PRIVILEGIOS (chmod)

El comando `chmod` sirve para cambiar los permisos de uno o varios ficheros. Esos mismos permisos que se pueden ver con `ls -l`.

```
$ ls -l
-rw-r--r-- 1 pedro oficina_malaga 0 2009-03-19 15:38 hola_mundo.rb
$ chmod +x hola_mundo.rb
$ ls -l
-rwxr-xr-x 1 pedro oficina_malaga 0 2009-03-19 15:38 hola_mundo.rb
```

Hemos añadido el permiso de ejecución al fichero `hola_mundo.rb`. Vemos que ahora hay tres `x`, la que corresponde al dueño del fichero, la de todos los usuarios que pertenecen al grupo y la del resto de usuarios.

Cuando no se especifica ninguna de estas tres letras correspondientes a los usuarios (`u`, `g`, `o`) como en el ejemplo anterior, se sobreentiende que nos referimos a todos ellos. Se puede indicar de forma explícita con el carácter `a` (all).

Para entenderlo mejor, en la siguiente tabla, se muestran de forma esquemática, los parámetros del comando `chmod`:

u	g	o	+ -	r	w	x
(user) dueño del fichero	(group) usuarios que pertenecen al mismo grupo	(others) el resto de usuarios	dar permiso quitar permiso	(read) lectura	(write) escritura	(execution) ejecución

Quitaremos ahora el permiso de ejecución para el resto de usuarios (others) y daremos permiso de escritura (write) a los usuarios del mismo grupo (group).

```
$ ls -l
-rwxr-xr-x 1 pedro oficina_malaga 0 2009-03-19 15:38 hola_mundo.rb
$ chmod o-x hola_mundo.rb
$ chmod g+w hola_mundo.rb
$ ls -l
-rwxrwxr-- 1 pedro oficina_malaga 0 2009-03-19 15:38 hola_mundo.rb
```

A este método, que utiliza los caracteres `rwX` se le denomina método simbólico. Podemos utilizar de forma análoga el método numérico.

4	2	1	Total
r	w	x	4 + 2 + 1 = 7
r	w	-	4 + 2 + 0 = 6
r	-	x	4 + 0 + 1 = 5
r	-	-	4 + 0 + 0 = 4
-	w	x	0 + 2 + 1 = 3
-	w	-	0 + 2 + 0 = 2
-	-	x	0 + 0 + 1 = 1

De esta forma, esta línea

```
$ chmod 755 hola_mundo.rb
```

sería equivalente a estas tres

```
$ chmod u+rwX hola_mundo.rb  
$ chmod g+rx-w hola_mundo.rb  
$ chmod o+rx-w hola_mundo.rb
```

En efecto

```
$ ls -l  
-rwxr-xr-x 1 pedro oficina_malaga 0 2009-03-19 15:38 hola_mundo.rb
```

Los permisos de los directorios se pueden cambiar de la misma forma que los ficheros, aunque el significado es algo diferente. Si un directorio tiene el permiso de lectura quiere decir que se puede ver su contenido. Si tiene permiso de escritura, quiere decir que se pueden crear ficheros dentro y si tiene permiso de ejecución quiere decir que se puede entrar dentro.

RESUMEN DEL CAPÍTULO 4

- Los comandos vistos en este capítulo son los siguientes:

<i>Comando</i>	<i>Acción</i>
ls -l	Muestra, entre otras cosas, información sobre los permisos, el usuario y el grupo al que pertenece el fichero.
sudo	Permite ejecutar comandos como root.
su	Cambia de usuario.
whoami	Muestra el nombre del usuario actual.
groups	Muestra el/los grupos/s a los que pertenece el usuario actual.
groupadd	Añade un nuevo grupo.
groupdel	Borra un grupo.
groupmod	Modifica las características de un grupo.
adduser	Añade un nuevo usuario.
userdel	Borra un usuario.
usermod	Modifica las características de un usuario.
passwd	Asigna o cambia la clave de un usuario.
chown	Cambia el dueño de un archivo.
chgrp	Cambia el grupo al que pertenece un archivo.
chmod	Cambia los permisos.

EJERCICIOS DEL CAPÍTULO 4

En ocasiones, la respuesta a los ejercicios no se puede completar únicamente con el material teórico que se proporciona en este capítulo y el alumno debe, por tanto, buscar en otras fuentes complementarias. En los ejercicios de este capítulo se recomienda consultar las páginas *man*.

Las soluciones a los ejercicios se encuentran al final del libro.

Los ejercicios están clasificados según su nivel de dificultad:

 **Facil.** El concepto viene explicado en el capítulo.

 **Dificultad media.** Es necesario relacionar conceptos y/o buscar información en Internet.

 **Difícil.** Hace falta una investigación concienzuda



1.  Completa la siguiente tabla:

654	
	<code>rwXrw-rw-</code>
	<code>rwXrwxrwx</code>
520	
764	
	<code>r--r-----</code>

2. Crea los grupos **oficina1** y **oficina2**.

3.  Crea los usuarios **paco** y **pablo**. Estos usuarios deben pertenecer únicamente al grupo **oficina1**.

4.  Crea los usuarios **alba** y **nerea**. Estos usuarios deben pertenecer únicamente al grupo **oficina2**.

5.  Como usuario **paco** Crea un fichero con nombre **topsecret.txt** en su directorio de trabajo al que únicamente él tenga acceso, tanto de lectura como de escritura.

6.  Crea otro fichero, también como usuario **paco**, con nombre **ventas_trimestre.txt** al que tengan acceso, tanto para leer como para escribir todos los usuarios que pertenezcan al mismo grupo. Se deben dejar los permisos que haya por defecto para el dueño y para el resto de usuarios. Comprueba como usuario **pablo** que puedes modificar el fichero.

7.  Como usuario **alba**, crea un fichero con nombre **empleados.txt** al que pueda acceder cualquier usuario para leer su contenido, y cualquier usuario del mismo grupo para leer o escribir.

8.  Copia el fichero **empleados.txt** al directorio de trabajo de **alumno** (crea también el usuario **alumno** si no está creado). Cambia el propietario y el grupo al que pertenece el fichero, ahora debe ser **alumno**.

9.  Como usuario **pablo**, copia un programa del directorio **/usr/bin** al directorio de trabajo con un nombre diferente. Por ejemplo **xclock** se puede copiar como **reloj**. Mira los permisos de este

programa. Comprueba que se puede ejecutar. Puede que sea necesario dar permiso para que otros usuarios distintos al actual puedan ejecutar aplicaciones en el entorno gráfico, basta con ejecutar como administrador: `xhost +`.

10.  Cambia los permisos de **reloj** de tal forma que sólo lo pueda ejecutar el propietario del archivo.
11.  Crea el usuario **modesto**, perteneciente a **oficina2**. Dentro de su directorio de trabajo, crea un directorio de nombre **compartido_con_todos**.
12.  Cambia de usuario en el entorno gráfico (botón **salir** y botón **cambiar de usuario**) y entra como modesto. Crea con **OpenOffice.org Calc** los ficheros **telefono_contactos.ods**, **gastos_marzo.ods** y **sueldos.ods**. Inserta varias entradas en cada uno de los ficheros y grábalo todo en el directorio **compartido_con_todos**.
13.  Da permiso de lectura a la carpeta **compartido_con_todos** y a todos los ficheros que contenga para todos los usuarios.
14.  Restringe el acceso de escritura sobre el fichero **telefono_contactos** para que sólo lo puedan modificar los usuarios del grupo al que pertenece su propietario.
15.  Cambia los permisos de **gastos_marzo** para que sólo pueda modificarlo su propietario y leerlo cualquiera del mismo grupo.
16.  Cambia los permisos de **sueldos** para que sólo su dueño tenga acceso a él, tanto para lectura como para escritura.
17.  Si un usuario tiene permiso de lectura sobre un fichero pero ese fichero se encuentra dentro de un directorio sobre el que no tiene permiso de lectura, ¿podrá leer el fichero?, haz la prueba.

Copyright © 2009 Luis José Sánchez González.

Se concede permiso para copiar, distribuir y/o modificar este documento bajo los términos de la Licencia de Documentación Libre de GNU, Versión 1.3 o cualquier otra versión posterior publicada por la Free Software Foundation; sin Secciones Invariantes, sin Textos de Cubierta Delantera y sin Textos de Cubierta Trasera. Una copia de la licencia está incluida en la sección titulada GNU Free Documentation License (Apéndice A)