Parte I

Linux para el usuario

Introducción a Linux

El sistema de archivos

El editor de textos `vi`

El intérprete de órdenes

Expresiones regulares y filtros

Programación del intérprete de órdenes

Servicios de red

El sistema XWindow

Capítulo 1

Introducción a Linux

1.1. Historia

Los antecedentes de Linux hay que buscarlos en el sistema operativo y los orígenes de UNIX se remontan a 1964. En este año, *Bell Telephone Laboratories* de AT&T, *General Electric Company* y el MIT (Instituto Tecnológico de Massachusetts) se plantearon desarrollar un nuevo sistema operativo en tiempo compartido para una máquina GE 645 (de *General Electric*) al que denominaron MULTICS. Los objetivos marcados inicialmente consistían en proporcionar a un conjunto amplio de usuarios una gran capacidad de computación y la posibilidad de almacenar y compartir enormes cantidades de datos si éstos lo deseaban. Todos esos objetivos eran demasiado ambiciosos para la época, sobre todo por las limitaciones del hardware. Como consecuencia de ello, los trabajos en el nuevo sistema operativo iban muy retrasados. Debido a eso, *Bell Laboratories* decidió dar por terminada su participación en el proyecto. A pesar del fracaso de MULTICS, las ideas empleadas para su diseño no cayeron en el olvido, sino que influyeron mucho en el desarrollo de UNIX y de otros sistemas operativos posteriores.

Ken Thompson, uno de los miembros del *Computing Science Research Center* de los *Laboratorios Bell*, encontró un computador DEC (*Digital Equipment Corporation*) PDP-7 inactivo y se puso a desarrollar en él un juego denominado *Space Travel*. El desarrollo de ese juego propició que Thompson adquiriese muchos conocimientos relacionados con la máquina en la que estaba trabajando. Con objeto de crear un entorno de trabajo agradable, Thompson, al que posteriormente se le unió Dennis Ritchie, se propuso la creación de un nuevo sistema operativo, al que denominó UNIX. Ritchie había trabajado anteriormente en el proyecto MULTICS, de mucha influencia en el nuevo sistema operativo. Como ejemplos de esa influencia podemos citar la organización básica del sistema de archivos, la idea del intérprete de órdenes (shell [1]) como proceso de usuario (en sistemas anteriores, el intérprete de órdenes formaba parte del propio núcleo del sistema operativo), e incluso el propio nombre UNIX deriva de MULTICS.

[1] A lo largo del texto utilizaremos el término shell a la hora de referirnos al intérprete de órdenes de Linux. Hemos optado por no emplear la traducción de concha o caparazón porque en la mayoría de los textos aparece el término original.

MULTICS *Multiplexed Information and Computing Service.*

UNICS *Uniplexed Information and Computing Service.*

Realmente, el término UNICS se empleó por la similitud de esta palabra con la palabra inglesa *eunuchs* (eunuco), con lo cual se venía a indicar que este nuevo sistema operativo era un MULTICS capado. Posteriormente, UNICS dio lugar al nombre definitivo UNIX. El nuevo sistema también se vio influenciado por otros sistemas operativos, tales como el CTSS (*Compatible Time Sharing System*) del MIT y el sistema XDS-940 (*Xerox Data System*) de la Universidad de California en Berkeley.

Aunque esta primera versión de UNIX prometía mucho, su potencial no pudo demostrarse hasta que se utilizó en un proyecto real. Así pues, mientras se planeaban las pruebas para patentar el nuevo producto, éste fue trasladado a un computador PDP-11 de Digital en una segunda versión. En 1973 el sistema operativo fue reescrito en lenguaje C en su mayor parte. C es un lenguaje de alto nivel (las versiones anteriores del sistema operativo habían sido escritas en ensamblador), lo que propició que el sistema tuviera una gran aceptación por parte de los nuevos usuarios. El número de instalaciones en *Bell Laboratories* creció hasta 25, aproximadamente, y su uso también se difundió gradualmente a unas cuantas universidades con propósitos educacionales.

La primera versión de UNIX disponible fuera de *Bell Laboratories* fue la Versión 6, en el año 1976. En 1978 se distribuyó la Versión 7, que fue adaptada a otros PDP-11 y a una nueva línea de ordenadores de DEC denominada VAX. La versión para VAX se conocía como 32V.

Tras la distribución de la Versión 7, UNIX se convirtió en un producto y no sólo en una herramienta de investigación o educacional, debido a que el *UNIX Support Group* (USG) asumió la responsabilidad y el control administrativo del *Research Group* en la distribución de UNIX dentro de AT&T.

En el periodo comprendido entre 1977 y 1982, *Bell Laboratories* combinó varios sistemas UNIX, de la Versión 7 y de la 32V, dando lugar a un único sistema cuyo nombre comercial fue UNIX System III. Ésta fue la primera distribución externa desde USG.

La modularidad, la sencillez de diseño y el pequeño tamaño de UNIX, hicieron que muchas entidades, tales como Rand, varias universidades e incluso DEC, se pusieran a trabajar sobre él. La Universidad de Berkeley en California desarrolló una variante del sistema UNIX para máquinas VAX. Esta variante incorporaba varias características interesantes, tales como memoria virtual, paginación por demanda y sustitución de páginas, con lo cual se permitía la ejecución de programas mayores que la memoria física. A esta variante, desarrollada por Bill Joy y Ozalp Babaoglu, se la conoció como 3BSD (*Berkeley Software Distributions*). Todo el trabajo desarrollado por la Universidad de Berkeley para crear BSD impulsó a la *Defense Advanced Research Projets Agency* (DARPA) a financiar a Berkeley en el desarrollo de un sistema UNIX estándar de uso oficial (4BSD). Los trabajos en 4BSD para DARPA fueron dirigidos por expertos en redes y UNIX, DARPA Internet (TCP/IP). Este soporte se facilitó de un modo general. En 4.2BSD es posible la comunicación uniforme entre los distintos dispositivos de la red, incluyendo redes locales (LAN), como *Ethernet* y *Token Ring*, y extensas redes de ordenadores (WAN), como la Arpanet de DARPA.

Los sistemas UNIX actuales no se reducen a la Versión 8, System V o BSD, sino que la mayoría de los fabricantes de micro y miniordenadores ofrecen su UNIX particular.

Así, Sun Microsystems los ofrece para sus ordenadores y lo denomina Solaris, Hewlet Packard lo comercializa con el nombre de HP-UX, IBM lo implantó en sus equipos RISC 6000 y lo denomina AIX, etc. Con el gran incremento en prestaciones de los ordenadores personales, también han aparecido versiones para ellos. Dentro de estas nuevas versiones cabe destacar aquéllas de distribución libre, como pueden ser FreeBSD, OpenBSD o el propio Linux, obtienen un alto rendimiento de los procesadores de la familia 80x86 de Intel (del 80386 en adelante).

1.2. Aparición de Linux

Linux es un sistema operativo de distribución libre desarrollado inicialmente por Linus Torvalds en la Universidad de Helsinki (Finlandia). Una comunidad de programadores expertos en UNIX, han ayudado en el desarrollo, distribución y depuración de este sistema operativo. El núcleo de Linux no contiene código desarrollado por AT&T ni por ninguna otra fuente propietaria. La mayoría del software disponible en Linux ha sido desarrollado por el proyecto GNU de la *Free Software Foundation* de Cambridge (Massachusetts). Sin embargo, es toda la comunidad de programadores la que ha contribuido al desarrollo de aplicaciones para este sistema operativo.

Con la aparición de ordenadores personales potentes aparece Linux. Inicialmente se trató sólo de un desarrollo llevado a cabo por Linus Torvalds por pura diversión. Linux se inspiró en Minix, un pequeño sistema UNIX desarrollado por Andrew S. Tanenbaum, de hecho, en el grupo de noticias `comp.os.minix` aparecen los primeros comentarios que tenían que ver con el desarrollo de un sistema operativo académico que fuese más completo que Minix.

Los primeros desarrollos de Linux tenían que ver con la conmutación de tareas en el microprocesador 80386 ejecutando en modo protegido, todo ello escrito en lenguaje ensamblador. En este punto, Linus comentaba:

> *"Después de esto la cosa era sencilla: todavía era complicado programar, pero disponía de ciertos dispositivos y la depuración resultaba más fácil. En este punto comencé a emplear lenguaje C y esto aceleró en gran medida el desarrollo. Esto supuso tomar en serio mis ideas megalomaniacas con intención de desarrollar 'un Minix mejor que Minix'. Deseaba ser capaz de recompilar gcc bajo Linux algún día..."*
>
> *"El desarrollo básico supuso dos meses de trabajo, disponía de un driver de disco (con muchos errores, pero en mi máquina funcionaba) y un pequeño sistema de archivos. En este punto es cuando desarrollé la versión 0.01 (a finales de agosto de 1991): no estaba contento, no disponía de driver para disquete y no podía hacer muchas cosas todavía. Creo que nadie compiló nunca esta versión. Pero estaba enganchado y no quería parar hasta deshacerme por completo de Minix."*

No se llevó a cabo ningún anuncio de la versión 0.01 de Linux. Por sí misma, esta versión sólo podía compilarse y ejecutarse en una máquina que tuviese cargado Minix.

El 5 de octubre de 1991 Linus dio a conocer la primera versión "oficial" de Linux, ésta fue la versión 0.02. En este punto Linux podía ejecutar el intérprete de órdenes

`bash` (*Bourne Again Shell* de GNU) y `gcc` (el compilador C de GNU) pero no mucho más. Seguía siendo una versión utilizable solamente por *hackers*[2] y no por personal "no cualificado".
Linus escribió en `comp.os.minix`:

> *"¿Añoras aquellos tiempos con Minix-1.1 cuando los hombres eran hombres y escribían sus propios drivers de dispositivo? No tienes ningún proyecto y deseas hincarle el diente a un sistema operativo para adaptarlo a tus necesidades? ¿Te frustras cuando todo funciona bajo Minix? ¿No quieres perder más noches poniendo en marcha un apestoso programa? Entonces puede que este mensaje sea para ti."*
>
> *"Como ya comenté hace un mes, estoy desarrollando una versión de libre distribución de un sistema similar a Minix para ordenadores 386-AT. Al fin he alcanzado un estado en el que el sistema incluso puede ser utilizado (dependiendo de lo que desees), y dejaré todos los programas fuente de libre distribución. Es solamente la versión 0.02... pero he conseguido ejecutar con éxito bash, gcc, gnu-make, gnu-sed, compress, etc. bajo él."*

Después de la versión 0.03, Linus pasó a lanzar la versión 0.10, en este punto fue cuando aumentó considerablemente el número de personas que se apuntó al desarrollo del sistema. Después de varias versiones intermedias, Linus incrementó el número y pasó directamente a la versión 0.95 para reflejar sus deseos de que pronto pasaría a ser una versión "oficial" (generalmente al software sólo se le asigna como número de versión la 1.0 cuando se supone que está en su mayoría libre de errores). Esto ocurrió en marzo de 1992. Un año y medio después, a finales de diciembre de 1993, el núcleo (*kernel*) de Linux estaba en la versión 0.99.pl14, aproximándose asintóticamente a 1.0.

Actualmente Linux es un UNIX en toda regla, compatible POSIX, capaz de ejecutar X-Window, TCP/IP, Emacs, UUCP, correo electrónico, servicios de noticias, etc. La mayoría de los paquetes software de libre distribución han sido portados a Linux y cada vez son más las aplicaciones comerciales disponibles. Actualmente Linux soporta casi todo el hardware existente en el entorno PC y ha sido portado con éxito a otras plataformas como PowerPC de IBM, SPARC de Sun Microsystems o Macintosh. Su robustez y el hecho de ser gratuito ha propiciado que Linux lo empleen como herramienta de desarrollo desde entidades de investigación como la NASA, hasta DreamWorks, Pixar o Industrial Light and Magic. En el campo de los servidores, Linux tiene en la actualidad una importante cuota de mercado y es en el ámbito de escritorio donde está teniendo un crecimiento importante. En resumen, Linux ha pasado en breve de ser un sistema operativo marginal a convertirse en una alternativa a sistemas comerciales como Windows o MacOS X.

Razones del éxito de Linux

Las razones del éxito de Linux hay que buscarlas en la idea de su diseño. Las características más relevantes del sistema son:

[2] El término *hacker* debemos entenderlo en su sentido estricto: gurú o experto en el tema. Muchas veces este término se aplica erróneamente a aquellas personas que operan con "no demasiadas buenas intenciones" en sistemas informáticos. El término correcto para este tipo de personajes es el de *crackers* (siempre en terminología anglosajona).

- Linux ha sido diseñado como un sistema multiusuario en tiempo compartido; es decir, un sistema en el que pueden trabajar varios usuarios simultáneamente compartiendo el procesador y todos los demás recursos del sistema. Cada usuario puede ejecutar varios procesos (programas en ejecución) a la vez.
- El sistema operativo está escrito en un lenguaje de alto nivel (lenguaje C), lo cual hace que sea fácil de leer, entender, modificar y transportar a otras máquinas con una arquitectura completamente diferente.
- La interfaz de usuario (shell) es sencilla y potente, y puede ser reemplazada por otra en cualquier momento si se desea.
- Proporciona primitivas que permiten construir grandes programas a partir de otros más sencillos.
- El sistema de archivos tiene una estructura de árbol invertido de múltiples niveles que permite un fácil mantenimiento.
- Todos los archivos de usuario son simples secuencias de bytes (8 bits), no tienen ningún formato predeterminado.
- Los archivos de disco y los dispositivos de entrada y salida (E/S) se tratan de la misma manera. Las peculiaridades de los dispositivos se mantienen en el núcleo (*kernel*). Esto quiere decir que impresoras, discos, terminales, etc., desde el punto de vista del usuario, se tratan como si fuesen archivos normales.
- La arquitectura de la máquina es completamente transparente para el usuario, lo que permite que los programas sean fáciles de escribir y transportables a otras máquinas con hardware diferente.
- Linux no incorpora diseños sofisticados; de hecho, han sido seleccionados por su sencillez y no por su rapidez o complejidad.
- Linux ha sido desarrollado por y para programadores, por lo tanto siempre ha sido interactivo, y las herramientas para el desarrollo de programas han tenido siempre mucha importancia.
- Desde un principio, los programas fuente estuvieron a disposición del usuario, facilitando en gran medida el descubrimiento y eliminación de deficiencias, así como nuevas posibilidades en su realización.

Todas estas características han hecho de Linux un sistema operativo de referencia, aceptado por completo tanto en el mundo empresarial como en ambientes educacionales.

Esquema de un sistema Linux

La configuración básica de un sistema Linux, de equipos se refiere, es la mostrada en la figura 1.1. A grandes rasgos, podemos distinguir las siguientes partes:

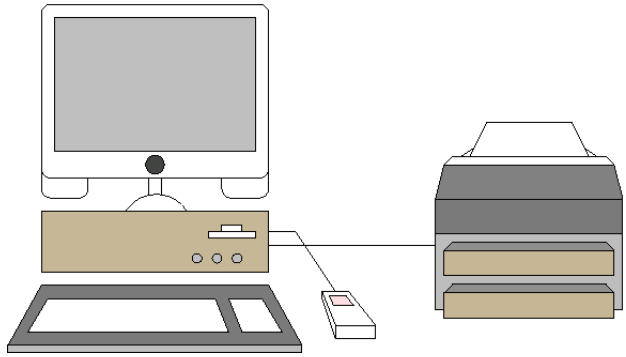

Figura 1.1: Esquema básico de un sistema Linux.

Unidad de proceso. La unidad de proceso es el verdadero corazón del sistema, puesto que en ella se ejecutan todos los programas, tanto los de los usuarios como los del propio sistema. El término unidad de proceso debemos entenderlo en un sentido amplio; es decir, al hablar de él no nos referimos únicamente al procesador, sino que debemos englobar dentro de él elementos tales como la memoria, la unidad de manejo de memoria (UMM), los procesadores en coma flotante, los dispositivos de acceso directo a memoria (ADM), etc. De esta unidad depende el resto del sistema, así como el conjunto de funciones ofrecidas.

Dispositivos de almacenamiento secundario. Los dispositivos de almacenamiento secundario son los elementos en los que vamos a guardar toda la información de forma permanente. Todo el sistema de archivos de Linux que describiremos en capítulos posteriores está montado sobre estos dispositivos. Los medios más comunes de guardar grandes cantidades de información suelen ser los discos rígidos, los discos flexibles, las cintas magnéticas y los CD-ROM.

Dispositivos periféricos. Son aquellos elementos que añadidos al sistema computador realizan, sobre todo, funciones de comunicación con las personas, y entre ellos podemos citar el ratón, la pantalla, el módem, la impresora, el lápiz USB, etc. Todos estos dispositivos están conectados a la central de proceso, la cual se encarga de manejarlos y planificarlos para que puedan ser compartidos sin problemas entre los usuarios.

1.3. Inicio de una sesión Linux

Antes de iniciar nuestra primera sesión de trabajo, debemos tener instalado Linux. Habitualmente, la instalación requiere que tengamos conocimientos de administración del sistema, con lo cual nos metemos en un círculo vicioso, ya que no podemos aprender porque no tenemos el sistema instalado y no podemos instalarlo porque todavía no conocemos el propio sistema.

Para ejecutar Linux en su máquina, tendrá que instalarlo previamente a partir de cualquiera de las diferentes distribuciones libres de este sistema operativo (Ubuntu, Fe-

dora, RedHat, Debian, S.u.S.E., etc.). El proceso de instalación de Linux ha mejorado muchísimo desde las primeras versiones. Actualmente casi es inmediato (mientras no se presente ningún problema), pero requiere que nos carguemos con una pequeña dosis de paciencia y de tiempo. Para aquellas personas que no tienen instalado Linux en su sistema, será necesario llevar a cabo la instalación del mismo previamente. Para ello será necesario leer la documentación proporcionada por el distribuidor o cualesquiera de los múltiples manuales que pueden encontrarse fácilmente en Internet. Un consejo que debería seguir en este punto es el de solicitar ayuda a alguna persona que conozca el proceso de instalación de Linux. Esta instalación puede suponer entre treinta minutos y una hora si no surge ningún problema.

Suponiendo que Linux está instalado en nuestro ordenador, después de iniciar el sistema aparecerá en la pantalla un mensaje similar al siguiente:

```
Ubuntu 7.10 valdebits /dev/pts/0

valdebits login:
```

En este punto teclearemos nuestro nombre de usuario (suponiendo que tenemos cuenta en el sistema) y pulsaremos la tecla ENTRAR. A continuación aparecerá un mensaje como el siguiente.

`password:` *(no se visualiza)*

En este momento, teclearemos nuestra clave y pulsaremos la tecla ENTRAR. Si la clave de acceso es correcta, iniciaremos la sesión, si no es así, no podremos entrar. La clave tecleada no se visualizará en pantalla para evitar que algún curioso pueda verla.

Si todo es correcto, una vez introducidos nuestro nombre de conexión y nuestra palabra clave, aparecerá una presentación similar a la siguiente:

```
Last login: Tue May 29 13:24:48 on tty1

[chan@valdebits chan] $
```

El símbolo `[chan@valdebits chan]$` (*prompt*) que aparece al final es generado por el intérprete de órdenes o shell (caparazón) para indicarnos que está esperando que le demos alguna orden. Este *prompt* puede ser cambiado por el usuario; más adelante veremos cómo, pero por defecto en nuestro sistema es el símbolo mostrado. Como podemos apreciar se compone de dos partes separadas por el carácter `@`. La primera parte coincide con nuestro nombre de conexión (`chan`) y la segunda con el nombre de nuestro ordenador (`valdebits`). A continuación aparece la cadena `chan` que nos indica el directorio donde nos encontramos situados, el cual inicialmente coincide con nuestro directorio de conexión. En otros sistemas el *prompt* por defecto es el carácter `$`. Nosotros a lo largo del libro mostraremos todos los ejemplos con el *prompt* `$` que es el empleado por defecto en muchos sistemas.

Si su sistema utiliza un procedimiento de conexión gráfico porque utilice X-Window con algún gestor de pantalla del tipo `xdm`, `gdm` o `kdm`, el procedimiento de conexión será similar. En este caso, también será necesario introducir un nombre de conexión (`login`) y una palabra clave (`password`), en una ventana similar a la presentada en la figura 1.2. Tal y como ocurría con el inicio de conexión en modo texto, si todo es correcto iniciaremos una

Figura 1.2: Ventana de inicio de sesión presentada por GNOME en Ubuntu.

sesión de trabajo, pero en este caso utilizando ventanas. Para poder comenzar a trabajar con las órdenes del sistema, será necesario iniciar una aplicación de tipo terminal como puede ser `xterm`, `gnome-terminal`, `kterm`, `eterm` o similar. Todas las órdenes que comentemos funcionarán del mismo modo, tanto si trabajamos en un terminal alfanumérico, como si trabajamos con un terminal gráfico.

1.4. Ejecución de órdenes

La forma de invocar cualquier orden y, en general de ejecutar cualquier programa, consiste en teclear su nombre y a continuación pulsar la tecla ENTRAR. Lo más común es que todas las órdenes admitan opciones modificadoras que suelen comenzar con un signo `-` (menos), además de los parámetros adicionales que necesite, tales como nombres de archivos, dispositivos físicos, nombres de usuario, etc. Los distintos parámetros deben ir separados por espacios en blanco para que sean identificados como tales. Debemos tener cuidado con las letras mayúsculas y minúsculas, puesto que Linux, al contrario que otros sistemas operativos, las distingue. Éste es un aspecto muy importante que debemos tener en cuenta.

Ejemplo de orden:

```
$ ls -l parser.c parser.h

-rw-r--r--   1 oscar    oscar        2657 Apr 23  2007 parser.c
-rw-r--r--   1 oscar    oscar         292 Apr 23  2007 parser.h
```

Como veremos en capítulos posteriores, la orden `ls` muestra los archivos que residen en un determinado directorio. En el caso del ejemplo le hemos añadido tres parámetros: `-l`, `parser.c` y `parser.h`. `-l` es un parámetro modificador que advierte a la orden `ls` que debe mostrar los archivos en formato largo, con toda la información referente al archivo. `parser.c` y `parser.h` son dos archivos que queremos visualizar en el formato anteriormente indicado.

En el caso de utilizar varios parámetros modificadores, éstos pueden ir seguidos sin necesidad de colocar espacios en blanco entre ellos.

Ejemplo:

```
$ ls -li parser.c parser.h

1079321 -rw-r--r--   1 oscar  oscar  2657 Apr 23  2007 parser.c
1079322 -rw-r--r--   1 oscar  oscar   292 Apr 23  2007 parser.h
```

En el caso del ejemplo, los modificadores `-l` y `-i` los hemos agrupado en uno solo: `-li`. También sería válida la expresión `ls -l -i parser.c parser.h`, aunque requiere escribir más.

Si intentamos ejecutar la orden anterior, pero empleando letras mayúsculas, ocurre lo siguiente:

```
$ LS -L

/bin/bash: LS: orden no encontrada
```

ya que, como hemos indicado previamente, Linux diferencia entre letras mayúsculas y minúsculas.

Si al teclear una orden nos equivocamos, tendremos tres modos de solucionar el problema para eliminar los caracteres que no son válidos:

<BackSpape> Elimina el último carácter tecleado.

<Ctrl-w> Elimina la última palabra.

<Ctrl-u> Elimina toda la línea de órdenes.

1.5. Algunas órdenes para comenzar

Vamos a ver a continuación la sintaxis y función de algunas órdenes sencillas con objeto de familiarizarnos con la técnica general utilizada en Linux para invocar programas.

exit

Sintaxis: exit

Cuando deseamos finalizar una sesión de trabajo, deberemos informar de ello al sistema. La orden `exit` se emplea para avisar al sistema de nuestro fin de sesión en modo terminal. Cuando ejecutamos esta orden, UNIX libera el terminal que estamos utilizando para que pueda conectarse otro usuario. Es aconsejable desconectarse del sistema siempre que nos alejemos del terminal. De esta manera, evitaremos que cualquier curioso pueda aprovechar esta circunstancia para acceder a nuestros archivos como si fuese el propietario. Si ocurriera eso, el sistema entendería que el usuario sigue conectado, y el intruso tendría plenos derechos para visualizar nuestros archivos, hacer copias, modificarlos y, en el peor de los casos, hasta borrarlos.

En sí mismo, Linux es un sistema muy seguro, porque proporciona todo tipo de mecanismos para protegernos de posibles enemigos. Pero, en última instancia, es el usuario el que se debe servir de las posibilidades que el sistema le brinda para protegerse. No sirve de nada que tengamos una caja fuerte de alta seguridad si la combinación para abrirla se puede conseguir fácilmente. El usuario debe cuidar mucho el que alguien pueda obtener su contraseña. Una forma de disminuir el riesgo es cambiarla periódicamente. También es buena costumbre desconectarse del sistema cuando debamos abandonar el terminal temporalmente.

Como decíamos, la forma de finalizar una sesión es tecleando `exit` o también pulsando Ctrl-d (^d), lo cual provocará el mismo efecto.

Ejemplo:

```
$ exit
Ubuntu 7.10 valdebits /dev/pts/0

valdebits login:
```

Al finalizar la sesión, vuelve a presentarse por pantalla el mensaje `login`. Con ello, el sistema nos invita a que iniciemos de nuevo otra sesión. La persona que inicie la nueva sesión puede ser cualquiera de las que tenga cuenta en el sistema. Los terminales de acceso, en caso de disponer de varios, no están asignados de forma fija a cada usuario; así pues, podremos iniciar la sesión desde distintos terminales, siempre con la misma identidad. Con Linux, tenemos la posibilidad de trabajar con terminales virtuales. Para conmutar de uno a otro, si trabajamos en modo texto, no tendremos más que pulsar simultáneamente las teclas Alt+F1, Alt+F2, Alt+F3, etc., de modo que conmutaremos a los terminales virtuales uno, dos, tres, etc., respectivamente. De este modo, y en la misma máquina, podremos tener iniciadas distintas sesiones de trabajo perfectamente diferenciadas.

who

Sintaxis: `who [am i]`

La orden `who` nos informa acerca de quién o quiénes están conectados actualmente al sistema. También muestra información, en la segunda columna, relativa al terminal asociado a cada usuario, y por último, en la columna tercera, la fecha y hora en la que el usuario entró en sesión. No debe extrañarnos el hecho de que pueda haber varias personas trabajando simultáneamente con el mismo ordenador. Incluso un mismo usuario puede tener varias sesiones abiertas simultáneamente. Esto anterior es factible por el hecho de que Linux es un sistema operativo multiusuario y multitarea.

Ejemplo:

```
$ who

oscar    tty7         2008-05-05 23:34 (:0)
oscar    pts/0        2008-05-08 01:45 (:0.0)
oscar    pts/1        2008-05-08 02:00 (:0.0)
oscar    pts/2        2008-05-08 02:09 (:0.0)
```

Si la orden `who` se ejecuta con el parámetro `am i`, visualizará por pantalla su nombre de conexión (`login`), su terminal asociado (al que está conectado) y la fecha y hora de inicio de sesión. Esta opción es útil en el caso de que hayamos modificado previamente nuestra identidad varias veces y queramos saber quiénes somos en cada instante. Posteriormente veremos cómo podemos modificar nuestra identidad.

Ejemplo:

```
$ who am i

oscar    pts/0        2008-05-08 01:45 (:0.0)
```

Podría darse el caso de que un usuario estuviese conectado de forma remota al sistema. En tales circunstancias, la orden `who` visualizaría también el nombre de la máquina desde la que el usuario se encuentra conectado. Dicho de otro modo, no es necesario estar físicamente conectado al terminal Linux, una sesión en el sistema.

mail

Sintaxis: `mail [usuario(os)]`

El sistema UNIX proporciona un mecanismo de correo electrónico o *e-mail* que permite enviar mensajes de unos usuarios a otros. Para enviar un mensaje, no es necesario que el usuario destinatario esté conectado en ese instante, ya que toda la correspondencia será depositada en su buzón, que podrá consultar posteriormente. Si tenemos correo pendiente, en el inicio de sesión podrá aparecer un mensaje como el siguiente `You have new mail` (tiene correo nuevo), indicándonos que tenemos mensajes en el buzón.

Esta orden puede utilizarse con o sin parámetros. Si la empleamos sin parámetros, visualizará en pantalla los diferentes mensajes, con su correspondiente remitente, que contenga nuestro buzón. Para pasar de un mensaje a otro, pulsaremos ENTRAR sin más, y si queremos eliminar el mensaje, pulsaremos `d` (*delete*). También tenemos la posibilidad de imprimir el mensaje visualizado pulsando `p` (*print*), o de guardarlo en un archivo pulsando `s` y a continuación el nombre del archivo (`s nombre_de_archivo`). Para salir de `mail`, simplemente pulsaremos `q` (*quit).* Todas estas opciones de `mail`, y algunas más, las podemos visualizar si pulsamos `?` (*help*) dentro de la propia orden.

Ejemplo:

```
$ mail

Mail version 8.1 6/6/93. Type ? for help.

"/var/spool/mail/chan": 2 messages 1 new 2 unread
 U 1 lucas@valdebits.aut.uah Mon Nov 16 12:47 14/368 "Prueba"
>N 2 lucas@valdebits.aut.uah Mon Nov 16 12:51 17/413 "Comida"
&
```

Si queremos leer el mensaje número 2, pulsaremos el número 2 y daremos ENTRAR:

```
& 2

Message 2:
From lucas Tue May 29 17:51:38 2004
Date: Tue May 29 17:51:38 2004 +200
From: lucas@valdebits.aut.uah.es
To: chan@valdebits.aut.uah.es
Subject: Comida...

Quedamos a comer a las dos...

¿Te parece?

Un saludo!

&
```

Si queremos ver la ayuda en línea que proporciona esta utilidad daremos la orden `?`.

```
& ?

Mail Commands

t <message list> type messages
n goto and type next message
e <message list> edit messages
f <message list> give head lines of messages
d <message list> delete messages
s <message list> file append messages to file
u <message list> undelete messages
R <message list> reply to message senders
r <message list> reply to message senders and all recipients
pre <message list> make messages go back to /usr/spool/mail
m <user list> mail to specific users
q quit, saving unresolved messages in mbox
x quit, do not remove system mailbox
h print out active message headers
! shell escape
cd [directory] chdir to directory or home if none given
A <message list> consists of integers, ranges of same, or user names
separated by spaces. If omitted, Mail uses the last message typed.
A <user list> consists of user names or aliases separated by spaces.
Aliases are defined in.mailrc in your home directory.

&
```

Para salir daremos la orden `q`:

```
& q
Saved 2 messages in mbox
$
```

También podremos usar la orden `mail` pasándole como parámetro el nombre de un usuario, y así podremos enviarle correo. Por ejemplo, si queremos contestar a Lucas operaríamos del modo siguiente:

```
\$ mail lucas

Subject: comida sí
De acuerdo.
Nos vemos a las dos,
un saludo.
.

Cc:
$
```

Después de invocar a `mail`, todo lo que tecleemos será interpretado por `mail` y no por el shell. Podremos incluir en el mensaje el número de líneas que queramos. Para finalizar el mensaje, pulsaremos el carácter punto "." o Ctrl-d (^d).

Existen muchas variantes de `mail`, cada una de ellas con sus propias peculiaridades, entre ellas podemos citar `mailx`, `elm`, `ean`, etc., aunque las más utilizadas actualmente son aquellas que disponen de una interfaz gráfica como `thunderbird`, `kmail` o `evolution`.

write

Sintaxis: write usuario

La orden `write` se utiliza para comunicarnos con otros usuarios que estén en ese momento conectados a nuestro mismo sistema (`write` no sirve para comunicarnos con usuarios ubicados en sistemas diferentes aunque se disponga de una red). El mensaje puede ser todo lo extenso que deseemos, y para terminar pulsaremos Ctrl-d (^d). Si intentamos enviar un mensaje a un usuario no conectado, se nos advertirá de que dicho usuario no se encuentra en sesión. Puede ocurrir que el usuario al que le enviamos el mensaje tenga desactivados los mensajes, en cuyo caso `write` también fallará. El usuario destinatario recibirá una cabecera como la siguiente, acompañada de un pitido.

```
Message from lucas@valdebits.aut.uah.es on ttyp1 at 13:06...
¿Estás ahí?
EOF
```

Para contestar a un mensaje enviado del modo anterior, debemos hacer algo similar a lo siguiente:

```
$ write lucas
Claro que estoy. Pásate por mi despacho
- Ctrl-d -
$
```

Lo normal es que cuando iniciamos una comunicación con otro usuario, éste nos responda también invocando a `write`, de tal manera que se establece una comunicación bidireccional. Es muy común que en momentos en que el sistema está muy cargado la salida de `write` se vea retrasada considerablemente, con lo que un usuario puede decidir responder a la persona que llama sin haber recibido el mensaje completo. Para evitar esta situación, lo normal es seguir un protocolo ampliamente difundido, que consiste en añadir una "`o`" (*over*) para cambiar, y emplear dos oes "`oo`" (*over and out*) para cambiar y cerrar la comunicación. Este protocolo no lo impone `write`, sino que se trata solamente de una norma muy común entre usuarios de Linux.

mesg

```
Sintaxis: mesg [y/n]
```

Esta orden se utiliza para modificar los derechos de escritura por parte de otros usuarios en nuestro terminal, de tal manera que si alguien nos quiere enviar un mensaje y tenemos desactivados estos derechos, no seremos interrumpidos. La prohibición de acceso de escritura no afecta al administrador del sistema. La orden `mesg` sin parámetros nos dirá si tenemos o no activa la recepción de mensajes.

Ejemplos:

```
$ mesg
is y
$ mesg n
$ mesg
is n
$
```

Cuando tenemos los mensajes desactivados, no recibiremos ninguno aunque alguien nos los envíe. Estos mensajes se perderán incluso si después volvemos a habilitar la posibilidad de recibirlos.

date

```
Sintaxis: date
```

La orden `date` informa sobre la fecha y la hora actuales. Para ello, `date` consulta previamente el reloj hardware del sistema, el cual incrementa su valor a intervalos regulares de tiempo. Estos intervalos suelen ser pequeños, de manera que pueda obtenerse bastante resolución. Este reloj sigue funcionando por medio de una batería aunque se apague el ordenador, para que siempre podamos tener una noción del tiempo correcta sin necesidad de actualizar dicho reloj cada vez que iniciamos el ordenador. Existen multitud de órdenes y programas que también utilizan este reloj para consultarlo y tomar decisiones en función del valor leído. Existen distintas opciones de la orden `date` que afectan al formato de salida. Colocando un campo determinado a continuación del operador `%`, precedido del signo `+`, podemos obtener respuestas como la del ejemplo que mostramos seguidamente.

Ejemplo:

```
$ date +"Son las \%r del \%d de \%h de \%y"
Son las 02:56:42  del 08 de may de 08
$
```

Los operadores asociados a `%` son:

`r` Hora en formato AM-PM

`d` Día del mes

`m` Mes

`y` Año

`w` Día de la semana

`H` Hora

`M` Minuto

`S` Segundo

De todas formas, la manera más común de utilizar la orden es la siguiente:

```
$ date
jue may  8 02:57:25 CEST 2008
```

La orden `date` también puede utilizarla el administrador del sistema para modificar el valor de cuenta del reloj hardware, y en consecuencia, la fecha y la hora. Los usuarios normales no pueden modificar ni la fecha ni la hora. Sólo podrá hacerlo la persona que posea los privilegios adecuados. Estos mecanismos de protección aseguran que el sistema funcione correctamente. Si todo el mundo que tiene acceso al sistema pudiese hacer lo que le viniese en gana, probablemente el sistema se convertiría en algo totalmente descontrolado.

echo

Sintaxis: echo cadena de caracteres

La orden `echo` repite todo lo que le pasemos como parámetro. Esta orden se utiliza mucho dentro de los programas de shell que veremos más adelante, y también para visualizar las variables del intérprete de órdenes. Las variables comentadas las utiliza el propio shell para almacenar valores de configuración e información.

Ejemplos:

```
$ echo Esta orden repite todo
Esta orden repite todo
$ echo $TERM
xterm
$
```

El ejemplo `echo $TERM` nos dice qué tipo de terminal estamos usando en ese momento. En este caso, vemos que se trata de un terminal `xterm`. `TERM` es una de las variables del shell comentadas anteriormente. Si a la orden `echo` le pasamos como parámetro la opción `-n`, entonces la salida no terminará con el carácter de nueva línea, de manera que el cursor queda colocado al final de la línea.

cal

```
Sintaxis: cal [mes] [año]
```

Sin ningún parámetro, `cal` visualiza el calendario correspondiente al mes actual. Si le pasamos como parámetro un año, por ejemplo 2004, mostrará el calendario completo correspondiente al año en cuestión. También podremos indicarle que nos informe sobre un mes en particular del año deseado, pasándole como primer parámetro el número del mes (1, 2, 3,..., 12), y como segundo parámetro, el año. Seguidamente se muestra un ejemplo que ilustra el uso de esta orden.

```
$ cal 5 2008

    mayo de 2008
do lu ma mi ju vi sá
             1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31
```

Si no especificamos ningún mes del año, esta orden visualizará todos los meses del año que le indiquemos.

uname

```
Sintaxis: uname [-amnrsv]
```

La orden `uname` se utiliza para obtener información acerca de nuestro sistema UNIX. Con ella podemos saber el tipo de máquina que estamos utilizando, la versión del sistema operativo, el tipo de procesador, etc. Las opciones más comunes se muestran a continuación:

`-a` Visualiza todo acerca de la máquina que estemos utilizando. Es equivalente a todas las opciones que se muestran a continuación

`-m` Tipo de hardware utilizado

`-n` Nombre de nodo

`-r` Actualización del sistema operativo

`-s` Nombre del sistema

`-v` Versión del sistema operativo

Ejemplo:

```
$ uname -a
Linux cabezon 2.6.22-14-generic #1 SMP
Tue Feb 12 07:42:25 UTC 2008 i686 GNU/Linux
```

Obviamente, si ejecutásemos esta orden en otra máquina, los resultados serían diferentes. Suponiendo que estuviésemos en otro sistema, los resultados podrían ser similares a los siguientes:

```
$ uname -a
Linux fatboy 2.6.24-16-generic #1 SMP
Thu Apr 10 12:47:45 UTC 2008 x86_64 GNU/Linux
```

passwd

Sintaxis: passwd [usuario]

La orden `passwd` se utiliza para modificar nuestra clave de acceso. El cambio de palabra clave debe hacerse con frecuencia por razones de seguridad. Cuando solicitamos un cambio de clave, `passwd` nos pide siempre nuestra antigua palabra de acceso, y lo hace así para comprobar nuestra identidad. De este modo, evita que alguien pueda cambiar nuestra contraseña si abandonamos temporalmente el terminal. Normalmente, en muchos sistemas no puede utilizarse cualquier contraseña, sino que ésta debe cumplir ciertas condiciones como las siguientes: poseer una longitud mínima, tener algún carácter especial, diferenciarse de la última clave en un mínimo de caracteres, no coincidir con el nombre de conexión (*login*), etc. Sólo el administrador del sistema no está sujeto a estas reglas. Cuando introducimos una palabra clave que cumple todas las especificaciones, se nos pide que la repitamos para evitar que nos confundamos al teclear.

Ejemplo:

```
$ passwd

Changing password for chan
(current) UNIX password: (No se visualiza lo escrito)
New UNIX password: (No se visualiza lo escrito)
BAD PASSWORD: case changes only
New UNIX password: (No se visualiza lo escrito)
BAD PASSWORD: it's WAY too short
New UNIX password: (No se visualiza lo escrito)
Retype new UNIX password: (No se visualiza lo escrito)

$
```

A la hora de elegir la palabra clave es bueno tener en cuenta ciertos aspectos que resumimos seguidamente:

- La palabra clave debe tener al menos seis letras, aunque es recomendable que tenga ocho o más.
- La palabra clave no debe aparecer en un diccionario. Si tenemos acceso a la palabra clave encriptada y poseemos un diccionario donde aparezca la contraseña, función `crypt` y un poco de paciencia, podremos descubrir la clave del usuario. Si la clave no aparece en ningún diccionario y además tiene la longitud adecuada, el proceso de descubrirla es algo muchísimo más complicado. Por este motivo es bueno elegir claves que combinen letras, números y caracteres especiales y además sean fáciles de recordar.

lpr

```
Sintaxis: lpr [-m] [-h] [-#n] archivo(s)
```

La orden `lpr` permite enviar archivos a la impresora que haya por defecto para que sean impresos. Estos archivos se colocarán en la cola de impresión en el orden en que se los pasemos. La cola de impresión es una cola que mantiene Linux, y en ella figuran todos los archivos que deben ser impresos.

Las opciones más comunes de `lpr` son:

`-m` (*mail*) Con esta opción, cuando se termina de imprimir el trabajo, `lpr` envía correo avisándonos de que podemos ir a recoger el trabajo.

`-h` Se utiliza para eliminar la cabecera del trabajo que se envía por defecto.

`-#n` Sirve para indicar el número de copias que queremos hacer. Si, por ejemplo, queremos tres copias, debemos indicárselo a `lpr` del modo siguiente:

```
$ lpr -#3 nom_archivo!
```

Ejemplo:

```
$ lpr programa.c
$
```

script

```
Sintaxis: script [-a] [archivo]
```

Esta orden se utiliza para almacenar en un archivo todo lo que el usuario teclee a partir del momento en que sea invocada, así como todo lo que es enviado a la pantalla. Para dejar de grabar información en el archivo, tenemos que invocar a la orden `exit`. Si deseamos guardar todo el contenido de una sesión en un archivo denominado `csesion`, daremos la siguiente orden:

```
$ script csesion
Script iniciado; el archivo es csesion
$
```

Si a `script` no se le especifica ningún archivo, enviará toda la salida a un archivo denominado `typescript`. La opción `-a` la emplearemos cuando queramos añadir información a un archivo. Esta orden puede ser muy útil para usuarios principiantes, ya que de este modo se les permite analizar con posterioridad todas las órdenes ejecutadas y sus resultados.

man

```
Sintaxis: man [sección] [-k] orden
```

Todas las órdenes vistas, y las que veremos en subsiguientes capítulos, están descritas en lo que se conoce como Manual del Programador de Linux. Dicho manual está dividido en secciones, que contienen lo siguiente:

- Sección 1. Órdenes y programas de aplicación.
- Sección 2. Llamadas al sistema.
- Sección 3. Funciones de biblioteca.
- Sección 4. Dispositivos.
- Sección 5. Formatos de archivos.
- Sección 6. Juegos.
- Sección 7. Miscelánea.
- Sección 8. Procedimientos de mantenimiento y administración del sistema.

Lo normal es que el manual esté cargado en el disco, con lo cual podremos consultarlo en todo momento para solventar cualquier problema. Así, para informarnos acerca de la orden `clear`, debe teclearse:

```
$ man clear

Formatting page, please wait...

clear(1)                                                clear(1)

NAME

        clear - clear the terminal screen

SYNOPSIS

        clear

DESCRIPTION
```

```
        clear clears your screen if this is possible. It looks in
        the environment for the terminal type and then in the ter-
        minfo database to figure out how to clear the screen.

SEE ALSO
        tput(1), terminfo(5) clear(1)
$
```

Como podemos observar, `man` nos ofrece una información bastante completa acerca de la orden especificada. La expresión `clear(1)` quiere decir que `clear` se encuentra en la primera sección del manual. La explicación nos indica que `clear` sirve para borrar la pantalla, y que para ello se sirve de la información de entorno y de la base de datos `terminfo`. Por último, nos dice que si queremos más información consultemos la palabra `tput` y `terminfo`, cuya explicación reside en las secciones 1 y 5 del manual respectivamente.

Generalmente, la explicación no es tan breve como la del ejemplo, sino que suele ser mucho más amplia, y en esos casos es conveniente conocer lo siguiente:

- Si pulsamos ENTRAR, visualiza la siguiente línea.
- Si pulsamos espacio, visualiza la siguiente pantalla.
- Si pulsamos `u`, visualiza la pantalla anterior.
- Si pulsamos `Q` o `q`, salimos.

En algunos casos es necesario especificar la sección del manual donde se halla la información deseada; en esos casos, la forma de especificar esta sección es la siguiente: `man n_seccion orden`.

Ejemplo:

```
$ man 2 chmod

CHMOD(2)        Manual del Programador de Linux       CHMOD(2)

NOMBRE

       chmod, fchmod - cambia los permisos de un fichero

SINOPSIS

       #include <sys/types.h>
       #include <sys/stat.h>

       int chmod(const char *path, mode_t mode);
       int fchmod(int fildes, mode_t mode);
```

```
DESCRIPCION

       Cambia el modo del fichero dado mediante path o
       referido por fildes

       Los  modos  se  especifican mediante un O logico
       de los siguientes valores:

       S\_ISUID   04000 asignar ID de usuario al ejecutar
       S\_ISGID   02000 asignar ID de group al ejecutar
       S\_ISVTX   01000 bit pegajoso (sticky bit)
:
```

Los dos puntos que aparecen en la parte inferior izquierda de la pantalla sirven para indicarnos qué orden deseamos dar (espacio, `u`, `q`, etc.).

También podremos obtener información acerca del propio manual, para lo cual daremos la orden `man man`. Existe una orden, denominada `apropos`, que permite obtener información acerca de cualquier término que desconozcamos y que aparezca en el manual de Linux. La orden `apropos` tiene la misma funcionalidad que la orden `man` con el parámetro `-k`. Esto puede sernos útil cuando deseemos información acerca de alguna orden que desconozcamos y que tenga relación con el término que pasamos como parámetro. En el ejemplo siguiente vamos a obtener todas las órdenes, archivos y términos relacionados con la palabra `terminal`. Siempre se nos dará información sobre la sección del manual donde se encuentra el elemento relacionado con el término buscado.

```
$ man -k ftp

apt-ftparchive (1)   - Utility to generate index files
cftp (1) [conchftp]  - Conch command-line SFTP client
conchftp (1)         - Conch command-line SFTP client
curlftpfs (1)        - mount a ftp host as a local directory
ftp (1)              - Internet file transfer program
lftp (1)             - Sophisticated file transfer program
lftpget (1)          - get a file with lftp(1)
lsftp (1)            - Client for the sftp subsystem
Net::Cmd (3perl)     - Network Command class (as used by FTP, SMTP etc)
Net::FTP (3perl)     - FTP Client class
netkit-ftp (1)       - Internet file transfer program
netrc (5)            - user configuration for ftp
obexftp (1)          - Mobile Equipment file transfer tool
pam_ftp (8)          - PAM module for anonymous access module
pftp (1)             - Internet file transfer program
Regexp::Common::URI::ftp (3pm) - - Returns a pattern for FTP URIs.
sftp (1)             - secure file transfer program
sftp-server (8)      - SFTP server subsystem
```

1.6. Ejercicios

1.1 Inicie una sesión de trabajo en Linux. ¿Qué *prompt* aparece? Intente ejecutar alguna orden. Finalice la sesión con `exit` o con Ctrl-d para comprobar que todo es correcto. ¿Qué pasaría si invocásemos a `exit` pero utilizando letras mayúsculas?

1.2 Vuelva a iniciar sesión y compruebe quién o quiénes están conectados al sistema y en qué terminal. Envíe un mensaje por correo al usuario que desee. Envíe otro mensaje, pero utilizando la orden `write`. ¿Qué diferencias hay entre `mail` y `write`? ¿Cómo se pueden evitar los mensajes enviados desde otro terminal con `write`?

1.3 ¿Tiene correo pendiente? Léalo.

1.4 Impida que otros usuarios le envíen mensajes. Habilite de nuevo la comunicación.

1.5 Intente enviar un mensaje de correo a un usuario que no exista. ¿Qué ocurre? ¿Dónde está el mensaje?

1.6 ¿Qué ocurre si invocamos a la orden `date` con la opción `-l`? Si la fecha y hora no son correctas, ¿cómo pueden ser modificadas?

1.7 Visualice la hora en el formato siguiente: `Son las HH horas y MM minutos`.

1.8 ¿Qué tipo de terminal está utilizando?

1.9 Visualice el calendario de 1950 y el del mes actual.

1.10 Visualice el mes de septiembre de 1752. Consulte mediante el manual la orden `cal` para comprobar qué pasó en el año 1752.

1.11 Determine el día de la semana en que nació.

1.12 Modifique su palabra de acceso y reinicie la sesión. ¿Qué ocurre si intenta acceder con su antigua palabra clave?

1.13 Visualice la siguiente información relacionada con su sistema: nombre, versión del sistema operativo y hardware que lo soporta.

1.14 ¿Qué órdenes están relacionadas con `uname`? ¿Y con `passwd`? Utilice el manual para resolver las anteriores preguntas.

1.15 Utilice el manual para consultar las opciones de `banner`. Obtenga información relativa al término `time`. Obtenga información de la llamada al sistema `open`.

1.16 Busque los juegos que estén cargados en su máquina. Para ello, consulte el manual y localice la sección de juegos.

1.17 ¿Dónde se localiza la orden `login`? ¿En qué sección del manual se halla? ¿Para qué puede utilizarse?