INFORMÁTICA BÁSICA

LIBRE CONFIGURACIÓN II, ITIG E ITIS

TEMA 1: SISTEMAS OPERATIVOS Y ENTORNOS DE USUARIO

Índice

1. El Sistema Operativo

En el tema anterior aprendimos que no existe un único modelo de ordenador, sino que, en realidad, existen varias *arquitecturas*. Incluso dentro de una misma arquitectura podemos encontrarnos con modelos posiblemente muy diferentes entre sí: por ejemplo un ordenador portátil es muy distinto de un ordenador de "sobremesa", aunque ambos sean PC's.

Figura 1: Diferentes tipos de ordenadores: un PC de sobremesa y un portátil.

¿Es necesario que tengamos que aprender a manejar y/o programar cada ordenador como si fuese totalmente diferente a los demás? Esto haría muy ineficiente el manejo de ordenadores y, desde luego, imposibilitaría su manipulación a usuarios que no tuviesen conocimientos altamente técnicos.

Para solucionar los problemas derivados de la *heterogeneidad* de los ordenadores existe un programa denominado Sistema Operativo (SO), que se encarga de "manejar" el ordenador por nosotros. Como vimos en el tema anterior, el ordenador controla los dispositivos y recibe nuestras órdenes para interactuar con ellos. Por tanto, el SO es un programa que proporciona una *visión homogénea* de computadores posiblemente (muy) distintos entre sí. En este sentido:

- Proporciona *abstracciones* del hardware que lo hacen independiente de los detalles específicos de su funcionamiento. Por ejemplo, tanto los discos duros como los CD-ROM muestran su contenido como una jerarquía de directorios que contienen ficheros y, a su vez, otros directorios con ficheros y así sucesivamente.
- Facilita al programador un repertorio de *funcionalidades idénticas* que le permiten acceder a dispositivos diferentes de la misma forma. Por ejemplo, el programador usa las mismas instrucciones para escribir datos en un disco duro o un disquete.

Sin embargo, no existe un *único* SO que sea comúnmente utilizado por todos los usuarios de ordenadores. En el "mercado" podemos encontrar varios, cada uno con sus ventajas e inconvenientes. Así el profesional de la informática y/o administrador de sistemas normalmente trabajará con SSOO de la "familia" Unix. El profesional del diseño gráfico probablemente trabaje con OS X en un Apple Macintosh. En el ordenador de casa probablemente se utilice un SO de la "familia" Windows más apropiado para juegos, ofimática y aplicaciones multimedia. ¿Qué podemos encontrar?

- Los de la familia **Microsoft Windows**: Windows 95, Windows 98, Windows NT, Windows 2000, Windows XP. Orientados a usuarios domésticos y ofimática.
- Los de la familia **Unix**: Unix, BSD, FreeBSD, Linux, Solaris, AIX, HP-UX... Orientados a servidores y programación. En los últimos años, también a usuarios domésticos y ofimática.
- Los de la familia **Apple Macintosh**: OS 9, OS X (*a là Unix*). Orientados a usuarios domésticos, ofimática y diseño gráfico. La nueva versión, OS X, es un derivado de Unix. Destacan por su facilidad de uso, pero *sólo funcionan sobre ordenadores de la marca Apple*.

1.1. Clasificación de los Sistemas Operativos

Los SSOO se pueden clasificar, desde el punto de vista de la ejecución de procesos, en:

- *Monotarea*: el primer proceso que entra en ejecución monopoliza la CPU hasta su fin, es decir sólo permiten ejecutar un programa cada vez. Esto resulta muy sencillo de gestionar para el SO. Ejemplos: MS-DOS, Windows 3.X.
- *Multitarea*: todos los procesos que se ejecutan comparten la CPU, es decir varios programas se ejecutan "simultáneamente[1]. Los sistema modernos son multitarea. Exige un buen *algoritmo de planificación* y una adecuada gestión de procesos por parte del SO. En la práctica hay un límite marcado por la estructura interna del SO y la memoria principal del ordenador. Ejemplos: Linux, Windows 9X.

[1] En realidad, se ejecutan poco a poco por turnos. En cada turno un único proceso monopoliza la CPU, pero como esto ocurre a intervalos de tiempo tan pequeños (del orden de milisegundos) se crea la ilusión de simultaneidad.

Desde el punto de vista del número de usuarios que están autorizados a trabajar con el ordenador al mismo tiempo, distinguimos:

- *Monousuario*: solamente un usuario puede estar trabajando con la máquina en un momento dado. No requiere que el sistema de ficheros adopte un mecanismo de protección de datos (permisos) entre usuarios, ni se evita que el usuario pueda reconfigurar el ordenador o destruir ciertos archivos del sistema. Ejemplos: Windows 9X.
- *Multiusuario*: hay varios usuarios autorizados a conectarse a la máquina. Requiere mecanismos seguros de identificación (identificación y clave) y de protección en el sistema de ficheros. Igualmente garantiza que las acciones de un usuario cualquiera no interfiere con las de otros (un usuario no puede, por ejemplo, destruir archivos de otro, a no ser que se le dé permiso para ello).Ejemplos: Linux, Windows NT, Windows 2000...

1.2. Sistemas multiusuario

UNIX es un sistema multiusuario. Esto hace que sea necesario establecer mecanismos para identificar al usuario y para proteger unos usuarios de otros. Cada usuario de la máquina tiene un UID (user id) o *nombre de usuario* y puede pertenecer a *uno o más grupos* identificados por su GID (group id). Cada grupo reúne a usuarios con determinado perfil. Por ejemplo: profesores, alumnos, servicios, etc. Uno de los grupos a los que pertenece el usuario es el que se toma como *grupo propietario*. Son el UID y el GID del usuario que lanza un proceso los que determinan qué puede o no hacer ese proceso. Cada usuario que se da de alta en el sistema suele tener un espacio en el disco duro para almacenar sus ficheros (normalmente limitado).

¿Sabías que ...?

En realidad, se puede hacer que el proceso tenga un UID y un GID distintos de los del usuario que los lanza. Esto puede tener importantes consecuencias de seguridad.

Para iniciar una sesión de trabajo, el usuario debe identificarse ante el sistema. Si lo hace correctamente, tiene acceso a determinados recursos y puede ejecutar ciertas acciones en función de los grupos a los que está adscrito. Para identificarse, los usuarios introducen su nombre de usuario (el login) y la correspondiente palabra de paso (*password*) o *contraseña*.

Cuando creas una nueva cuenta (un nuevo usuario) como administrador, debes de introducir una contraseña para el usuario. La universidad te ha asignado ya una contraseña para acceder a un ordenador de alumnos de la UJI (`anubis`)que proporciona servicios de correo electrónico y te ofrece una cuenta para prácticas. Has de cambiar la contraseña cada cierto tiempo.

Para satisfacer tu curiosidad ...

Las palabras de paso de todos los usuarios están, junto con otras informaciones, en el fichero `/etc/passwd`, que puede ser leído por cualquiera.
Para evitar problemas de seguridad, los *passwords* se guardan codificados mediante una función que es relativamente fácil de calcular, pero cuya inversa es muy difícil de encontrar.
Para verificar el password introducido por el usuario, primero se codifica y después de compara con la versión guardada.

Has de tener cuidado a la hora de elegir la contraseña que permite el acceso a tu cuenta. Si eliges una muy "fácil" cualquiera podría entrar en el sistema haciéndose pasar por tí. Algunas indicaciones sobre cómo elegir una buena contraseña.

- No hay que elegir palabras comunes (presentes en un diccionario) ni nombres, especialmente el propio, de familiares o de personajes famosos.

- Mezclar letras y números. No utilizar *passwords* que sean totalmente numéricos y nunca el número de teléfono, DNI o similares.
- Elegir contraseñas de seis a ocho caracteres (o más si es posible).
- Utilizar contraseñas distintas para máquinas distintas.
- Hay que ser cuidadoso con los caracteres especiales, pues pueden ser representados "internamente" de formas distintas según la configuración del sistema (veáse el tema siguiente sobre codificación de la información en los ordenadores). *Es mejor que emplees sólo letras y dígitos*.
- Una idea sencilla es pensar un nombre relacionado con alguna afición o actividad que nos guste (p.e. mitología romana) y después *cambiar* todas las letras *i* por 1, las letras *o* por 0, etc. La palabra resultante la usaríamos como *password*. Por ejemplo: Mercurio (el mensajero de los dioses) da como resultado la contraseña mercur10 (no uséis esta).
- No escribáis nunca la contraseña en un papel.

Pros y contras del `/etc/passwd`

Al guardarse las palabras de paso codificadas, se evita el riesgo de que alguien las encuentre "accidentalmente".
Al tener permisos de lectura para todo el mundo, permite mayor flexibilidad a los programas (p.e. un programa salvapantallas puede preguntar de nuevo la palabra de paso al usuario, para asegurarse de que no se ha ido a tomarse café y otro ha ocupado su lugar).
Sin embargo, si las palabras de paso elegidas son poco seguras, es un sistema vulnerable ante el "ataque con diccionario", se compara el fichero con una lista de palabras comunes ya codificadas.
Para solucionarlo se han introducido las llamadas *shadow password*. La idea es guardar las palabras de paso en un fichero separado y protegido contra lectura.

En los sistemas de la "familia" Unix, existe un número de UID especial, el 0. Éste corresponde a un usuario especial: el *superusuario*, que generalmente recibe el nombre de *root*. Ese usuario puede "saltarse" los permisos de los ficheros; puede leer y escribir (es decir, modificar o borrar) cualquier fichero. Es el encargado de la (buena) gestión de la máquina a su cargo. Tiene poder para detener cualquier proceso del sistema. Es importante que su contraseña no sea conocida más que por la persona a cargo del sistema.

Por tanto, debes ir con cuidado con la información que consideres privada o confidencial, ya que el superusuario siempre puede acceder a ella. Aunque habría que resaltar que, *por cuestiones de ética profesional*, el superusuario *no debe* violar la confidencialidad de los datos de los usuarios[2]; más bien al contrario. Cualquier actitud abusiva al respecto podría ser incluso objeto de denuncia.

1.3. Plataformas

Como vimos en el tema anterior, el SO se encarga de la gestión de procesos, siendo, por tanto, el responsable de la ejecución de los programas. La ejecución de programas está, pues, *íntimamente* ligada al SO. De hecho el SO puede ejecutar los programas porque en los ficheros que los contienen existe información que le indica cómo hacerlo. Esa información es particular para cada SO.

Esto quiere decir que los programas se compilan también para ser ejecutados en un SO concreto. Sin embargo, igualmente en el tema anterior, habíamos visto que los programas se compilaban para una *arquitectura* específica. ¿Qué quiere decir esto?

Ambas cosas son ciertas. Los programas se compilan para un SO concreto que se ejecuta en una arquitectura específica (tipo de ordenador). El conjunto arquitectura + SO conforma lo que comúnmente se denomina *plataforma*. Así pues los programas se compilan para ser ejecutados en una plataforma determinada. Así, por ejemplo,

[2] Salvo por imperativos legales o razones de "peso" que tengan que ver con la correcta administración del sistema a su cargo.

un programa compilado para PC bajo Windows no funcionará en un PC con Linux y viceversa. Necesitamos coger el código fuente del programa y compilarlo en la plataforma correspondiente[3].

Cada sistema operativo se acompaña, por tanto, de un conjunto de utilidades característico (programas compiladas para ese SO y el tipo de ordenador sobre el que van a funcionar). Así, en muchos Unix encontrarás la misma colección de utilidades para gestionar ficheros, editar texto, etc. El sistema operativo Linux, por ejemplo, se acompaña de una serie de utilidades construidas por GNU (un grupo de programación de software libre), así que hablamos de la plataforma GNU/Linux. Estas utilidades han sido compiladas para las distintas arquitecturas en las que se puede instalar Linux (x86, Alpha, Sparc, . . .).

1.4. Entornos de usuario

Cada Sistema Operativo ofrece uno o más *entornos de usuario* (o entornos de trabajo). El entorno de usuario proporciona *herramientas* para acceder a la información, ejecutar aplicaciones, configurar el ordenador, etc. Los entornos de usuario son la interfaz que sirve para establecer la "comunicación" entre el usuario y el SO. Tipos de entornos que podemos encontrarnos:

- De *línea de órdenes*: las órdenes se expresan como texto.

  Ejemplo: Los sistemas Unix proporcionan un rico juego de órdenes y uno o más intérpretes de órdenes o *shells* (programas que leen órdenes de teclado y controlan la ejecución de las órdenes).

- De *interfaz gráfico*: las órdenes se expresan accionando sobre iconos visuales, botones y menús con un ratón.

  Ejemplos: Microsoft Windows y Apple Macintosh presentan sendos entornos de interfaz gráfico. Linux permite elegir entre varios entornos de usuario (aunque hay dos más avanzados y populares: Gnome y KDE).

Acerca de los intérpretes de órdenes

La interacción de los usuarios con el sistema operativo se hace a través de los intérpretes de órdenes (*shells*).
Como veremos, las órdenes suelen ser poco amigables (modo texto en consola). Se suele intentar evitar la redundancia y hacer órdenes especializadas. En la mayoría de los casos, el éxito se traduce en el más absoluto silencio.
Los *shells* incluyen sus propios "lenguajes de programación" que, además, suelen ser muy potentes. Podemos elegir entre diversas opciones: por ejemplo, `/bin/sh`, `/bin/ksh`, `/bin/csh`, `/bin/bash`. El superusuario asigna el programa intérprete de órdenes en el momento de dar de alta al usuario en el sistema.

Es importante que observes que dos ordenadores diferentes con un mismo entorno de usuario se manejan del mismo modo. Por ejemplo, el entorno KDE funciona igual ejecutado en un PC que en una estación de trabajo SUN. Antes de estudiar los entornos de usuario, hemos de introducir conceptos como el de sistema de ficheros.

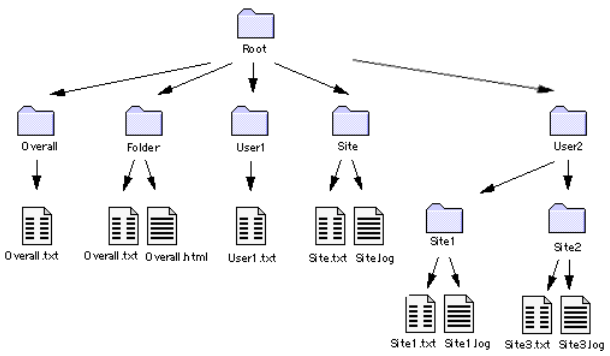


Figura 2: Organización jerárquica de un sistema de ficheros.

2. Sistemas de ficheros

El sistema de ficheros es la parte del SO que se encarga de la representación *abstracta* de los dispositivos de almacenamiento secundario que pueden conectarse a un ordenador: discos duros, otros discos externos o internos (p.e. SCSI o USB), disquetes, CD-ROM, DVD, etc. Desde este punto de vista el SO se encarga fundamentalmente de:

[3] Puede que incluso tengamos que hacer alguna modificación en el código original.

- La gestión de los ficheros, esto es de la correspondencia entre colecciones de información relacionadas entre sí y agrupadas bajo un nombre *único* dado por su creador (una imagen, una pista de música, un documento, una hoja de cálculo, etc.) con sectores físicos.
- La organización *jerárquica* de los ficheros agrupados en colecciones de carpetas o directorios, posiblemente distribuidos en dispositivos diferentes.

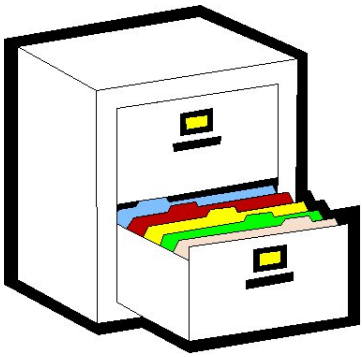


Figura 3: Visión del sistema de ficheros como un archivador.

Por consiguiente, el sistema de ficheros proporciona una visión de la información almacenada en unidades de almacenamiento como *ficheros agrupados en una jerarquía de directorios*. Se pretende establecer una analogía con el archivador de una oficina.

- El *ordenador* es como un gran *archivador*.
- Cada dispositivo o *unidad* (disco duro, disquette, CD-ROM, etc.) es un *cajón* del archivador.
- Cada *directorio* es una *carpeta*. Dentro de una carpeta podemos almacenar subcarpetas y, del mismo modo, un directorio puede contener otro directorios.
- Cada *fichero* es un *documento*.

¿Sabías que . . . ?

Para poder llevar a cabo eficazmente las tareas de gestión del sistema de ficheros, el SO define internamente ciertas *unidades lógicas* de información que permiten realizar una representación básica de las *unidades físicas* que conforman los dispositivos
Por ejemplo, los sectores en que divide la cinta magnética del disco duro y que sirven para organizar la información en él almacenada.
La operación consistente en determinar la cantidad y la representación inicial de las unidades lógicas necesarias para establecer la correspondencia con las unidades físicas de un dispositivo y almacenarlas en el mismo se denomina comúnmente *formatear* o *dar formato*
Formatear supone la “construcción” del sistema de ficheros en el dispositivo. Esta operación normalmente destruye los datos que pudiera haber almacenados en la unidad.
Cualquier unidad de almacenamiento secundario que queramos conectar a nuestro ordenador y a la que deseemos acceder mediante las herramientas proporcionas por el sistema de ficheros de un determinado SO, habrá de estar *formateada* de manera que pueda ser accesible para dicho SO.
Pensarás que hay excepciones a esto que hemos dicho: por ejemplo el CD-ROM. Piensa no obstante que, como veremos más adelante, los CD-ROM vienen ya “formateados” con un sistema de ficheros específico que se crea en el momento de grabarlos y que “entienden” y reconocen todos los SSOO.

En los entornos gráficos de usuario los *ficheros* se muestran con diferentes iconos en función del tipo de contenido. Distinguimos, fundamentalmente, dos tipos de fichero:

- *Aplicaciones* o programas: son ficheros ejecutables.

- *Documentos*.

Los *directorios* se muestran como carpetas:

Las *unidades* o dispositivos conectados al ordenador se representan en Windows con iconos especiales. Linux puede utilizar indistintamente iconos especiales o carpetas para representar los dispositivos.

El *ordenador* se representa con un icono que representa un PC (Windows) o como una carpeta (Linux):

2.1. Ficheros y directorios

Las características o propiedades que tiene un *fichero* son:

- *Nombre*: secuencia de caracteres que lo identifica en un directorio.

  Se suele descomponer en dos partes:

  - *Extensión*: los caracteres que siguen al último punto del nombre. Se usan para indicar el tipo del archivo. El icono del fichero se suele determinar a partir de la extensión. ¡No es obligado utilizar una extensión! Sí es cierto que se acostumbra a usarla y que muchos entornos gráficos la utilizan para representar el fichero con un determinado icono e incluso para "asociar" ficheros con programas (los programas que se ejecutarán al abrir el fichero).
  - Y el *nombre* propiamente dicho.

  Ejemplo: `fichero.txt` es el nombre de un archivo con extensión `txt`, indicando que es un fichero de texto. Ten en cuenta que, en sistemas antiguos como MS-DOS, se imponían ciertas restricciones a los nombres de archivo: el nombre propiamente dicho no podía superar 8 caracteres y la extensión 3.

- *Atributos*: una serie de valores que indican cosas tales como:

  Ubicación, dónde está situado en el disco (*path* o ruta de acceso, como veremos posteriormente).

  Tamaño, el tamaño actual (en alguna unidad adecuada) y (posiblemente) el máximo tamaño permitido.

  Protección, información para controlar quién y cómo puede utilizar el fichero; esto es los permisos de seguridad (o utilización) que indican si el fichero es legible, escribible, ejecutable, etc.

  Contador de uso, numero de procesos (programas) que están usando el fichero.

  Hora y fecha de la última modificación efectuada.

  ...

- *Contenido*: la información que hay en su "interior" (texto, dibujos, música, etc.).

Las características o propiedades que tienen los *directorios* son:

- *Nombre*: normalmente sin extensión. Nuevamente indicar que ésta es una costumbre, *no una obligación*. Puedo usar un nombre con extensión para identificar un directorio.
- *Atributos*:
  - fecha de creación y fecha de la última modificación,

  - permisos de seguridad (“listable”, escribible, con contenido accesible, etc.),
  - ...

- *Contenido*: ¿Qué contiene un directorio? Pues otros directorios y/o ficheros.

2.2. Acciones

Los ficheros son los elementos básicos para la organización lógica de la información en el sistema de ficheros. Las carpetas o directorios son un mecanismo para la organización jerárquica de la información[4]. Las carpetas contienen ficheros y pueden contener otras carpetas. Las operaciones básicas que podemos realizar para administrar carpetas y ficheros son:

- Crearlos.
- Copiarlos.
- Borrarlos.
- “Moverlos”, esto es cambiarlos de ubicación. Esto incluye también cambiarles el nombre (renombrarlos).
- Comprimirlos.
- Modificar su contenido.
- Ver (listar) su contenido.
- Empaquetarlos.
- ...

La forma en que se realizan estas acciones depende del entorno de usuario. Por ejemplo: borrar un fichero en un entorno gráfico suele consistir en desplazar el icono que lo representa sobre el icono que representa una papelera[5]:

Sin embargo, borrar un fichero en un intérprete de órdenes Unix consiste en ejecutar la orden `rm` (*remove*): `rm nombrefichero`

Llegado este punto, uno puede preguntarse *“¿Por qué aprender a usar un intérprete de órdenes si existen sistemas con interfaz gráfico?”*. Para poder responder a esta pregunta, debes de tener en cuenta lo siguiente:

- *Algunos sistemas profesionales no tienen un interfaz gráfico* o éste no es estándar cuando, por contra, sí presentan juegos de órdenes estándar (o muy similares). Cuando los uses, necesitarás saber manejarte con el intérprete de órdenes.
- Los intérpretes de órdenes permiten *automatizar ciertas tareas* que son tediosas y propensas a la comisión de errores con interfaces gráficos. Ejemplos: elimina todos los ficheros del ordenador de tal tipo, haz una copia de seguridad de los ficheros modificados a partir de tal fecha, elimina los ficheros de tal directorio que no se han modificado en 15 días, etc.
- Los intérpretes de órdenes suelen incluir un potente lenguaje de programación (normalmente similar a C en cuanto a sintaxis y capacidad expresiva se refiere) que permite poder programar de forma automática tareas extraordinariamente complejas. Los programas realizados de esta manera se denominan *scripts*. Por ejemplo, muchos sistemas de copia de seguridad (*backup*) consisten en ejecutar una serie de *scripts* desde un intérprete de órdenes.

3. El entorno de trabajo Unix

[4] Este tipo de organización es la más común.

[5] Normalmente, después hay que vaciar la papelera para borrarlo *definitivamente*. Mientras esté en la papelera y ésta no se vacíe, el fichero se puede *recuperar*.

¿Sabías que . . . ?

UNIX es un sistema multiusuario y multitarea que comenzó a desarrollarse a finales de los 60, tomó "forma" durante los 70 y alcanzó su madurez y pleno desarrollo durante la década de los 80.
UNIX ha sido considerado tradicionalmente el SO de los grandes ordenadores: máquinas de cálculo intensivo, *mainframes* o estaciones de trabajo habitualmente presentes en los centros de proceso de datos o en los centros de cálculo de grandes empresas, universidades u organismos internacionales.
En la actualidad también disponemos de "versiones" UNIX para PC (p.e. Linux y FreeBSD), algo ciertamente demandado dados los grandes avances realizados en los últimos años en este tipo de arquitectura.

En las sesiones de prácticas aprenderás a utilizar los entornos gráficos (los "escritorios") de Microsoft Windows y Linux. Bueno, en el caso de Linux solamente estudiaremos el entorno KDE (K Desktop Environment). Existen otros, como GNOME.

En este tema aprenderemos fundamentalmente a usar las órdenes básicas de un entorno de línea de órdenes: el entorno de trabajo *Unix (Linux)*. Iremos introduciremos nuevos conceptos del sistema de ficheros conforme presentemos el entorno de trabajo Unix.

Hitos en el desarrollo de Unix

1965: Se diseña un SO multiusuario muy ambicioso: *MULTICS* (MULTiplexed Information and Computing Service). Participan, entre otros, MIT, General Electric y Bell Labs de AT&T. Se pretendía dar capacidad de cálculo siguiendo el modelo de las compañías eléctricas: gran número de usuarios se conectarían a un ordenador central (aproximadamente de la potencia de un PC-AT).
1968: El proyecto fracasa. Bell Labs se retira de MULTICS y uno de sus empleados, Ken Thompson (involucrado en el proyecto), empezó a escribir para un PDP-7 una versión muy simplificada de MULTICS, escrita en lenguaje ensamblador.
1969: Dennis Ritchie y Rudd Canaday se unen a Thompson en su proyecto. Esta versión funcionaba bien y uno de los colegas de Thompson la llamó *UNICS* (UNiplexed Information and Computing Service).
1970–1974: Brian Kernighan lo bautiza *UNIX*. Se reescribe en B, un lenguaje inventado por Thompson, pero con ciertas carencias. Ritchie diseño a partir de él C. Él y Thompson reescribieron UNIX en C para PDP-11, que permitía tener varios usuarios simultáneamente al proteger la memoria por hardware. Además se incluyen innovadoras herramientas de procesamiento de texto.
1974–1979: Bell Labs publica la descripción de UNIX: SO portable, escrito en lenguaje de alto nivel (C), multiusuario, multitarea, sistema de archivos jerárquico y con editor, compilador de C y herramientas de procesamiento de texto. Las universidades manifiestan interés por compartirlo. Bell lo distribuye *con fuentes* de manera *abierta y libre* a instituciones académicas y científicas. Sólo obliga a reportar investigaciones y mejoras. Al utilizarse en las universidades más importantes de Europa y EEUU y haber cada vez más gente trabajando en su desarrollo "progresa" muy rápidamente. Aparece la primera versión "estándar": la versión 6.
1979: Se licencia UNIX versión 7, la cual se populariza enormemente como plataforma para experimentar y desarrollar tecnología, probar algoritmos, protocolos de comunicación, lenguajes de programación y gestores de bases de datos. Por entonces ya estaba ampliamente distribuido por gran cantidad de estaciones de trabajo y muchos estudiantes se habían formado a partir de UNIX.

continúa en la página siguiente . . .

. . . viene de la página anterior

1980–1983: La universidad de Berkeley proporciona UNIX BSD 4.0 al departamento de defensa de EEUU (BSD es la versión de UNIX de Berkeley). En ARPANET comienza a utilizarse TCP/IP bajo UNIX BSD 4.0. UNIX System III constituye el primer intento de unificación de versiones.
1980–1983: Aparecen versiones comerciales: Xenix, UNIX para arquitectura IBM S/370. Otro estándar: System V y sus diversas *releases*. SUN, HP y Silicon Graphics, entre otros, lanzan estaciones de trabajo bajo UNIX dedicadas al procesamiento gráfico profesional y CAD.
1984: UNIX System V Release 2 incorpora los últimos avances tecnológicos en manipulación y administración de recursos, protocolos de comunicación, lenguajes, bases de datos, procesamiento distribuido y paralelismo. El proyecto de interfaz gráfica de usuario Athena X-Window System se desarrolla e implementa bajo UNIX.
1985–1986: Xenix System V para PC (80286 y 80386). SUN implementa NFS (Network File System) sobre TCP/IP para compartir volúmenes (discos) en red y hace pública la definición de los protocolos necesarios. La red de la fundación nacional de ciencia de EEUU y Arpanet utilizan TCP/IP a plenitud: nace *Internet*. Se publican críticas a UNIX para tratar de frenar su avance. TCP/IP se convierte en el protocolo estándar de comunicación en todo tipo de redes.
1987: Ante las diferencias entre las distintas versiones, la IEEE definió el estándar POSIX, que intenta ser una combinación entre System V Release 3 y BSD 4.3.
1988: *X Window System* se proclama sistema estándar para el desarrollo de interfaces y aplicaciones gráficas. Andrew Tanenbaum publica un libro sobre sistemas operativos acompañado de una versión de UNIX basado en POSIX, escrita por él y llamada *Minix*.
1989: UNIX e Internet caminan de la mano: TCP/IP, NFS, e-mail, . . .
1991: Un estudiante universitario en Finlandia llamado Linus Torvalds está trabajando en modificaciones a Minix. Solicitando información por correo electrónico en Internet sobre la definición y contenido del estándar POSIX, recoge apoyo de curiosos e interesados en su proyecto: *Linux*, una versión de UNIX POSIX de libre distribución. Las consideraciones de diseño del núcleo del sistema Linux son portabilidad y eficiencia. Internet está gestando la WWW para todo el mundo (UNIX).

3.1. Directorios estándar

En Unix la jerarquía del sistema de ficheros tiene un nodo principal: el directorio raíz. El directorio raíz no tiene nombre y nos referimos a él con una barra (`/`).

Los SSOO de la familia UNIX siguen, normalmente, ciertas “convenciones” en los nombres de algunas de las carpetas donde se localizan los ficheros de configuración del sistema, programas ejecutables, librerías, documentación, ficheros de configuración de aplicaciones, etc. Así:

/bin contiene los ejecutables básicos del sistema (órdenes para crear, copiar, mover, . . . ficheros y carpetas, etc.).

/dev contiene los dispositivos susceptibles de ser conectados a nuestro ordenador representados en forma de archivos especiales (en Unix, el ratón, cada disco duro, etc. llevan asociados sendos ficheros).

Si quieres saber más . . .

Una de las funciones principales de un SO consiste en controlar todos los dispositivos de entrada/salida de la computadora: debe emitir órdenes a los dispositivos, capturar interrupciones y manejar errores.
Pero también debe proporcionar una *interfaz* entre los dispositivos y el resto del sistema que sea simple y fácil de utilizar. Además la interfaz debe ser la misma para todos los dispositivos procurando conseguir la *independencia* de los mismos.
En general, los SSOO de la familia UNIX “unifican” (en cierta manera) la gestión de la entrada y salida de datos con la del sistema de ficheros. Así, los dispositivos se representan mediante ficheros *especiales* en `/dev`.

continúa en la página siguiente . . .

. . . viene de la página anterior

Esto permite disponer de una interfaz única, independiente del dispositivo y una *facilidad* para los programadores: acceder a los dispositivos requiere las mismas llamadas al sistema que se usan para acceder, en general, a ficheros, ya que no hay que tener mecanismos especiales de escritura o lectura dependiendo del dispositivo, todo se trata como si fuese un fichero.

/etc contiene ficheros para la configuración de aplicaciones, programas del sistema y servicios de la máquina (p.e. Internet) entre otras cosas. Las aplicaciones o servicios que requieran más de un fichero para su configuración, crean una subcarpeta para agruparlos.

home directorio en el que se disponen los directorios de cada usuario del ordenador.

lib lugar en el que se almacenan ficheros auxiliares del ordenador (las denominadas librerías o bibliotecas).

/mnt carpeta desde la cual se "montan" los dispositivos a cuyo sistema de ficheros se quiere acceder. Si se quiere poder montar varios dispositivos, podemos crear varias carpetas al efecto (lo detallamos más adelante).

/tmp directorio "temporal" del sistema. Es una carpeta que pueden usar los programas para crear ficheros dinámicamente (operaciones intermedias, etc.); esto es ficheros auxiliares que pueden eliminarse sin peligro cuando apagamos el ordenador.

/usr contiene todo lo necesario para ejecutar programas de aplicación (o de usuario). Documentación, ejecutables, librerías, etc.

/var contiene ficheros de registro (bitácora), información del sistema, el correo de los usuarios, las "colas" de las impresoras, etc.

3.2. Rutas

Mediante el uso de carpetas la información se organiza en forma de *árbol*, de ahí la denominación "organización jerárquica". En este tipo de sistemas de ficheros:

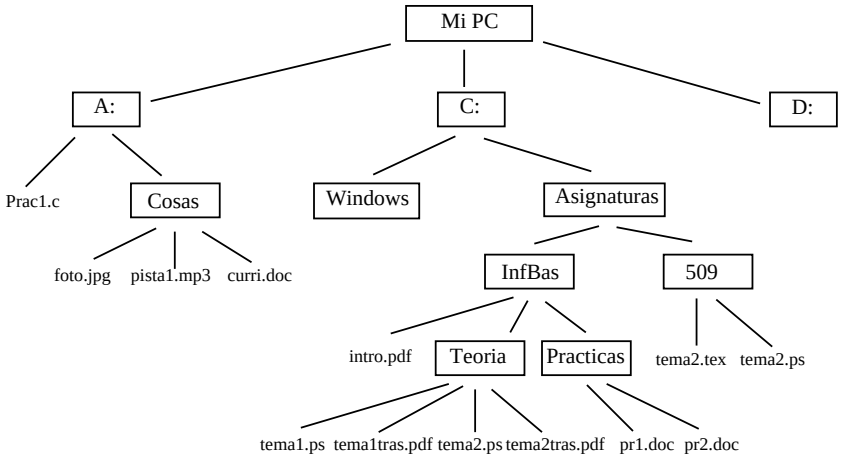
- un nodo interno es una carpeta,
- un nodo externo u hoja del árbol es una carpeta *vacía* o un fichero,
- una rama es un apunte (anotación) en la carpeta que hace referencia a:
  - otro directorio (subdirectorio), o
  - a un fichero.
- la raíz del árbol es la carpeta principal o directorio *raíz*.

Si llamamos *nombre simple* al nombre que tiene un fichero dentro de la carpeta que lo contiene, entonces llamaremos:

- *nombre absoluto* (ruta) de un fichero a su nombre simple precedido de la secuencia de carpetas que hay que abrir desde la raíz para encontrarlo.
- *nombre relativo* a una carpeta cualquiera X de un fichero, a su nombre simple precedido de la secuencia de carpetas que llevan al fichero *desde* X.

Dentro del sistema de ficheros, cada fichero debe ser referenciado (nombrado) de forma *no ambigua*. Esto quiere decir que no podrá haber dos ficheros con el *mismo nombre absoluto* en el sistema de ficheros.

También dentro del sistema de ficheros, el usuario siempre está trabajando en una carpeta, a la cual se le llama *carpeta o directorio actual*. El nombre relativo de un fichero siempre se interpreta como relativo al directorio actual. Veamos un ejemplo de un sistema de ficheros en Windows:



En la jerarquía de la figura anterior, Mi PC\C:\Asignaturas\509 es un nombre absoluto de carpeta y Mi PC\C:\Asignaturas\InfBas\intro.pdf es un nombre absoluto de fichero.

InfBas\Teoria es un nombre de carpeta relativo a la carpeta Asignaturas, e InfBas\Practicas\pr1.doc es un nombre de fichero igualmente relativo a la carpeta Asignaturas.

Si el directorio actual es Mi PC\C:\Asignaturas\InfBas, entonces Teoria\tema2.ps y Practicas\pr2.doc son nombres relativos al directorio actual, pero 509\tema3.tex no lo es. Existen nombres simples iguales en el ejemplo (tema2.ps), pero pueden existir así en el sistema porque tienen distinto nombre absoluto.

En el ejemplo anterior , el directorio raíz es Mi PC, y así se ha considerado en los ejemplos de nombres absolutos, situándolo como primer directorio de todos los caminos. Sin embargo, en la práctica Mi PC no aparece como primer directorio de ninguno de los caminos, siendo el directorio raíz en realidad el *nombre de la unidad* de almacenamiento secundario.

La razón de ello es que cada unidad de almacenamiento en Windows define su *propio* sistema jerárquico de ficheros y el nombre de la unidad es la raíz real de cada sistema. Mi PC es una raíz virtual que se presenta en el escritorio para aparentar un único sistema de ficheros global. Esto es, el sistema de ficheros *no* es único en Windows.

Por consiguiente, en Windows los nombres absolutos comenzarán desde el nombre de la unidad de almacenamiento secundario en el que se encuentren los respectivos elementos (carpetas o ficheros) que referencian.

En el caso de Unix, el concepto de ruta es el mismo que para Windows. Simplemente cambia el separador de carpetas; ya no es \, sino /. Los sistemas de directorios de UNIX y de MSDOS/Windows son similares: en ambos se permiten rutas absolutas (comienzan con /, el directorio raíz, que es el primero que existe en la jerarquía) y relativas (*nunca* comienzan con /) y se tienen las entradas `.` y `..` en cada directorio para representar, respectivamente, al directorio actual de trabajo y al directorio que lo contiene (el "padre"). Sin embargo, como veremos más adelante, hay importantes diferencias entre los dos sistemas de ficheros.

Cuando te conectas a un ordenador Unix has de identificarte proporcionando un nombre de usuario (`login`) y una contraseña secreta (`password`). Si se introducen ambos correctamente, se inicia una sesión de trabajo. El sistema nos pide una orden mostrando un *prompt*, que aquí representaremos con un dólar (`$`). Cada usuario tiene un directorio personal (normalmente) en `/home/` cuyo nombre es igual al nombre del usuario.

Supongamos que tu nombre de usuario es `juan`: tu directorio personal es `/home/juan`. Al iniciar una sesión de trabajo "entras" directamente a tu directorio personal. Decimos que tu directorio personal es, ese momento, tu *directorio activo*. El directorio activo permite evitar nombres de ruta largos al referirse a un fichero: nos podemos referir al fichero `/home/juan/datos.txt` por su nombre, `datos.txt`, si el directorio activo es `/home/juan/`.

Recuerda que si una ruta no empieza por `/` decimos que es una *ruta relativa*. En caso contrario es una *ruta absoluta*. Una ruta relativa añade implícitamente la ruta del directorio activo para formar así la ruta completa de un fichero.

3.3. Cambio de directorio

La orden `cd` (*change directory*) permite *situarse en el directorio* especificado, es decir cambiar el directorio activo. El directorio se puede especificar mediante la *ruta completa* o *relativa* al directorio de trabajo actual. Si utilizamos `cd ..` volvemos al directorio que incluye al que estamos actualmente situados (directorio "padre"). Por ejemplo, podemos cambiar el directorio activo a `/home` con:

```
$ cd /home
```

o al de otro usuario con:

```
$ cd /home/ana
```

Desde nuestro directorio personal podemos cambiar a `/home` con:

```
$ cd ..
```

Desde nuestro directorio personal podemos cambiar a `/home/ana` con:

```
$ cd ../ana
```

Desde nuestro directorio personal podemos cambiar a la raíz `/` con:

```
$ cd ../..
```

Finalmente, si ejecutamos `cd` sin una ruta, nos devuelve al directorio personal.

3.4. ¿Cuál es el directorio activo?

La orden `pwd` (*path working directory*) muestra por pantalla el directorio activo actual (ruta completa).:

```
$ pwd
/home/juan
$ cd ..
$ pwd
/home
$ cd
$ pwd
/home/juan
```

3.5. Listar el contenido de un directorio

La orden `ls` (*list*) permite listar el contenido de un directorio. Si no hay presente ningún argumento (aparte de las opciones que se pueden usar, como veremos a continuación) se supone como argumento predeterminado el directorio activo actual (“.”).

```
$ ls /
bin    dev    etc    home   lib    mnt  usr    tmp    var
$ cd /
$ ls
bin    dev    etc    home   lib    mnt  usr    tmp    var
```

La opción `-l` (abreviatura de *long*) permite mostrar la información completa (“listado detallado”) de lo que contiene el directorio.

```
$ ls -l /
drwxr-xr-x    2 root     root         2144 ago  5 09:20 bin
drwxr-xr-x   27 root     root        75264 sep 23 07:45 dev
drwxr-xr-x   53 root     root         6760 sep 23 07:45 etc
drwxr-xr-x    4 root     root         4096 mar 21  2002 home
drwxr-xr-x    6 root     root         2232 ago  5 09:21 lib
drwxr-xr-x    2 root     root           48 mar 21  2002 mnt
drwxrwxrwt   33 root     root         3528 sep 23 13:19 tmp
drwxr-xr-x   13 root     root          376 abr 25 21:32 usr
drwxr-xr-x   18 root     root          456 jun 27 09:53 var
```

Las órdenes Unix pueden modificar su comportamiento mediante *opciones*. Las opciones empiezan por un guión (`-`). El listado detallado da mucha información sobre cada fichero:

- los *permisos de seguridad* (el texto que presenta este aspecto: `drwxr-xr-x`),

- número de *enlaces* (no lo estudiaremos por ahora),
- nombre del *propietario* (en el ejemplo, el usuario `root`),
- nombre del *grupo* propietario (cada usuario pertenece a uno o más grupos; en el ejemplo, el grupo se llama igual que el usuario),
- el *tamaño* expresado en bytes (en el caso de ficheros de texto, un byte es un carácter; en el caso de directorio tiene un significado especial que no detallaremos),
- la fecha de *última modificación*,
- el *nombre* del fichero o directorio.

Un par de opciones interesantes . . .

La opción `-R` procede de forma *recursiva* (esto es, subdirectorio por subdirectorio hasta que no quede ninguno por explorar) a listar el contenido de los directorios y de sus subdirectorios.
La opción `-a` permite mostrar los ficheros ocultos (aquéllos cuyo nombre comienza por ".").

3.6. Creación de ficheros

Hay muchos modos de crear ficheros. Lo más normal es que los crees como resultado de usar un programa:

- un editor de textos permite crear y modificar ficheros con texto,
- un editor gráfico permite crear y modificar ficheros con imágenes,
- un conversor MP3 permite crear un fichero de música comprimida (MP3) a partir de un fichero de música sin comprimir (por ejemplo, en formato WAV),
- …

De momento crearemos ficheros *vacíos* con una orden Unix denominada `touch`.

```
$ touch mifichero
$ ls -l
-rw-r--r--    1 juan  migrupo          0 sep 24 13:58 mifichero
```

Observa que el fichero recién creado no ocupa espacio (0 bytes): está vacío.

3.7. Creación de directorios

La orden `mkdir` (*make directoty*) crea un nuevo directorio con el nombre que indiquemos. Si el nombre es *relativo*, se crea a partir del directorio activo actual.

La primera orden crea un directorio llamado `apuntes` en el directorio activo.

```
$ mkdir apuntes
$ ls -l
drwxr-xr-x    2 juan  migrupo       4096 sep 25 08:46 apuntes
-rw-r--r--    1 juan  migrupo          0 sep 24 13:58 mifichero
$ cd apuntes
$ pwd
/home/juan/apuntes
```

Y esta otra crea un directorio llamado `juan` en `/tmp`:

```
$ mkdir /tmp/juan
```

3.8. Usuario y grupo propietarios

Cada fichero y directorio del sistema pertenecen a un *usuario* y *grupo* determinados: el usuario y el grupo propietarios. Éstos, junto con los permisos,determinan:

- *quién puede leer* el contenido de un fichero o directorio,
- *quién puede modificar* el contenido de un fichero o directorio,
- *quién puede ejecutar* un fichero de programa o acceder al contenido de un directorio.

Se puede cambiar el propietario y grupo de un fichero si se tiene permiso para ello.

- La orden `chown` (*change owner*) cambia el propietario de un fichero.

```
$ chown ana fichero.txt
```

- La orden `chgrp` (*change group*) cambia el grupo al que pertenece un fichero.

```
$ chgrp alumnos fichero.txt
```

3.9. Permisos

Acabamos de aprender que todos los ficheros y directorios en Unix tienen un propietario y un grupo. Asociados a éstos tienen unos permisos, que determinan qué se puede hacer. Los *ficheros* tienen tres tipos de permiso diferentes, abreviados con una letra:

- `r`: permiso de *lectura* (por *read*). ¿Se puede ver el contenido del fichero?
- `w`: permiso de *escritura* (por *write*). ¿Se puede modificar el contenido del fichero?
- `x`: permiso de *ejecución* (por *execute*). ¿Se puede ejecutar el fichero? (Sólo tiene sentido si el fichero es un programa ejecutable.)

Los permisos se representan mediante 9 bits, agrupados en tres grupos de tres bits de la siguiente forma:

- Cada grupo fija los permisos de lectura (`r`), escritura (`w`) y ejecución (`x`).
- El primer grupo de 3 bits los fija para el *usuario* propietario del fichero: `u` (por *user*).
- El segundo grupo de 3 bits, para todos los usuarios adscritos al *grupo* propietario del fichero: `g` (por *group*).
- El tercer grupo de 3 bits, para el resto de usuarios (*otros*) registrados en el sistema: `o` (por *others*).

Cuando solicitamos el listado largo (`ls -l`) de un directorio, se muestra a mano izquierda una serie de caracteres con este aspecto.

```
-  rwx  rwx  rwx
      ↑    ↑    ↑
   usuario grupo otros
```

Observa que el primer carácter es un guión (`-`) para ficheros y una `d` para directorios. Cada grupo de tres caracteres indica si un permiso está concedido (aparece la letra `r`, `w` o `x`) o no (aparece un guión en el lugar correspondiente) al usuario, grupo y otros, respectivamente. Algunos ejemplos para que te hagas una idea:

- Permisos de lectura escritura para usuario y grupo, pero no para otros, en un fichero: `-rw-rw---`
- Permisos de lectura y escritura para usuario y sólo lectura para el grupo y otros, en un fichero: `-rw-r-r-`
- Permiso de ejecución y lectura para todos: `-r-xr-xr-x`
- Todos pueden hacer todo: `rwxrwxrwx`
- El propietario y los usuarios del grupo pueden hacerlo todo, los otros nada: `rwxrwx--`

- El propietario puede hacerlo todo, los demás sólo pueden leer y ejecutar: `rwxr-xr-x`
- Puede que aquí nos hayamos equivocado con los permisos, ¿no crees?: `----rwx`

Cuando se trata de un *directorio* (la primera letra es una `d`), los permisos se interpretan con ciertos matices:

- `r`: permiso de *listado*. ¿Se puede listar el contenido del directorio?
- `w`: permiso de *modificación*. ¿Se puede añadir (creándolo, por ejemplo) o eliminar un fichero del directorio?
- `x`: permiso de *lectura de ficheros*. ¿Se puede leer el contenido de un *fichero del directorio* si, además, tenemos permiso de lectura sobre él?

La orden `chmod` (*change modifiers*) permite modificar los permisos de un fichero. Se puede aprender cómo se usa viendo unos pocos ejemplos. Imagina que tenemos un fichero llamado `mio` cuya información de permisos es `-rw-r-r-`:

- Dar permiso de lectura, escritura y ejecución a los miembros del grupo:

  ```
  $ chmod g=rwx mio
  ```

  Queda así `-rw-rwxr-`.
- Dar permiso de ejecución al usuario:

  ```
  $ chmod u+x mio
  ```

  Queda así `-rwxrwxr-`.
- Eliminar el permiso de lectura a otros:

  ```
  $ chmod o-r mio
  ```

  Queda así `-rwxrwx--`.
- Eliminar permiso de escritura a todos:

  ```
  $ chmod a-w mio
  ```

  (Nota: `a` abrevia *all*). Queda así `-r-xr-x--`.
- Dar permiso de escritura al usuario y al grupo:

  ```
  $ chmod ug+w mio
  ```

  Queda así `-rwxrwx--`.
- Podemos combinar acciones distintas para varios grupos separando por comas. Así, quitar permiso de ejecución para usuario, quitar permisos de escritura y ejecución para grupo y añadir permiso de lectura para otros:

  ```
  $ chmod u-x,g-wx,o+r mio
  Queda así \texttt{-rw-r--r--}
  ```

3.10. Más sobre rutas

Es tan frecuente referirse al directorio personal propio que disponemos de una abreviatura para él: `~`. Por ejemplo:

```
$ cd ~
$ pwd
/home/juan
$ cd ~/apuntes
$ pwd
/home/juan/apuntes
```

En ocasiones necesitarás referirte al directorio activo. Recuerda que puedes hacerlo usando un punto.

```
$ cd ~
$ ls .
apuntes   mifichero
$ cd ./apuntes
$ pwd
/home/juan/apuntes
```

3.11. Copia de ficheros

La orden `cp` (*copy*) se utiliza para copiar ficheros (y carpetas enteras con todo su contenido). El número de opciones que presenta es muy elevado, así como las diversas formas en las que puede ser utilizada. Nosotros vamos a centrarnos en las operaciones de uso más frecuente. Se necesitan dos datos: la ruta del fichero original y la ruta de la copia.

Para copiar un fichero a otro: `cp fichero1 fichero2`. Esta orden crea una copia del fichero referenciado por la *ruta* `fichero1` al fichero referenciado por la *ruta* `fichero2`. En ambos casos se pueden usar *rutas absolutas* o *relativas* al directorio activo. Si `fichero2` no existe, *se crea al hacer la copia* y si existe el fichero, el contenido anterior se *destruye* al hacerla.

```
$ cp mifichero otrofichero
$ ls
apuntes  mifichero   otrofichero
```

Si la ruta de la copia (destino) identifica a un directorio *existente* en el sistema de ficheros, se crea una copia con el nombre original del fichero, pero dentro de dicho directorio.

```
$ cp mifichero apuntes
$ ls apuntes
mifichero
```

En principio, la orden `cp` sólo funciona con ficheros, no con directorios. La opción `-r` permite efectuar una *copia recursiva* de un directorio con todo su contenido. Sólo se ha de especificar el nombre (absoluto o relativo) de ambos directorios. Primero el directorio cuyo contenido *completo* (ficheros y subdirectorios y los que éstos contienen) se desea copiar (debe existir). Después el directorio donde se desea efectuar la copia, `dir2`. Si no existe, se crea. Si existe, se añade todo `dir1` como un *subdirectorio más* de `dir2`, *sin destruir* su contenido anterior.

```
$ cp -r apuntes nuevo
$ ls
apuntes mifichero nuevo otrofichero
$ ls apuntes
mifichero
$ ls nuevo
mifichero
```

3.12. Cambio de ubicación de ficheros y directorios

La orden `mv` (*move*) permite cambiar la *ruta* (nombre incluido) de un fichero o carpeta, lo que en algunos casos puede suponer *cambiarlo* de posición en el sistema de ficheros, simplemente cambiarle el nombre o bien ambas cosas a la vez. Recuerda que la orden `mv` funciona tanto con ficheros como con directorios.

Así, `mv origen destino` cambia el fichero (directorio) identificado por la *ruta* `origen` a la *ruta* especificada en `destino`. Los nombres pueden ser absolutos o relativos.

Si la ruta `destino` identifica a un directorio existente, `mv` mueve el fichero identificado por la ruta `origen` a ese directorio conservando su mismo nombre dentro del directorio.

```
$ mv otrofichero apuntes
$ ls
apuntes   mifichero
$ cd apuntes
$ ls
otrofichero
```

¡Ojo! Si ya existe un fichero con la ruta del segundo argumento, dicho fichero es *sustituido* por el primero, perdiéndose la información del segundo. También podemos cambiar el nombre del fichero al mismo tiempo que lo trasladamos de lugar.

```
$ mv otrofichero ../conotronombre
$ cd ..
$ ls
apuntes   conotronombre  mifichero
```

3.13. Eliminación de un fichero

La orden `rm` (*remove*) permite eliminar ficheros. Al igual que `cp` presenta muchas opciones y variedades.

```
$ cd ~
$ rm conotronombre
$ ls
apuntes mifichero
```

¡Cuidado! La información de los ficheros borrados con `rm` no puede recuperarse.

3.14. Eliminación de un directorio

La orden `rmdir` (*remove directoty*) elimina un directorio *si está vacío*. Si la ruta o nombre especificado no identifica a un directorio *existente y vacío*, se produce un error.

```
$ ls
apuntes mifichero
$ touch apuntes/unfichero
$ rmdir apuntes
rmdir: ‘apuntes’: El directorio no está vacío
$ ls apuntes
unfichero
$ rm apuntes/unfichero
$ rmdir apuntes
$ ls
mifichero
```

Sobre la orden `rm`

Es posible que, según como esté configurado el sistema, el programa *pida confirmación* para borrar el fichero que deseamos eliminar. La opción `-f` permite ignorar este comportamiento (`rm -f`).

continúa en la página siguiente ...

...viene de la página anterior

¿Y si queremos borrar un directorio *y todo su contenido*? Realmente no hace falta que el directorio esté vacío para borrarlo, ya que la opción `-r` primero elimina su contenido y después el directorio en sí. Por supuesto podemos utilizar esta opción combinándola con `-f` para evitar las preguntas pidiendo confirmación (`rm -r`, `rm -r -f` o bien `rm -rf`).

3.15. Edición de ficheros

La edición del contenido de un fichero se realiza a través de aplicaciones específicas para el tipo de fichero que deseamos crear o modificar (imágenes, sonido, texto, bases de datos, películas...). El tipo de fichero fundamental que usaremos en el curso (en ésta y otras asignaturas) es el *fichero de texto* (por ejemplo, el código fuente – las instrucciones – de los programas se almacena en ficheros de texto). El contenido de un fichero de texto se crea y modifica con un *editor de textos*. Hay infinidad de editores de textos. Varían en facilidad de uso y potencia. En el entorno Unix hay dos editores muy populares:

- `vi`: el "editor visual". Es un editor muy antiguo (data de los años 70). Su manejo es bastante complejo. Se sigue usando porque todas las versiones de Unix lo ofrecen (ocupa muy poco espacio en disco y consume pocos recursos – memoria).
- `emacs`. Es un editor algo más moderno (¡años 80!). Su manejo es algo más fácil que el de `vi`. Su característica más importante es la capacidad de extender su funcionalidad para adaptarlo a tareas muy concretas. Es más "pesado" que `vi`.

Para editar un fichero basta con usar el comando `vi` o `emacs` (o su variante `xemacs`) seguido del nombre del fichero. Si no existe, el editor lo crea:

```
$ vi novela.txt
$ emacs redaccion.txt
```

Aprenderás a utilizar el editor `emacs` en clase de prácticas (en realidad, una versión de `emacs` llamada `xemacs`).

3.16. Consultar el contenido de un fichero

Puedes consultar el contenido de un fichero de texto abriéndolo con el editor correspondiente, pero es una operación tan frecuente que Unix proporciona órdenes más cómodas y rápidas. La orden `cat` (*catenate*) muestra el contenido del fichero por la pantalla (la de la consola). El contenido de un fichero es visualizable en pantalla si es texto (ASCII), ya que así es como lo "interpreta" `cat` (en el tema siguiente estudiaremos más cosas al respecto).

```
$ cat novela.txt
Era un noche oscura y tormentosa. El perfil de la universidad
se dibujaba contra el fulgor de los relámpagos. Las siniestras
...
a casa.
```

Si visualizas el contenido de ficheros largos, estos no caben en una pantalla, así que pasan rápidamente y acabas por ver sólo el final. La orden `more` permite visualizar el contenido "pantalla a pantalla".

Esta orden es lo que se llama un *paginador*, esto es permite visualizar el contenido de un fichero página a página ("pantalla a pantalla"). Al igual que `cat`, interpreta el contenido del fichero como texto.

Al ejecutarlo, si el contenido del fichero no "cabe" en la pantalla, el programa se detiene y sólo muestra la parte del fichero que sí "cabe". Al final aparece una línea indicando el porcentaje sobre el tamaño total del fichero que llevamos visualizado. Si pulsamos la tecla `Enter`, se visualiza la siguiente línea del fichero en la parte inferior de la pantalla. Pulsando la *barra espaciadora* (espacio en blanco) se visualiza la siguiente página del fichero en pantalla. No es posible *volver atrás*. El programa termina su ejecución bien al llegar al final del fichero, bien al pulsar la tecla `q`.

Hay una variante de `more` llamada `less` que es más cómoda: permite ir hacia adelante y hacia detrás. La orden `less` es otro paginador, mucho más "potente" que `more`. Permite desplazarse a izquierda, derecha, arriba y abajo usando las teclas de los cursores.

`May+p` permite desplazarse al principio del fichero. `May+g` permite desplazarse al final del fichero. Pulsar la barra espaciadora produce el mismo efecto que `more` (pasa página). A diferencia de `more` *no termina* su ejecución al llegar al final del fichero. Para finalizar el programa, es *necesario* pulsar la tecla `q`.

3.17. Más utilidades para ficheros de texto

- `head` muestra las primeras líneas de un fichero (la "cabeza").

```
$ head -2 novela.txt
Era un noche oscura y tormentosa. El perfil de la universidad
se dibujaba contra el fulgor de los relámpagos. Las siniestras
```

- `tail` muestra las últimas líneas de un fichero (la "cola").

```
$ tail -1 novela.txt
a casa.
```

- `sort` muestra el contenido de un fichero ordenado alfabéticamente línea a línea:

```
$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/bin/bash
daemon:x:2:2:Daemon:/sbin:/bin/bash
lp:x:4:7:Printing daemon:/var/spool/lpd:/bin/bash
...
news:x:9:13:News system:/etc/news:/bin/bash
uucp:x:10:14:Unix-to-Unix CoPy system:/etc/uucp:/bin/bash
$ sort /etc/passwd
adabas:x:36:100:Adabas-D database admin:/usr/lib/adabas:/bin/bash
amanda:x:37:6:Amanda admin:/var/lib/amanda:/bin/bash
amarzal:MxGwFkl878xxR:1001:1999:Andrés Marzal:/home/amarzal:/bin/bash
at:x:25:25:Batch jobs daemon:/var/spool/atjobs:/bin/bash
...
wwwrun:x:30:65534:WWW daemon apache:/var/lib/wwwrun:/bin/bash
zope:x:64:2:Zope daemon:/var/lib/zope:/bin/false
```

- `grep` muestra las líneas de un fichero que contienen una palabra o fragmento de texto.

```
$ grep oscura novela.txt
Era un noche oscura y tormentosa. El perfil de la universidad
su zarpa oscura y putrefacta agarró al estudiante por el cuello
$ grep daemon /etc/passwd
daemon:x:2:2:Daemon:/sbin:/bin/bash
lp:x:4:7:Printing daemon:/var/spool/lpd:/bin/bash
mail:x:8:12:Mailer daemon:/var/spool/clientmqueue:/bin/false
at:x:25:25:Batch jobs daemon:/var/spool/atjobs:/bin/bash
wwwrun:x:30:65534:WWW daemon apache:/var/lib/wwwrun:/bin/bash
irc:x:39:65534:IRC daemon:/usr/lib/ircd:/bin/bash
zope:x:64:2:Zope daemon:/var/lib/zope:/bin/false
```

3.18. Ayuda: manuales en línea

Unix presenta centenares de órdenes y es fácil que olvidemos qué hace cada una, cómo se usa y qué opciones presenta. La orden `man` (*manual*) permite consultar páginas de manual que describen cada orden (¡incluso el propio man!).

```
$ man ls
NAME
       ls - list directory contents

SYNOPSIS
```

```
       ls [OPTION]... [FILE]...

DESCRIPTION
       List information about the FILEs (the current directory by default).
       Sort entries alphabetically if none of -cftuSUX nor --sort.

       -a, --all
              do not hide entries starting with .

       -A, --almost-all
              do not list implied . and ..
...
```

3.19. Comodines

Hasta ahora hemos aprendido a manipular ficheros y directorios de uno en uno. En ocasiones queremos actuar del mismo modo sobre un conjunto de ficheros; por ejemplo, borrar todos los ficheros de texto de un directorio o mostrar el contenido de todos los ficheros cuyo nombre empieza por la letra "a". Los intérpretes de órdenes Unix permiten usar comodines al especificar el nombre de uno o más ficheros.

- Asterisco (`*`): el asterisco significa cero o más caracteres. Por ejemplo, borrar todos los ficheros del directorio activo que acaban con extensión `.txt`:

  ```
  $ rm *.txt
  ```

  Borrar todos los ficheros del directorio activo que empiezan por la letra `a`:

  ```
  $ rm a*
  ```

  Borrar todos los ficheros del directorio activo que empiezan por la letra `a` y acaban con `.txt`:

  ```
  $ rm a*.txt
  ```

  Borrar todos los ficheros del directorio `/tmp`:

  ```
  $ rm /tmp/*
  ```

  Borrar todos los ficheros del directorio activo:

  ```
  $ rm *
  ```

- Interrogante (`?`): representa siempre *un único carácter* (que puede ser cualquiera). Por ejemplo, si en el directorio activo tuviésemos los ficheros `f1.txt`, `f2.txt`, `fa.txt`, `fm.txt` y `f.txt` la siguiente orden borraría a todos excepto al último:

  ```
  $ rm f?.txt
  $ ls
  f.txt
  ```

  Pero con esta orden se borrarían todos:

  ```
  $ rm f*.txt
  $ ls
  ```

Los comodines permiten trabajar con muchos ficheros a la vez, seleccionándolos con criterios bastante sofisticados. Los entornos gráficos suelen fallar, por ejemplo, en aspectos como expresar criterios similares para *seleccionar cómodamente ficheros*. Recuerda que otra ventaja importante de los intérpretes de órdenes es que son *programables*: es posible crear un fichero con una secuencia de órdenes que se ejecuta cada vez que lo deseamos (*script*).

3.20. Redirección

El símbolo `>` *redirige la salida que produce la ejecución de un determinado programa (u orden) en la pantalla del ordenador a un fichero*, es decir, lo que debería salir en pantalla pasa a formar parte de un fichero. Por ejemplo, usando redirección de salida, podríamos crear un fichero que contuviese un listado de ficheros:

```
$ ls > contenido
$ ls
apuntes contenido mifichero
$ cat contenido
apuntes mifichero
```

Hemos visto antes que `cat` muestra el contenido de un fichero. Si lo usas sin ningún nombre de fichero, hace una cosa distinta: lee el texto que escribas por teclado y lo muestra por pantalla línea a línea.

```
$ cat
un ejemplo que tecleamos
un ejemplo que tecleamos
para ver cómo
para ver cómo
cat muestra el texto.
cat muestra el texto.
^D
```

Para finalizar la ejecución de `cat` *en estas circunstancias*, hemos de pulsar la tecla de control y la tecla `D` simultáneamente (`^D`): así avisamos de que no vamos a teclar más texto. De hecho, con la redirección podemos usar `cat` como un editor de textos muy primitivo:

```
$ cat > unfichero
un ejemplo que tecleamos
para ver cómo
cat muestra el texto.
^D
$ cat unfichero
un ejemplo que tecleamos
para ver cómo
cat muestra el texto.
```

La redirección funciona con cualquier orden que muestre información en pantalla:

```
$ sort unfichero > ordenado
$ cat ordenado
cat muestra el texto.
para ver cómo
un ejemplo que tecleamos
```

Hemos aprendido a redireccionar *la salida* (la información que se muestra por pantalla). Se puede redireccionar también *la entrada* (la información que tecleamos), sólo que entonces usamos el símbolo `<`. En el ejemplo que se muestra a continuación, los datos de entrada para `cat` son las líneas de texto almacenadas en el fichero `ordenado`:

```
$ cat < ordenado > otrofichero
$ cat otrofichero
cat muestra el texto.
para ver cómo
un ejemplo que tecleamos
```

Una opción muy potente[6] consiste en *enlazar* la *salida* de un programa con la *entrada de otro*, formando una cadena o *tubería* a través de la cual un programa envía datos y el otro los recibe. Este tipo de enlace se denomina precisamente *tubería* y se expresa con la barra vertical (`|`).

Por ejemplo, la orden `who` indica qué usuarios están conectados a un ordenador en un momento determinado, pero proporciona una lista desordenada. La podemos obtener ordenada combinando las “habilidades” de `who` y `sort` con una tubería:

[6] Y que se utiliza frecuentemente en la realización de *scripts* o programas del intérprete de órdenes.

```
$ who
tenaj    pts/1    Sep 25 07:45 (stargate.si.uji.es)
smarti   pts/2    Sep 25 09:15 (chaini.act.uji.es)
pepe     pts/0    Sep 25 08:48 (pepe.si.uji.es)
root     pts/4    Sep 24 09:12 (dentaku.si.uji.es)
barreda  pts/5    Sep 24 08:18 (fermi.act.uji.es)
vilata   pts/7    Sep 25 09:32 (mezzanine.act.uji.es)
llopis   pts/6    Sep 24 17:22 (ced0113.act.uji.es)
climente pts/11   Sep 25 09:49 (nobel.phys.uni.torun.pl)
jvilar   pts/13   Sep  9 11:00 (dafne.act.uji.es)
navarro  pts/12   Sep 24 15:05 (kiko.si.uji.es)
$ who | sort
barreda  pts/5    Sep 24 08:18 (fermi.act.uji.es)
climente pts/11   Sep 25 09:49 (nobel.phys.uni.torun.pl)
jvilar   pts/13   Sep  9 11:00 (dafne.act.uji.es)
llopis   pts/6    Sep 24 17:22 (ced0113.act.uji.es)
navarro  pts/12   Sep 24 15:05 (kiko.si.uji.es)
pepe     pts/0    Sep 25 08:48 (pepe.si.uji.es)
root     pts/4    Sep 24 09:12 (dentaku.si.uji.es)
smarti   pts/2    Sep 25 09:15 (chaini.act.uji.es)
tenaj    pts/1    Sep 25 07:45 (stargate.si.uji.es)
vilata   pts/7    Sep 25 09:32 (mezzanine.act.uji.es)
```

Y si nos interesase conocer sólo los tres primeros en orden alfabético, podríamos combinar la tubería actual con la orden `head` mediante una nueva tubería:

```
$ who | sort | head -3
barreda  pts/5    Sep 24 08:18 (fermi.act.uji.es)
climente pts/11   Sep 25 09:49 (nobel.phys.uni.torun.pl)
jvilar   pts/13   Sep  9 11:00 (dafne.act.uji.es)
```

Si, además, queremos almacenar esa información en un fichero, usamos redirección:

```
$ who | sort | head -3 > primeros
$ cat primeros
barreda  pts/5    Sep 24 08:18 (fermi.act.uji.es)
climente pts/11   Sep 25 09:49 (nobel.phys.uni.torun.pl)
jvilar   pts/13   Sep  9 11:00 (dafne.act.uji.es)
```

Observa que puedes obtener una funcionalidad similar a las tuberías utilizando redirección y un fichero auxiliar:

```
$ who > auxiliar
$ sort < auxiliar
```

MS-DOS, un SO antiguo predecesor de Windows, "copió" el mecanismo de redirección de Unix. Sin embargo, utilizaba esta "estrategia" para recrear (imitar) el funcionamiento de las tuberías de Unix. Las tuberías de Unix ofrecen dos importantes ventajas:

- no hace falta un fichero auxiliar, así que ahorras espacio en disco (memoria);
- los procesos implicados se ejecutan *simultáneamente* (no acaba el primero y entonces empieza el segundo) comunicando directamente la salida de uno con la entrada del otro.

En Unix, la mayoría de las órdenes están concebidas como herramientas que luego pueden concatenarse mediante tuberías para crear otras más potentes.

3.21. Otras órdenes útiles

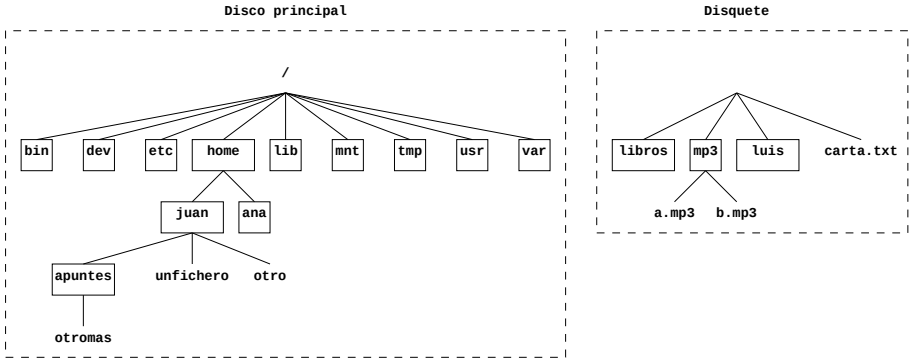
- `lpr` (*line printer*) manda un trabajo a impresora (con la opción `-P` se puede indicar a cuál si hay más de una configurada en el sistema).
- `quota` consulta la cuota de disco (la cantidad de megas que puedes usar en el disco duro) y el espacio disponible (úsala con la opción `-v` en `anubis` para saber en todo momento cómo estás de espacio en disco).

- `gzip` comprime un fichero. Crea un fichero con el mismo nombre y una extensión adicional (`.gz`). Si el fichero es de texto, puede reducirlo significativamente (por ejemplo, pasando a ocupar un 30 % del tamaño original).
- `gunzip`: descomprime ficheros comprimidos con `gzip`.
- `file` informa acerca del tipo de contenido de un fichero.
- `wc` (*word count*) cuenta el número de caracteres, palabras y líneas de un fichero de texto.
- `tar` (*tape archiver*) crea un único archivo que empaqueta varios ficheros y directorios en su interior, manteniendo la estructura de directorios que tuvieran en el momento de la creación del archivo. Útil para hacer copias de múltiples ficheros y directorios. Igualmente permite hacer copias de seguridad con cierta comodidad. *Compatible* con WinZip. Para su utilización, nos centraremos en:
  - `tar -cvzf archivo.tgz ruta-directorio` para archivar y comprimir en el fichero llamado `archivo.tgz` todos los ficheros contenidos en el directorio identificado por `ruta-directorio`, incluyendo los contenidos de los subdirectorios y respetando la *estructura* existente en el momento de ejecutar la orden.
  - `tar -tvzf archivo.tgz` para visualizar en pantalla el contenido de un archivo creado con `tar`.
  - `tar -xvzf archivo.tgz` recupera los ficheros y subdirectorios contenidos en el fichero denominado `archivo.tgz`, *manteniendo la estructura almacenada*, en el directorio en el que nos encontremos.

  Si eliminamos la letra `z` de las opciones no se realiza (des)compresión alguna, simplemente se archiva o se extrae.

4. Unidades

Las unidades (discos duros, disquetes) suelen tener, cada una, su propia jerarquía de directorios y ficheros; es decir tienen un sistema de ficheros completo.



Windows ofrece cada sistema de ficheros (jerarquía) por separado. Se accede a cada una de las unidades[7] con letras:

- `A:` suele ser el disquete.
- `B:` está reservado, por cuestiones "históricas"[8], a una segunda unidad de disquete.
- `C:` es el disco duro principal.
- `D:` suele ser otro disco duro o un CD-ROM (o DVD, CD-RW, etc).
- `E:` suele ser otro CD-ROM (o DVD, CD-RW, etc).

[7] Mejor dicho, al sistema de ficheros de cada unidad.

[8] Que tienen que ver con el hecho de que las primeras generaciones de PCs no tenían disco duro. Sin embargo, muchos PCs venían equipados con dos disqueteras.

No obstante, los SSOO de la familia UNIX *sí* ofrecen una jerarquía de ficheros única. Luego, ¿Qué hacemos cuando queremos acceder a otras unidades distintas del disco principal?

Es posible vincular una unidad a un directorio Unix. A esa acción se la denomina *montaje*. Por tanto, en un determinado directorio, se puede "colgar" la jerarquía de ficheros y directorios de una unidad.

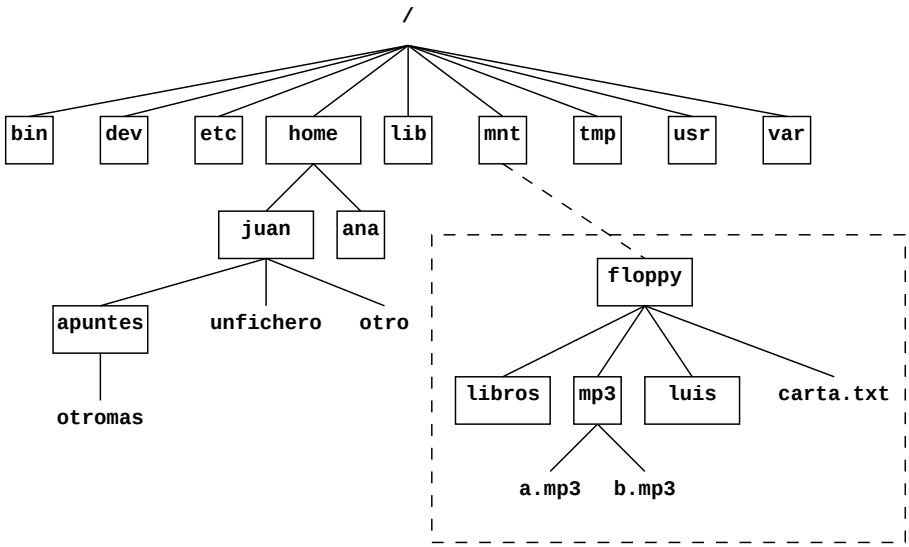
A partir de ese momento, los ficheros y directorios de la unidad montada son indistinguibles (salvo cosas como la velocidad) de los del disco principal. Para montar unidades disponemos de la orden `mount`. Típicamente las unidades se montan en subdirectorios de `/mnt`.

- `/mnt/floppy` suele ser el disquete.
- `/mnt/windows` suele ser el disco en el que tenemos instalado Windows (si tenemos Windows y Linux en la misma máquina).
- `/mnt/cdrom` suele ser el CD-ROM.
- `/mnt/dvd` suele ser el DVD.
- ...

Por ejemplo, para montar un disquete y así poder tener acceso a su sistema de ficheros, has de ejecutar:

```
$ mount /mnt/floppy
```

Éste es el resultado, teniendo en cuenta la imagen anterior:



Aunque insertes el disquete en la disquetera, si no lo montas, Unix no lo "ve". Cuando has acabado con el disquete, has de *desmontarlo* antes de retirarlo de la disquetera. Si no lo haces, puedes perder datos. La orden para desmontar es `umount` (de *unmount*).

```
$ umount /mnt/floppy
```

¿Sabías que ...?

En Unix existen *grosso modo* dos tipos de dispositivos:

De bloques, que están divididos en una serie de bloques numerados, permitiendo acceder a ellos de forma individual. El ejemplo más claro son los discos duros.

De caracteres, que conceptualmente se ven como una corriente de bytes que se leen o escriben secuencialmente. Ejemplos de estos dispositivos son el ratón o el teclado.

continúa en la página siguiente ...

. . . viene de la página anterior

5. Formatos de sistemas de ficheros

Ya sabemos que los sistemas de ficheros son una *abstracción* que nos permite organizar los datos en un disco duro. Pero, por ejemplo, un disco duro es una superficie magnética que memoriza unos y ceros. Hay varias formas de escribir/leer los unos y ceros para que se interpreten correctamente como ficheros y directorios. Decimos que hay varios *formatos* o *tipos* de sistemas de ficheros.

No existe un tipo único de formato para representar un sistema de ficheros. Esta es una cuestión que depende del SO que queramos utilizar (y a veces de los dispositivos). Diferentes SSOO usan mecanismos internos diferentes y, en consecuencia, mecanismos distintos para representar los sistemas de ficheros con los que operan. Los tipos más comunes que podemos encontrarnos son:

FAT-msdos: usado por MS-DOS. Obsoleto. Es la base del VFAT o FAT extendido, actualmente usado en algunos sistemas Windows (95, 98, Me). Bastante limitado (por ejemplo, los nombres de ficheros y carpetas se limitan a 8 caracteres más 3 para la extensión).

VFAT-FAT32: también conocido como FAT extendido. Usado por muchas unidades Windows. Básicamente es el FAT con soporte para nombres largos. Los disquetes suelen presentar ese formato.

ext2: el sistema de ficheros ext2 es el sistema de ficheros estándar de Linux, de momento[9]. Soporta nombres largos.

ReiserFS: también propio de los sistemas Linux, aunque más moderno y avanzado que ext2.

hfs: éste es el sistema de ficheros estándard de los clásicos Apple Macintosh. Soporta nombres largos y ciertos caracteres especiales.

iso9660: es el sistema de ficheros utilizado en los CD-ROM, esto es discos compactos de *datos*. Es muy limitado, pero afortunadamente existen "extensiones" como Rock Ridge (en Linux) y Joliet (en Windows) que permiten dotarlo de soporte para nombres largos, uso de mayúsculas y minúsculas, permisos, etc.

NTFS: sistema de ficheros "nacido" con Windows NT (NTFS significa New Technology File System). Otros Windows posteriores como Windows 2000 y Windows XP lo han adoptado. En estos sistemas se sugiere emplearlo[10] para mejorar el rendimiento.

Hay muchos más: como *udf*, el nuevo sistema de ficheros usado, fundamentalmente, en los DVD (Digital Video Disc).

Completando información:

Windows soporta nativamente los sistemas FAT y VFAT. También soporta iso9660 con las extensiones denominadas *Joliet*. Estas extensiones fueron desarrolladas por Microsoft y permiten la utilización de nombres largos en formato *unicode* (un código de caracteres de 16 bits que soporta casi todos los idiomas del mundo –veremos más sobre ello en el tema siguiente). El formato udf está igualmente soportado en Windows.
Para poder utilizar sistemas de ficheros formateados como ext2 se pueden utilizar herramientas especiales *no* propias de Windows.
Linux soporta todos los tipos anteriormente mencionados, suponiendo que haya sido *adecuadamente* configurado para ello. Básicamente esta "configuración" consiste en tener preparados (compilados) los módulos necesarios para que puedan ser cargados por el *kernel*[11] de Linux.

[9] Muchos sistemas están sustituyéndolo por la siguiente generación: ext3, el cual tiene prestaciones más similares a las que proporciona ReiserFS.
[10] Al menos en la partición donde se va a instalar el sistema y el software.
[11] Núcleo del SO.

Cuando añades un disco duro a tu ordenador, el disco está *virgen*, es decir, no tiene formato alguno. La primera acción es, pues, *formatear* el disco con el formato adecuado. Windows detecta automáticamente que un disco no tiene formato, pero sólo permite formatearlo con formatos Windows. Ciertas herramientas del ordenador permiten configurar los discos y muchos otros elementos del ordenador, tanto en Linux como en Windows.

6. Configuración del ordenador

Los entornos gráficos permiten configurar *ciertos aspectos* de funcionamiento del sistema con relativa facilidad y por usuarios relativamente inexpertos. Sin embargo, algunas características sólo se pueden configurar correctamente *cuando se sabe lo que se está haciendo*.

En muchas ocasiones, configurar adecuadamente ciertos servicios y/o componentes del sistema implica tener que leerse documentación, manuales, etc. y tener que editar determinados ficheros de configuración *desde la consola*.

6.1. Configuración de Windows

¿Sabías que . . . ?

Dentro del menú de configuración de Windows podemos encontrar las siguientes opciones:
Panel de control: es un acceso directo a una carpeta donde encontramos iconos que permiten configurar diversos aspectos de Windows.
Impresoras: para configurar las impresoras que tengamos instaladas en el sistema y controlar lo que mandemos a imprimir.
Barra de tareas: para cambiar las propiedades de la barra de tareas.
Opciones de carpeta: para cambiar opciones referentes al comportamiento de las carpetas (si se abren en la misma ventana o no, etc.) y el aspecto del Escritorio, qué información se va a mostrar (archivos ocultos, p.e.) y *asociación* de extensiones (tipos de fichero) con aplicaciones.

Figura 4: El panel de control de Windows.

Windows facilita la configuración a través de un *panel de control*: un conjunto de utilidades de configuración agrupadas. Se accede a él a través del menú "Inicio" (esquina inferior izquierda) o también desde el icono "MiPC". En ambos casos aparecerán una serie de iconos con los que podemos configurar diversos aspectos de Windows. Muchas de estas utilidades se encuentran con el mismo nombre o similar en distintas versiones: 98, Me, NT, 2000 y XP. Hay herramientas en el panel de control para:

- *Agregar hardware*: permite indicar a Windows que hemos añadido una tarjeta de expansión, un dispositivo USB, etc[12]. Sólo es necesario recurrir a él si no se produjo una detección automática al arrancar. Si tenemos "conflictos" entre dispositivos (veáse el tema siguiente), hay que utilizar el panel *Sistema*.
- *Agregar/quitar programas*: permite instalar/desinstalar programas. Como los programas suelen venir con sus instaladores propios, este elemento se usa casi exclusivamente para desinstalar programas. Es recomendable, si el programa fue correctamente instalado, eliminarlo desde aquí para asegurarse de que se realizan los cambios necesarios. Igualmente podemos añadir o eliminar componentes optativos de Windows como juegos, controladores de fax o accesorios multimedia. *También permite crear un disco de arranque, muy útil si se avería el disco duro.*

[12] Es decir, que tenemos un nuevo dispositivo instalado en el sistema.

- *Configuración regional*: adapta características del ordenador al español o catalán. En general, permite cambiar una serie de características que dependen del país y la lengua que utilicemos. Las dos alfabetizaciones que aparecen junto al español son la tradicional (ch y ll son consideradas letras) y la moderna (que no las considera como tal).
- *Administración de energía*: controla el APM (advanced power management). Mediante esta utilidad se puede hacer que los dispositivos (típicamente los discos duros y monitores) que no estén siendo utilizados durante un tiempo pasen a un modo de bajo consumo de energía. Con equipos portátiles aparecen más opciones (batería, configuración consumo de energía del equipo, etc.) y permite "dormir" al ordenador, el monitor o el disco duro tras cierta inactividad.
- *Fecha y hora*: permite ajustar la zona horaria, fecha y hora y si se quiere que se realice automáticamente el cambio de horarío verano/invierno.
- *Fuentes*: permite instalar/desinstalar tipografías.
- *Impresoras*: permite acceder a cada impresora conectada y configurar algunas características (papel, calidad de impresión, etc.), ver la lista de tareas de impresión pendientes o configurar una como predeterminada.
- *Internet*: controla aspectos de la presentación de páginas, modo de conexión; permite indicar qué software usaremos para correo, navegación, etc.
- *Módem*: configura el módem.
- *Mouse*: permite cambiar diversas características del comportamiento del ratón: control de los botones para personas diestras o zurdas; velocidad del doble clic; forma del puntero (cursor) en pantalla; velocidad de movimiento; y otras dependientes del dispositivo utilizado.
- *Multimedia*: permite configurar el manejo de diversos aspectos de audio y vídeo.
- *Accesibilidad*: adapta características para facilitar el uso a discapacitados.
- *Pantalla*: permite cambiar las propiedades de la pantalla:
  - Fondo: para cambiar la imagen de fondo del escritorio.
  - Protector de pantalla: son pequeños programas que evitan que el monitor presente siempre la misma imagen cuando durante un tiempo no hay interacción por parte del usuario.
  - Apariencia: permite cambiar características de los distintos elementos de las ventanas.
  - Configuración: permite cambiar la resolución y los colores de la pantalla.
  - Efectos: efectos visuales en ventanas e iconos y permite cambiar iconos para determinados objetos (papelera, etc.).
  - Web: para "ver" el Escritorio como si fuese una página Web.
- *Red*: controla el dispositivo de red y permite introducir algunos datos necesarios (servidor de nombres, etc.) para configurar la conexión a Internet (o a una "red local").
- *Sistema*: muestra información del ordenador y permite resolver posibles conflictos entre dispositivos. Presenta las siguientes opciones:
  - General: presenta datos de la versión del sistema Windows y del ordenador.
  - Administrador de dispositivos: permite ver los dispositivos conectados, sus propiedades y posibles conflictos.
  - Perfiles de hardware: permiten habilitar y deshabilitar selectivamente distintos dispositivos al comenzar la sesión. Para ello se crea un nombre de perfil y se determina que dispositivos estarán disponibles con él.
  - Rendimiento: permite configurar ciertos aspectos que influyen en el rendimiento del sistema.
- *Sonidos*: permite asociar sonidos a los distintos eventos del sistema.
- *Teclado*: permite configurar el teclado en aspectos como la velocidad de repetición de las teclas o el idioma.

Sobre la barra de tareas de Windows:

Podemos cambiar su posición mediante un click en alguna zona libre y arrastrándola. Para cambiar el tamaño, nos movemos al borde y actuamos como lo haríamos con cualquier ventana.
Además podemos cambiar sus propiedades desde el menú de inicio: cambiar cuándo se ve y cuándo no, la presencia del reloj y el tamaño de los iconos. También podemos cambiar los programas que aparecen en el menú de inicio.

6.2. Configuración de Linux

La configuración depende del entorno de usuario concreto que usemos. En KDE, por ejemplo, hay un panel de control similar al de Windows. Para poder configurar un sistema Linux debes ser administrador (*root*) del sistema, es decir conectarte al sistema como superusuario.

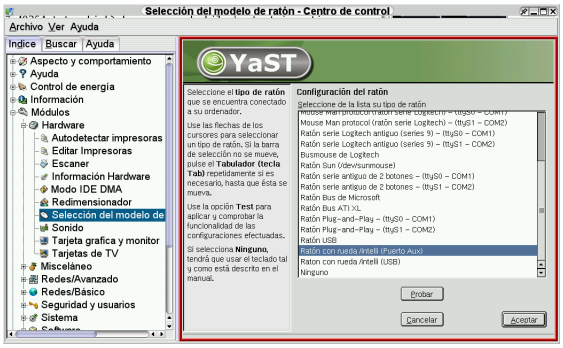


Figura 5: El centro de control de KDE en SuSE.

7. Más sobre Linux

Linux es un sistema operativo completamente diferente de Microsoft Windows. Durante tus estudios te verás obligado a utilizar Linux para realizar diferentes trabajos. ¿Por qué usar Linux, si ya sabes usar Windows?

- Linux es un sistema operativo de la familia *Unix*. Unix cuenta con tres décadas de desarrollo e infinidad de utilidades disponibles y sigue muchos estándares para redes, comunicación, representación de datos, etc.
- Hay muchas herramientas (gratuitas) para *programación* que se consideran estándar *de facto*.
- También dispone de *entornos de usuario* que facilitan su uso: KDE y GNOME por ejemplo.
- Linux es *software libre* y, además, *gratuito*. Lo primero es muy importante porque eso significa que disponemos del código fuente del programa para su estudio y posible modificación.

Hay una razón todavía más importante. Utilizando Linux, si lo deseas, puedes aprender muchas cosas sobre informática. Existe mucha documentación, manuales, etc. Con Linux podrás estudiar y aprender y, lo más importante, observar la aplicación práctica con todo lujo de detalles de muchos de los conceptos que estudiarás en la carrera. Probablemente al principio, te basarás mucho en la utilización de herramientas gráficas de configuración, pero conforme vayas aprendiendo y cojas soltura, te atrevarás a modificar los ficheros de configuración por tí mismo, sin la ayuda de otras herramientas. Ésta es una de las ventajas que proporciona la versatilidad de Linux: si quieres, puedes “abstraerte” de los detalles de “bajo nivel” como en Windows. Pero, *a diferencia de Windows*, si quieres o necesitas “arremangarte y coger el destornillador” en Linux puedes hacerlo (y encontrarás mucha documentación que te ayudará en tu trabajo).

¿Sabías que . . . ?

Linux es un sistema que fue creado por Linus Torvalds y posteriormente desarrollado con la ayuda de multitud de personas (*hackers*).
El objetivo que se perseguía era crear un sistema UNIX completo (que fuese POSIX), incluyendo: multitarea real, memoria virtual, bibliotecas compartidas, redes con conexión TCP/IP, . . . Aunque inicialmente se desarrolló para la familia x86 de Intel (desde el 386 en adelante), existen ya versiones (experimentales) para máquinas como Alpha (Digital), Sparc (SUN), 68000 (Mac), MIPS, Power PC y otras.
continúa en la página siguiente . . .

...viene de la página anterior

Tiene muchas de las características de los UNIX comerciales pero se distribuye bajo licencia GNU. En realidad, no forma parte del proyecto GNU (`http://www.gnu.org`), pero muchas de las utilidades que lo acompañan sí.
El proyecto GNU pretende crear un entorno de trabajo para ordenador empleando software 100 % libre. Ello incluye un sistema operativo GNU (*Hurd*) actualmente en desarrollo (existe ya una versión experimental). De momento GNU, a falta de un kernel propio, ha adoptado Linux.

Propiamente hablando, Linux es exclusivamente el kernel. Existen varias ramas de desarrollo (2,0, 2,2, 2,3, 2,4, 2,5) y distintas versiones dentro de cada rama (p.e., la última versión disponible en la rama 2,4 es la 2,4,19). Las ramas impares (p.e. 2,3) se asignan a versiones *inestables* y experimentales (en desarrollo), mientras que las pares (p.e. 2,2) son versiones *estables*.

Hay gran cantidad de programas libres, *shareware* e incluso comerciales, que se pueden utilizar con el kernel: compiladores, entornos de ventanas, editores, procesadores de texto, hojas de cálculo, etc. Reunir estos programas supone una gran inversión de tiempo y puede requerir bastante esfuerzo hacer que funcionen de manera conjunta.

Para que sea más fácil montar un sistema basado en Linux, se han creado las *distribuciones*, que reunen un kernel (o varios, siempre versiones *estables*, posiblemente incluyendo ramas distintas) y una serie de programas que permiten tener una máquina lista para las tareas más habituales. Además incluyen utilidades que facilitan las tareas de configuración y administración de los programas que se instalan.

A continuación se muestran algunas de las distribuciones *principales* más conocidas. Esta información se ha sacado del *"Rincón de Linux. Linux para hispanohablantes"*, `http://www.linux-es.com/`, en el apartado distribuciones:

RedHat: tiene muy buena calidad en contenidos y soporte a los usuarios por parte de la empresa que la distribuye. Fácil de instalar. Es la distribución estándar de Linux en EEUU. `http://www.redhat.com`

Debian: no la produce una empresa, sino una comunidad de voluntarios. Tal vez sea la mejor distribución de Linux que existe en la actualidad. Si tienes una buena conexión, podrás actualizarte fácilmente vía Internet. El proceso de instalación es un poco más complicado, aunque se aprende mucho. Se compensa de sobra con la facilidad de actualización que proporciona, muy superior al de otras distribuciones. No se recomienda para no iniciados, aunque si quieres aprender de verdad y tienes tiempo que invertir, atrévete con ella. `http://www.es.debian.org/` (mirror en España)

S.u.S.E: una distribución que ha entrado con mucha fuerza en el mercado español; es una de las más populares en Europa. La realiza una empresa alemana. Fácil de instalar. Programa de instalación muy bueno. `http://www.suse.com`

Mandrake: otra de las más populares en Europa. Fácil de instalar. No obstante, resulta algo menos completa que la SuSE. `http://www.mandrake.com/es/`

Caldera: Caldera proporciona tres versiones de su distribución: Çaldera eDesktop", Çaldera eServerz Çaldera Linux Technology Preview". Todas ellas se pueden bajar desde Internet, excepto las aplicaciones comerciales que se incluyen en las mismas. `http://www.calderasystems.com`

Slackware: de las primeras. Ha pasado por un periodo de "crisis", durante el cual no se ha actualizado muy a menudo, aunque parece que está volviendo con fuerza. `http://www.slackware.com`

Las empresas que hacen las distribuciones suelen cobrar por sus servicios (la excepción es Debian, que solamente cobra un donativo, aparte del soporte en CDs). A cambio ofrecen:

- manuales,
- ayuda técnica,
- software que facilita la instalación.

Aunque estas empresas cobren por la distribución, es perfectamente *legal* efectuar copias para uso personal. De hecho, algunas distribuciones permiten descargarse las "imágenes" de los CDs vía Internet, para que uno mismo pueda grabarse los CDs de instalación. Por contra, si haces copias de Windows, cometes un acto *ilegal* y vulneras los derechos de propiedad intelectual de la empresa Microsoft.

Las distribuciones suelen ofrecer mecanismos para controlar qué está instalado y qué no. La manera más normal es agrupar los programas y posible documentación asociada en *paquetes* y prever mecanismos para instalar o eliminar dichos paquetes. De esta manera se puede tener un cierto control sobre lo que hay instalado y permitir que sea más fácil actualizar las versiones.

En el momento de la instalación, se suelen elegir los paquetes agrupados según alguna categoría adecuada (p.ej, para uso en portátil, en estación de desarrollo de programas o para uso como servidor de red). Después, se pueden ir añadiendo y quitando paquetes mediante órdenes especiales o entornos más o menos cuidados. Cada distribución suele incorporar mecanismos (uno o varios programas, gráficos o no) para la gestión de paquetes desde varios dispositivos (incluso desde Internet). Hay fundamentalmente dos "*sistemas de paquetes*" para distribuir software en Linux "listo para funcionar":

RPM : sistema usado por RedHat, Mandrake y SuSE. Un paquete de RedHat, por ejemplo, *suele* funcionar sin problemas en SuSE.

deb : sistema usado por Debian. Es un sistema con mejoras importantes, superior a RPM, pero en principio incompatible con éste.

¿RPM y deb incompatibles?

Aunque en principio puedan parecer formatos de distribución de software incompatibles, en realidad existe una herramienta que permite convertir paquetes de software "en formato" RPM a deb y viceversa. Esta herramienta es el programa `alien`. Si quieres saber más, abre una consola, teclea `man alien` y pulsa Enter.

7.1. Sobre la instalación de Linux

Aunque suele variar entre distribuciones, la instalación tiene los siguientes "grandes pasos":

- Arrancar un sistema básico. Generalmente tenemos tres opciones:
  - Utilizar uno o más disquetes (y realizar la instalación del resto del software por Internet).
  - Utilizar el propio CDROM (el CD1 de la instalación) como dispositivo de arranque (si éste lo permite).
  - Utilizar el DVD de instalación como dispositivo de arranque (si éste lo permite).
- Utilizar el sistema básico para realizar las particiones y preparar el disco.
- Instalar los paquetes (aplicaciones) deseados.
- Configurar algunos aspectos (contraseña del superusuario, crear usuarios "normales", configurar las X, etc.).

Un paso previo a considerar

En muchas ocasiones, puede resultar conveniente ajustar y/o cambiar ciertas opciones de configuración de la BIOS (consulta las transparencias del tema 3 sobre hardware):
Si instalamos desde CDROM o DVD (autoarrancable), hay que elegir el orden adecuado de arranque en el ordenador, es decir, indicar en la BIOS que se desea poder arrancar desde CDROM (o DVD) antes que del disco duro.
Poner el máximo de memoria RAM posible como *extendida*. Linux requiere memoria extendida y *no* puede manejar memoria *expandida*.
Deshabilitar las opciones anti-virus de la BIOS. No suelen ser compatibles con Linux. En cualquier caso, y gracias al sistema de permisos de los ficheros y el mecanismo de memoria protegida de Linux, los virus son casi desconocidos.

continúa en la página siguiente ...

. . . viene de la página anterior

Si la placa proporciona algo como *shadow RAM* o *BIOS caching*, hay que deshabilitarlo en la configuración de la BIOS. Este tipo de uso de la memoria puede interferir en el mecanismo de operación de Linux con los dispositivos, bastante eficaz por "naturaleza".
Deshabilitar las opciones de *Advanced Power Management* (APM). Linux puede hacer un mejor control de la administración de la energía que cualquier BIOS.
Si se tiene RAM con detección de errores (soporte de bits de paridad en la RAM) y la placa base es capaz de manejarla (nuevamente consúltense las transparencias del tema 3), habilitar en la BIOS la(s) opción(es) que permiten disparar la interrupción correspondiente cuando se detecte un error en la memoria.
Deshabilitar opciones que digan algo así como "15-16 MB *memory hole*" o algo parecido. Esto puede entorpecer la gestión de la memoria en Linux.

Arranque del sistema básico

Para comenzar a hacer la instalación, es necesario tener un sistema Linux ya ejecutándose. Para conseguirlo, hay dos opciones principales.
Una posibilidad es crear uno o más disquetes de arranque con un sistema linux preparado para el hardware más habitual y una serie de disquetes que tengan las utilidades básicas para configurar el sistema.
Estos disquetes se preparan a partir de ficheros especiales (*imágenes*) que se escriben en el disco utilizando el programa MS-DOS `rawrite` (se puede descargar de Internet). Este programa sirve para hacer copias *binarias* (por bloques o sectores) de ficheros. No obstante es preferible usar la orden `dd` si se tiene acceso a un Unix (o Linux). Cada imagen tiene el tamaño justo del disco (excepto posiblemente la última) y en ella está su propio sistema de ficheros.
Una alternativa mucho más cómoda a los disquetes es la utilización de un CDROM (o DVD) como disco de arranque que ejecuta automáticamente un programa de instalación que incluye un sistema Linux básico.
Para esto, se tendrá que cambiar la configuración de la BIOS de modo que pruebe el arranque desde el lector de CD (o DVD) antes que el disco duro.
También es necesario que el CDROM (o DVD) de la distribución esté ya preparado para ser utilizado como disco de arranque. Prácticamente cualquier distribución moderna se puede conseguir en CDROMs, de los cuales el primero se puede utilizar como arranque.

Usualmente instalas Linux sobre un ordenador que lleva Windows preinstalado. Puede ocurrir lo siguiente:

- Si sólo tienes un disco duro, has de hacer espacio para Linux. Ello implica *particionar* el disco y *redimensionar* el sistema de ficheros de Windows. Los discos duros pueden ser divididos en distintas partes (*particiones*), permitiendo que el ordenador las vea como si fueran discos distintos. Esto permite, por ejemplo, organizar mejor los datos dentro del disco o tener simultáneamente más de un sistema operativo en el ordenador. Para instalar Linux, será necesario particionar el disco de modo que pueda coexistir con MS-DOS, Windows u otros sistemas que puedas tener instalados. Necesitas al menos una partición para Linux y otra para Windows.

- Si tienes más de un disco, puedes destinar un disco a cada sistema operativo. Aunque sólo vayas a tener Linux en un disco determinado, también es aconsejable hacer diversas particiones puesto que necesitarás al menos una partición de *swap*.

Es aconsejable tener, al menos, las siguientes particiones en Linux:

- Una partición para el directorio raíz (/), esto es el sistema de ficheros principal. Esta partición es absolutamente necesaria para que el sistema pueda funcionar. Su tamaño dependerá de la cantidad de disco que hayamos asignado a Linux, pero lo razonable sería al menos 2Gb (conviene reservar entre 3 y 4Gb) para una

instalación con numerosos paquetes de software. No obstante, dependiendo de la cantidad de programas que vayamos a instalar en Linux incluso podríamos necesitar más.

- Una partición para *swap*. Se utiliza para la memoria virtual. Su tamaño dependerá de la cantidad de memoria real que se tenga. Una regla útil es tener como mínimo el mismo tamaño que la memoria real y como mucho el doble. Un espacio superior a los 300MB normalmente resultará excesivo (incluso aunque tengamos 512MB o más de memoria RAM).
- Una partición para los datos de usuario (el directorio `/home`). El tenerla separada de la partición del sistema de ficheros principal (/) permite que sea más fácil cambiar de distribución o hacer copias de seguridad. Si lo vas a usar para prácticas, entre 200Mb y 1Gb es suficiente. Si lo vas a usar para aplicaciones multimedia, reserva mucho más.

¿Sabías que . . . ?

Los requisitos mínimos teóricos para instalar Linux en un PC son muy básicos: procesador 386 o superior, 4MB de memoria RAM y disquetera.
En la práctica, para poder hacer algo útil, las distribuciones suelen necesitar algo más: 8MB sería el mínimo aceptable y 16MB lo aconsejable para utilizar las X – servidor de gráficos – y disco duro con un mínimo de unos 400 MB (variará según lo que instalemos, es decir según distribuciones y el tipo de instalación que se haga).
Prácticamente cualquier placa base funciona. Se puede utilizar prácticamente cualquier unidad de CDROM (aunque no es necesario, es muy recomendable instalar desde CDROM).

Vamos a comentar algunas cosas más acerca de las particiones en un disco duro. La información acerca de las "partes" en las que se ha dividido el disco duro se guarda en la *tabla de particiones*, la cual se almacena en el sector 0 del disco. No puedes crear cuantas particiones quieras. Hay una serie de restricciones:

- En el sector 0 del disco sólo hay espacio para almacenar información relativa a 4 particiones, llamadas *primarias*.
- Una de las primarias puede ser una partición *extendida*, esto es una "caja" que mantiene la descripción de una serie de particiones que se llaman *lógicas*.
- A las 4 particiones primarias, hayan sido o no definidas, se les asignan los números del 1 al 4. Las particiones lógicas se numeran del 5 en adelante.

Cuando instalas Linux, arrancas de un CD-ROM. El CD-ROM entra rápidamente en el modo de particionado y puede que te sugiera una distribución de particiones razonable. En cualquier caso debes saber que efectuar particiones es una acción potencialmente peligrosa. Antes de instalar Linux, haz una copia de seguridad de la información que tengas en Windows.

Más sobre particiones

El esquema de particiones descrito es válido *sólo* para la arquitectura PC.
Distintas arquitecturas presentan formas *diferentes* de particionar los discos. En discos para sistemas BSD/SUN sólo se pueden hacer 8 particiones siendo la tercera un caso "especial"; los discos para sistemas IRIX/SGI admiten un máximo de 16 particiones, siendo la novena y la undécima casos "especiales".
Las particiones pueden realizarse con el programa `fdisk`. Es aconsejable utilizar el de Linux que es más completo y manejable y no el de MS-DOS, que no entiende que existen otros sistemas operativos. No obstante, las distribuciones suelen ofrecer otras alternativas (programas en modo gráfico) que funcionan bastante bien.

continúa en la página siguiente . . .

. . . viene de la página anterior

Si, como es habitual, el disco está ocupado por Windows, se puede utilizar `FIPS` para cambiar el tamaño de la partición sin tener que formatear. `FIPS` es un programa DOS capaz de *dividir* (solamente) particiones del disco duro sin destruir datos. Primero se desfragmenta el disco y después se utiliza `FIPS`. Como siempre que se utilizan programas de esta naturaleza, conviene leer con detenimiento las instrucciones antes de usarlo. Además el manual resulta muy instructivo.

`PartitionMagic` es otro programa muy interesante, aunque éste comercial. Este programa es capaz de *crear*, *redimensionar*, *dividir*, *unir*, *recuperar* y *cambiar* de tipo particiones del disco duro sin destruir datos. También es capaz de gestionar el arranque de varios sistemas operativos (Linux incluido).

A destacar: interfaz gráfica, lo cual permite gran facilidad de uso, soporte para discos de más de 20 GB, soporte de `ext2` (sistema de ficheros de Linux) y potentes herramientas para chequeo de discos. Existen versiones para Windows 95/98, Windows 2000, Windows Me, Windows NT 4.0. y Windows XP.

También desde el "rincón" del software libre existen alternativas interesantes, aunque no tan desarrolladas. Concretamente, hay un programa llamado `parted` que es de GNU. Este programa permite crear, destruir, redimensionar, mover y copiar particiones. Es útil para dar espacio a nuevos sistemas operativos, reorganizar el disco y copiar datos a nuevos discos. Recientemente se están desarrollando programas que ofrecen una intuitiva interfaz gráfica para manejarlo.

En cualquier caso, es mejor, si se tiene la oportunidad, preparar el espacio antes de instalar nada.

Durante el proceso de instalación se configuran una serie de aspectos del sistema. Estas configuraciones pueden hacerse antes, durante o después del proceso de instalación de paquetes. Algunas de las configuraciones más típicas incluyen:

- Elegir un password para el superusuario, o `root`.
- Dar de alta a un usuario en el sistema para el trabajo "habitual" (no es bueno, ni debe hacerse por motivos de seguridad, trabajar siempre como `root`).
- Detectar y configurar hardware (tarjeta de sonido, tarjeta ethernet, etc.)
- Configurar el acceso a Internet.
- Configurar la interfaz gráfica de usuario (detección de la tarjeta gráfica, configuración del servidor X).
- Preparar adecuadamente LILO (LInux LOader) para el arranque de Linux (y posiblemente Windows).

Los usuarios

Para añadir usuarios al sistema hay distintas opciones (y siempre ejecutándolas como `root`).

Editar directamente los ficheros `/etc/passwd` y `/etc/group` y crear manualmente las entradas y directorios correspondientes. No se recomienda a no ser que se sepa muy bien lo que se está haciendo.

Otra es utilizar órdenes especificas. Para añadirlos: `adduser`, `addgroup`. Para eliminarlos: `userdel`, `groupdel`.

Finalmente, es posible emplear programas como `linuxconf` o los programas gráficos de gestión de usuarios que proporcionan las distribuciones, los cuales permiten utilizar ventanas y menús.

Vamos a hablar con más de detalle sobre el gestor de arranque LILO. Debes saber que Linux puede convivir tranquilamente con otros sistemas operativos. Existen distintas maneras de conseguir esta convivencia, una de las más cómodas es utilizar *LILO* (LInux LOader). Cuando Linux se instala con éxito, suele permitir la selección del sistema operativo en el arranque mediante este programa. La idea básica es que LILO se carga al principio, dando al usuario la oportunidad de decidir cuál de los sistemas presentes en la máquina quiere arrancar. Además, LILO es capaz de arrancar sistemas operativos que no estén necesariamente en una *partición primaria*.

Más sobre LILO

LILO tiene diversas opciones que permiten fijar cuánto tiempo se espera para que el usuario decida qué quiere arrancar o qué parámetros se pasan a un sistema determinado (Linux admite que se le den ciertas opciones al arrancar).
Para configurarlo se utiliza el fichero `/etc/lilo.conf` o algún programa como `linuxconf`; aunque resulta conveniente leerse la documentación en `/usr/doc/lilo` y ejecutar `man lilo` y `man lilo.conf`.
Un problema que tiene LILO es que utiliza la BIOS para leer la geometría del disco, lo que con BIOS antiguas y discos grandes puede dar problemas. Por eso se aconseja que las particiones de arranque (las que contienen los ejecutables de los sistemas operativos) estén dentro de los primeros 1024 cilindros (8 GB), a no ser que tengas una BIOS moderna.
Algunas veces los problemas pueden resolverse fijando el modo *LBA* (Linear Block Access) de acceso a disco en la configuración de la BIOS y/o usando la opción *linear* en la configuración de LILO.
En la actualidad se recomienda usar la opción *lba32* en sistemas con BIOS “modernas” (equipos comprados con posterioridad a 1998). Esto permite arrancar sistemas instalados en particiones más allá del cilindro 1024.

Ten en cuenta que, normalmente, desde Linux podrás acceder al contenido de los discos y particiones Windows. Salvo en el caso del sistema NTFS (*sólo lectura*), puedes leer y escribir ficheros. Desde Windows *no* puedes ver el contenido de los discos y particiones Linux, a no ser que utilices programas específicos ajenos a Microsoft. ¡Ojo! Si instalas Windows después de instalar Linux (cosa más frecuente de lo que uno quisiera), hay varios peligros potenciales:

- Windows puede reasignar particiones o instalarse en un partición Linux, destruyendo su contenido.
- Windows puede borrar a LILO.

Por tanto, es conveniente que crees un disquete de arranque Linux antes de instalar Windows (el disquete de arranque se crea surante la instalación). En cualquier caso, asegúrate antes de haber hecho una copia de seguridad de los datos importante que tengas en Linux. En resumen, el proceso de instalación de Linux sobre una máquina con Windows o de Windows sobre una máquina con Linux es *potencialmente peligroso*. Otras fuentes potenciales de problemas:

- Hardware muy reciente: Linux puede no disponer todavía de controladores.
- Ventanas (sistema X Window): algunas tarjetas de vídeo son problemáticas. Consulta si tu tarjeta está soportada en Linux.
- Dispositivos USB que no siguen los estándares (cosa frecuente en los módems USB).
- Módems WinMódem (son módems internos incompletos que no se ajustan a estándares).

Coexistencia de los SSOO

Linux no tiene problema en coexistir con otros sistemas operativos. Sin embargo son interesantes algunas observaciones.
Esta buena vecindad no siempre es correspondida. Por ejemplo, algunas versiones de Windows cuando se instalan asumen que son los únicos sistemas; ello significa que escriben en el sector de arranque del disco "sin mirar" si había algo antes (o sin importarles que se pueda perder lo que antes había). Además exigen estar en el disco primario. Solución: instalar primero Windows (y darle la *primera* partición del *primer* disco) y después Linux. No obstante, antes de instalar Windows, se deberían hacer las particiones, si no Windows se "queda" con todo el disco.
Linux entiende gran cantidad de sistemas de ficheros (p.e. FAT, FAT extendido, HFS o Hierarchical File System de Macintosh, NTFS, etc.).
Si se tienen problemas con LILO, es posible arrancar Linux utilizando LOADLIN[13], lo que en algunos casos es una solución sencilla y cómoda. Éste es un programa DOS que, básicamente, es un *cargador* de imágenes del kernel de Linux. Es altamente adaptable a diferentes configuraciones de DOS y ahora tiene muy pocas restricciones.
Se pueden ejecutar otros sistemas en una "máquina virtual" utilizando `vmware`.
La máquina virtual es un software que *emula* una *computadora personal* (PC). Esto es, un ordenador con sus discos duros, su tarjeta de sonido, su unidad de CD-ROM, su disquetera, su memoria RAM, su tarjeta gráfica, su propia BIOS y, por supuesto, su *procesador*, incluyendo soporte completo (emulación) del juego de instrucciones x86 de Intel.
Podemos configurar varias máquinas virtuales (múltiples PCs) dentro de nuestro ordenador *real*. Por supuesto, en cada uno de estos PCs virtuales podemos instalar el SO que queramos.
El SO instalado y las aplicaciones que, sobre él, instalemos se ejecutarán *dentro* de la máquina virtual, usando el juego de instrucciones del procesador emulado.
La emulación del disco duro se realiza sobre un fichero. Este fichero irá creciendo *dinámicamente* hasta alcanzar el tamaño máximo establecido en la configuración de la correspondiente máquina virtual. Ello implica que todos los cambios se pueden deshacer fácilmente.
`vmware` es comercial. Es necesario adquirir una licencia para poder usarlo. En `www.vmware.com` podéis descargaros versiones de prueba (30 días). Existen versiones para Windows y para Linux. Sin embargo, existen alternativas desde el software libre, *aunque no tan completas y funcionales*.
`plex86`, popularmente conocido como `FreeMWare`, `savannah.nongnu.org/projects/plex86`. El objetivo de este proyecto es crear un programa libre y extensible para virtualización de PC, el cual permitirá a los usuarios de PCs y estaciones de trabajo ejecutar múltiples SSOO concurrentemente en la misma máquina. `FreeMWare` ejecutará el SO y las aplicaciones tan nativamente como sea posible, el resto será emulado por el monitor de virtualización de PC. Esto no es un emulador de PC como `vmware`.
`bochs`, `bochs.sourceforge.net`. Éste sí es idéntico, en cuanto a concepto a `vmware`, ya que es un emulador de PC (escrito en C++). Incluye emulación de los procesadores x86 de Intel (386, 486 o Pentium), soporte para dispositivos de entrada y salida comunes y una BIOS propia. Actualmente es capaz de ejecutar Linux, Windows 95, MS-DOS y Windows NT 4 en la emulación. Existen versiones para Windows y para Linux.
Finalmente, una alternativa a LILO que esta creciendo día a día en popularidad (además es GNU :-): GRUB (GRand Unified Bootloader), un "cargador" que pretende unificar el proceso de arranque de los SSOO en PCs: `www.gnu.org/software/grub/`. Características más destacadas:
Carga de múltiples imágenes (interesante para sistemas *modulares* como *Hurd*).
Puede cargar Linux, Hurd, DOS, Windows, OS/2, etc.

continúa en la página siguiente ...

[13] Éste es el sistema utilizado en las aulas informáticas.

. . . viene de la página anterior
Ofrece una interfaz amigable (menús) para gestionar la carga de SSOO.
Soporta múltiples sistemas de ficheros.
Independencia de la geometría del disco que proporciona la BIOS. Si ésta soporta LBA y el disco puede ser accedido mediante esta técnica, *no hay problema* en arrancar imágenes situadas más allá de los 8 Gb (cilindro 1024).
Arranque de imágenes desde red local.

Apéndices (información adicional, no para examen)

Compilación del kernel

Entendemos por kernel el *núcleo* del sistema operativo, esto es, su parte principal. Normalmente, no tendremos que compilar el código fuente (las instrucciones del programa) del kernel, ya que el que viene ya compilado e incluido en las distribuciones suele estar preparado para gran cantidad de hardware distinto. Sin mebargo, esto hace que tenga gran tamaño y no sea el más eficiente para nuestro sistema (aunque esto lo solucionan las distribuciones utilizando módulos con todas las partes del kernel que así lo permitan).

Puede suceder que adquiramos nuevo hardware, tengamos nuevas necesidades o consigamos una nueva versión del kernel, la cual deseamos instalar (por ejemplo, porque resuelve ciertos problemas de seguridad). En cualquier caso, pudiera ser que necesitásemos compilar el kernel. A la hora de compilar el kernel hay que tener bastante claro qué hardware tenemos y cómo queremos utilizarlo. Los pasos que hay que seguir son:

- Antes de nada leer la documentación, concretamente el `Kernel-HOWTO`.
- Configurar el kernel.
- Hacer la compilación propiamente dicha.
- Compilar e instalar los módulos.

Lógicamente, esto hay que hacerlo como superusuario y en el directorio correspondiente (`/usr/src/linux` suele ser un enlace simbólico al directorio de la versión que toca del kernel).

Configuración del kernel

La configuración del kernel consiste en ir decidiendo qué características queremos incluir en él, p.ej: capacidad para manejar discos SCSI, impresoras en el puerto paralelo, controlar sistemas multiprocesador o incluir soporte para nuestra tarjeta de sonido. Hay tres opciones:

- `make config`: es la más primitiva. Va preguntando las opciones una a una. Hoy día prácticamente no se utiliza.
- `make menuconfig`: se utilizan menús en modo texto. Es bastante cómoda y más flexible que la anterior.
- `make xconfig`: la configuración se realiza con ayuda de ventanas y botones. Probablemente sea la más aconsejable.

Con cualquiera de estas maneras, se puede acceder a una ayuda que aconseja acerca de cada opción del kernel.

Módulos

Hay determinadas partes del kernel que se pueden configurar como módulos: partes que, bajo demanda, pueden cargarse dinámicamente en memoria y descargarse automáticamente mientras el kernel está ejecutándose. Algunas ventajas de utilizar módulos son:

- El kernel puede ser más pequeño y cargar sólo aquellos módulos que realmente utilice (si usamos LILO, el kernel no puede exceder de un cierto tamaño, p.e.).
- La carga y descarga de los módulos puede hacerse de forma automática.
- Determinados periféricos (p.e. SCSI) necesitan estar encendidos en el momento de inicializarse el controlador. Si se utilizan módulos es posible no tener los dispositivos encendidos desde el arranque.

Compilación

Una vez configurado el kernel, hay que realizar la compilación propiamente dicha. Los pasos necesarios son:

- `make dep`: con esto, el sistema encuentra las dependencias entre las distintas partes del kernel.
- `make clean`: esta orden limpia los ficheros que puedan quedar de compilaciones anteriores. Esto es fundamental para asegurar que estamos compilando todo lo que toca. Por ello existen otras opciones "más fuertes", como `make mrproper` la cual "hace una limpieza más profunda"[14].

Finalmente creamos el fichero que contendrá el kernel. Las opciones son:

- `make zImage`/`make bzImage`: crea una imagen del kernel en un fichero, que después hay que colocar donde interese.
- `make zdisk`/`make bzdisk`: crea un disco de arranque con el nuevo kernel.
- `make zlilo`/`make bzlilo`: crea el nuevo kernel y ejecuta `lilo`.

La diferencia entre la forma `z*` y la `bz*` (ambas indican el tipo de compresión, ya que la imagen del kernel se guarda siempre comprimida) es que la segunda permite kernels de mayor tamaño, ya que el nivel de compresión es mayor.

Compilación e instalación de los módulos

Ésta es la parte final. Basta con:

- `make modules`: compila los módulos.
- `make modules_install`: copia los módulos en el sitio adecuado.

Después es conveniente utilizar la orden `depmod -a` para que el sistema sepa que relación tienen entre sí los distintos módulos.

Fuentes de documentación

En este tema sólo hemos hecho una somera introducción a Linux. Existen gran cantidad de aspectos que no hemos tratado y que se pueden encontrar en distintos sitios. En `/usr/share/doc` (o `/usr/doc`) existe una carpeta por cada uno de los paquetes instalados en el sistema con información sobre su uso, configuración, etc.

Muchos de los directorios se refieren a programas concretos, pero dentro de `/usr/share/doc` encontrarás un `FAQ` y un `HOWTO` (el lugar específico depende de la distribución), donde hay muchos documentos sobre Linux en general. Además existe mucha información en forma de *info* accesible, por ejemplo, desde Emacs.

En particular, recomendamos la utilización del programa `dwww` que permite *navegar* (y hacer búsquedas específicas) a través de la documentación del sistema usando Netscape, p.e. El único "inconveniente" es que requiere la instalación de un servidor `http` (Apache, p.e.), pero recuerda que no hace falta tener una conexión permanente (ni siquiera temporal) a Internet para instalar en el ordenador un servidor web.

Existen muchos grupos de *news* relacionados con Linux: los de la jerarquía `comp.os.linux` y en español los de `es.comp.os.linux`. Indirectamente, también están los relacionados con GNU (`gnu.*`).

También hay distintas *listas de correo*. Un ejemplo es *l-linux*. Para subscribirse, basta con mandar un mail a `majordomo@calvo.teleco.ulpgc.es` con el texto `subscribe l-linux` en el *cuerpo* del mensaje, no hace falta que pongas nada como *subject*. Muchas distribuciones tienen también sus propias listas de correo (p.e. Debian). A través de sus páginas web puedes consultarlas, suscribirte, ...

Entornos gráficos de trabajo

La interfaz gráfica de usuario de los sistemas con Linux se basa en el entorno Xwindow. Existen diversas implementaciones de las X, la más común en las distribuciones es la XFree86, que es una versión libre basada en X11R6 (versión 11 de las X, la primera ampliamente aceptada y difundida, release 6).

Uno de los principales principios de diseño del sistema Xwindow y del propio UNIX, es que la funcionalidad se consigue mediante la *cooperación* de componentes separados en lugar de tener un único componente que haga

[14] Esto *no* es una broma.

todo el trabajo. La principal ventaja de esta filosofía es que una parte del sistema puede ser fácilmente cambiada simplemente reemplazando la componente adecuada. En el caso de UNIX, tenemos por ejemplo el intérprete de comandos (fácilmente cambiable por otro programa distinto del que viene por defecto).

En el caso de las X, el mejor ejemplo de esto es el concepto de *gestor de ventanas*, el cual es esencialmente el componente que controla la apariencia de las ventanas y proporciona los medios para que el usuario pueda interactuar con ellas. Prácticamente todo lo que aparece en la pantalla con las X está dentro de una ventana y un gestor de ventanas simplemente las *maneja*. La gente es diferente y usa los ordenadores de formas diversas y para tareas distintas ¿Por qué pensar que todos tenemos que usar la misma interfaz? Afortunadamente, las X no padecen de esta rigidez y se han desarrollado un gran número de gestores de ventanas, los cuales proporcionan no sólo diferentes aspectos, sino también diferentes "comportamientos". Es más, muchos de estos programas son altamente configurables y adaptables a los deseos del usuario.

Más aún, recientemente se han desarrollado los primeros *entornos de escritorio* (p.e. GNOME y KDE). Estos programas estarían un nivel por encima de los gestores de ventanas (de hecho, se encargan de manejarlos). Pretenden proporcionar una interfaz gráfica de trabajo mucho más completa tanto con el sistema operativo como con las diversas aplicaciones que podamos manejar. Además proporcionan su propio conjunto de utilidades y aplicaciones integradas en el entorno de trabajo (mismo aspecto y forma de funcionar). El trabajo se reparte en tres niveles:

- el servidor X,
- el gestor de ventanas y
- el entorno de escritorio.

Servidores X

Los servidores X son una serie de programas que interactúan con la tarjeta de vídeo y ofrecen a los programas que están por encima las facilidades básicas para el manejo de ventanas, uso de tipos de letra, dibujo, generación de eventos de los dispositivos de entrada de datos (teclado, ratón, ...), etc. Normalmente, las distribuciones (como SuSE) incluyen programas que detectan la tarjeta y configuran el servidor X de forma adecuada. Por si acaso algo fallase, te ofrecemos esta pequeña guía.

Según la tarjeta de vídeo que tengamos, tendremos que elegir uno u otro servidor. Para ello resulta conveniente mirar la documentación, `/usr/X11R6/lib/X11/Cards` (su ubicación puede variar en función de la distribución). Una versión más actualizada se puede encontrar en la web de las X, `www.xfree86.org`. Datos que son importantes conocer para su configuración: Nombre, chipset y fabricante de la tarjeta y la memoria que ésta tiene. Por ejemplo: ATI Mach64 3D RAGE II, fabricante ATI, chipset ati, memoria 32 Mb. Para más detalles: `man XF86Config`.

Una vez tenemos el servidor, es necesario configurarlo mediante el archivo `XF86Config` (versión 3.3.6) o `XF86Config-4` (versiones 4.*.*). Esto se puede hacer a mano o con programas de configuración específicos. Un programa que siempre "está", aunque funciona en modo texto, es `xf86config`. Funciona respondiendo a una serie de preguntas y eligiendo las opciones adecuadas de entre todas las posibles.

Herramientas "gráficas". Suelen ser específicas de las distribuciones: `xf86cfg`, `XFDrake` (Mandrake), `SaX2` (SuSe, muy buena). Es posible probar el efecto de la configuración "en directo" utilizando el programa `xvidtune`. Los servidores X suelen sacar bastante provecho de la tarjeta y con un poco de *hacking* podemos tener resoluciones y frecuencias de refresco muy aceptables (el `XFree86-Video-Timings-HOWTO` es una lectura dura pero muy aconsejable).

Recientemente, las X han sufrido un cambio considerable. Se ha pasado de la versión 3.3.6 a una nueva rama de desarrollo, la 4. Cuestiones importantes:

- La arquitectura ha sido completamente rediseñada en la versión 4.
- Introduce un sistema *modular*, donde se requiere especificar los módulos deseados en el `XF86Config-4`.
- Los módulos se cargan *dinámicamente* en memoria bajo demanda por medio del servidor X.
- Sólo hay un *único programa* que actúa como servidor X independientemente de la tarjeta que tengamos (esto se ha unificado), en contraposición a la lista de programas *dependientes de la tarjeta* que podían actuar como servidores X en la versión 3.
- El nuevo servidor "unificado" soporta la mayoría de tarjetas modernas de la mayoría de fabricantes e incluye, además, la mayor parte de las tarjetas soportadas en la versión 3 (pero no *todas*).

- Es por eso que alguna tarjeta soportada en la versión 3, no lo esté aún en la versión 4. Antes de hacer el cambio, debemos asegurarnos.
- Información sobre los módulos y sus posibilidades, `man XF86Config` y `www.xfree86.org`.
- Otra "puerta que se abre" en Linux con la versión 4 de las X es la posibilidad de usar la aceleración 3D de nuestra tarjeta (caso de que la tenga). Esto es útil para programas de CAD, multimedia (películas), juegos, ...
- Para ello debemos especificar en la sección `"Module"` del `XF86Config-4` las líneas `Load "glx"` y `Load "dri"` (cargar módulos GLX y DRI).
- Además hay que cargar el módulo del kernel (rama 2.4) de Linux específico para nuestra tarjeta (la aceleración la proporciona realmente el kernel). Existen fabricantes, como Nvidia, que proporcionan los *drivers* específicos para el kernel en su página web.
- No todas las tarjetas 3D soportadas en la versión 4, tienen soporte para aceleración en el kernel. En algunos casos, hay que esperar ...
- Antes de hacer nada *conviene consultar*: `/usr/share/doc/xserver-xfree86/README.DRI.gz` y `dri.sourceforge.net`.

Gestores de ventanas

Los manejadores de ventanas son los encargados de hacer cosas como decidir los bordes de las ventanas, manejar los eventos (teclas, ratón, etc.) que genera el servidor X, etc.

Hay bastantes manejadores, algunos comerciales con cierto prestigio como *Motif* y otros libres como *fvwm*. Cada uno de ellos tiene bastantes aspectos que se pueden personalizar. Entre los más populares: *fvwm, twm, Sawfish, Icewm, WindowMaker, AfterStep, Enlightenment,* ...

Entornos de escritorio

Se encargan de proporcionar la interfaz última de trabajo con el usuario. Vienen con una serie de programas de aspecto y comportamiento uniforme y coherente, con el objeto de facilitar el trabajo del usuario. Además, ofrecen, entre otras, herramientas de:

- Atención al terminal (*login*) en modo gráfico,
- Gestión de acciones "arrastrar y soltar" a nivel de escritorio y entre aplicaciones que las soporten,
- Barras de tareas, menús flotantes y contextuales e iconos en el escritorio,
- Acceso a menús desde los cuales se pueden lanzar la mayoría de las aplicaciones instaladas en el sistema,
- Manejo del gestor de ventanas en lugar del usuario (con posibilidad de cambiarlo de entre una lista de opciones dada por el usuario),
- Configuración de todos los aspectos del entorno (incluyendo invocar a las aplicaciones gráficas de configuración del gestor de ventanas, etc.).
- Destacan GNOME y KDE (además, son software libre).

GNOME El proyecto GNOME pretende construir un escritorio completo y amigable basado enteramente en software libre. No tiene un gestor de ventanas propio, aunque, por defecto, sugiere *Sawfish*. Existen otros compatibles: *Enlightenment, Icewm* y *WindowMaker*, aunque solapan ciertas funciones con GNOME y conviene deshabilitarlas. Está basado en las librerías GTK como las aplicaciones que incluye, ofreciendo de esta forma un aspecto y comportamiento coherente. GNOME forma parte del proyecto GNU.

KDE Proporciona una interfaz consistente para las aplicaciones X, tanto en apariencia como en funcionalidad. KDE contiene un conjunto básico de aplicaciones tales como un gestor de ventanas (*kwm*), gestor de archivos, emulador de terminal, sistema de ayuda y configuración de la pantalla. KDE especifica un *toolkit GUI* estándar así como unas librerías gráficas, las QT, que usan todas las aplicaciones. Han publicado una guía de estilo que recomiendan seguir a todos los desarrolladores de aplicaciones KDE. La última versión incluye navegador integrado en el sistema de ficheros y un conjunto integrado de herramientas ofimáticas.

Direcciones de interés para Linux

Finalmente, podemos comentar que hay numerosas páginas *web* dedicadas a Linux. A continuación, algunas URLs de interés.

En castellano:

- `http://calvo.teleco.ulpgc.es/`: con mirrors de páginas sobre Linux (p.ej. del *Linux Documentation Project*).
- `http://slug.ctv.es/`: *home* del grupo español de usuarios de Linux.
- `http://sunsite.rediris.es`: tiene linux y otras cosas.
- Barrapunto, `http://barrapunto.com`: contiene noticias varias sobre Linux y software libre, secciones sobre temas específicos y acceso a URLs relacionadas.
- Proyecto Lucas, `http://lucas.hispalinux.es`: la mayor biblioteca en español dedicada a GNU/Linux de todo el planeta (dicen). Lo que sí he comprobado es que hay mucha información, incluyendo manuales a todos los niveles y todo en castellano.
- ORCA, `http://quark.fe.up.pt/orca/`: documentación sobre Linux y aplicaciones varias. Acceso a grupos de traducción de documentación al castellano.
- `http://r34linux.virtualave.net`: FAQ (preguntas frecuentemente respondidas) sobre Linux de Fidonet España.
- `http://web.jet.es/s.romero/`: tiene una sección muy interesante sobre Linux.
- `http://segurinet.com/gsal/default.htm`: guía de seguridad para el administrador de Linux.
- `http://www.gulic.org/cosecha/lcabrera/M.I.L/`: un manual de introducción a Linux.
- `http://projects.openresources.com/`: apuntes sobre software libre.
- `http://www.es.kernel.org/`: acceso a varias versiones del kernel de Linux.
- Proyecto La Espiral, `http://quark.fe.up.pt/laespiral/`: documentación, software, manuales, etc. relacionada con Debian.

En inglés tenemos:

- `http://www.linuxhq.com`: con información sobre el kernel.
- `http://www.linux.org`: un poco de todo.
- `http://www.linuxdoc.org/`: un montón de documentación acerca de Linux y acceso a otras URLs de interés.
- `http://www.linuxpress.com/`: acceso a recursos, libros y publicaciones sobre Linux.
- `http://mercury.chem.pitt.edu/~tiho/LinuxFocus/`: revista on-line sobre Linux (versión en castellano disponible).
- `http://www.gnome.org/`: página principal del proyecto GNOME.
- `http://www.gnu.org/`: página principal del proyecto GNU.
- `http://www.linux.org/hardware/laptop.html`: cómo instalar Linux en un montón de portátiles.
- `http://www.cs.utexas.edu/users/kharker/linux-laptop/`: más sobre Linux en portátiles.
- `http://pcmcia-cs.sourceforge.net/`: información sobre PCMCIA (las tarjetas que se usan en los portátiles).
- `http://users.bart.nl/~patrickr/hardware-howto/Hardware-HOWTO.html`: este documento muestra la mayoría del hardware soportado por Linux y te ayuda a localizar los drivers necesarios.

- `http://www.o2.net/~gromitkc/winmodem.html`: algunas consideraciones a tener en cuenta a la hora de elegir un módem para Linux.
- `http://www.linuxprinting.org/`: todo acerca de la impresión desde Linux.
- `http://www.linmodems.org/`: cómo hacer funcionar nuestro módem en Linux (si se puede).
- `http://freshmeat.net/`: para buscar software de todo tipo (y de lo más variopinto) para Linux. Si no sabes si un determinado programa de Windows existe en Linux, éste es un buen sitio para empezar a averigüarlo.
- `http://lwn.net/`: las últimas noticias del mundo Linux.
- `http://www.securityportal.com/`: cuestiones relacionadas con la seguridad en los sistemas.
- `http://stjohnchico.org/~jtmurphy/`: más sobre seguridad (en este caso cuestiones relacionadas exclusivamente con Linux).
- `http://www.PLiG.org/xwinman/`: información sobre gestores de ventanas y entornos de escritorio para X y acceso a URLs relacionadas.
- `http://www.windowmaker.org/`: página oficial del gestor de ventanas WindowMaker.