2 SISTEMAS OPERATIVOS

Con el término entorno operativo englobamos al sistema operativo, a su interfaz de usuario asociado y a algunas aplicaciones que suelen venir con él (administrador de archivos, programas de configuración y optimización y otros).

El sistema operativo es el software básico que controla una computadora. A grandes rasgos tiene tres grandes funciones: coordinar y manipular el hardware del sistema informático (memoria, impresoras, unidades de almacenamiento, periféricos, etc.), organizar los archivos en los dispositivos de almacenamiento y gestionar los diferentes errores que se generen.

2.1 CONCEPTOS GENERALES. TIPOS Y CLASIFICACIÓN DE SISTEMAS OPERATIVOS

Cualquier sistema operativo actual está en completa evolución. Se puede decir que la cuota de mercado de los sistemas operativos de sobremesa se la llevaría Windows®, Linux y en menor medida el sistema operativo de Apple®.

Figura 2.1. *Logos de sistemas operativos libres basados en Linux*

Los sistemas operativos antes citados no son funcionales en los dispositivos móviles como puedan ser *smartphones*, MID, *tablets*, PDA, etc. Estos dispositivos necesitan un sistema operativo ligero, y que cuente con muchos de los servicios que proporcionaban los sistemas operativos tradicionales. Entre estos sistemas operativos móviles destacan Android, Ubuntu phone, BoottoGecko (B2G), BADA, MeeGo, Symbian OS, Blackberry OS, iPhone OS y Windows® Phone.

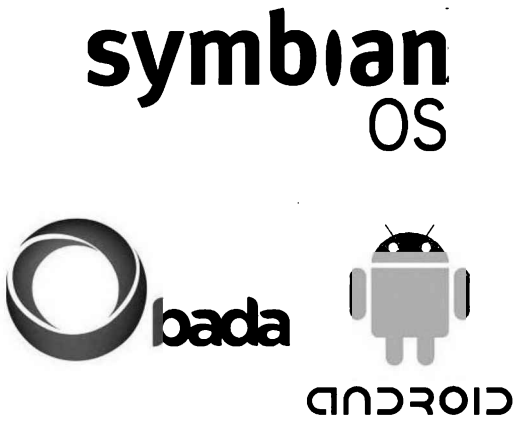

Figura 2.2. *Logos de sistemas operativos libres de dispositivos móviles*

¿SABÍAS QUE...?

Google Chrome está pensado para *netbooks* y es código abierto. El interfaz de usuario es mínimo ya que está pensado básicamente para trabajar sobre Internet. Pretende rediseñar la arquitectura de seguridad que lo soporta para estar blindado de virus y de software malicioso.

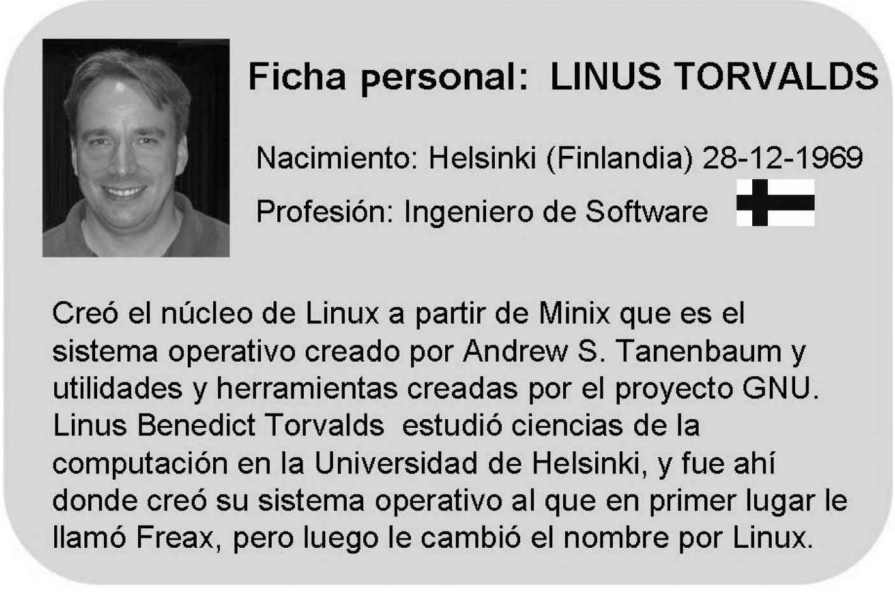

Figura 2.3. *Ficha personal de Linus Torvalds. Fotografía cortesía de Kylehase*

Hay que decir que muchos sistemas operativos móviles basan su filosofía en Linux. Esto tiene mucho de lógico dado que es un sistema operativo libre y cualquier desarrollo partiendo desde cero sin tener en cuenta lo ya realizado por Linux no tiene mucho sentido.

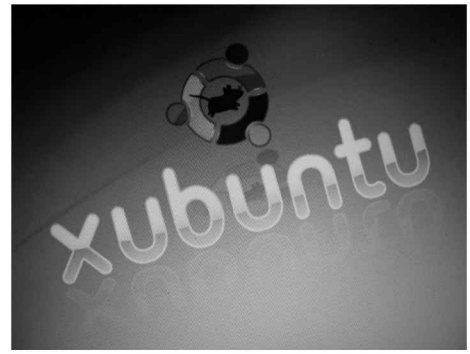

Figura 2.4. *Instalación de XUbuntu. Fuente: Rafa Espada*

El futuro de los sistemas operativos está basado en el *cloud computing* trasladando lo que eran los servicios que ofrecía nuestro sistema a Internet. En un futuro será Internet la que proveerá de servicios y almacenará los datos de los clientes. Los equipos informáticos harán de caché y trabajarán con esos datos pero su almacenamiento realmente estará en servidores de Internet. Habrá muchos servicios basados en la web y los clientes podrán acceder a ellos teniendo un abanico de posibilidades muy variado y seguramente se establecerá el sistema de pago por consumo.

Los sistemas operativos del futuro se van a basar cada vez más en el navegador (el primero en girar hacia esta filosofía fue Google Chrome y Windows® 8 está desarrollado de esta manera). Muchas personas, aunque reconocen las bondades de esta tecnología, no son partidarias de esta idea pues se genera una dependencia de los proveedores de servicios y del acceso a Internet.

2.1.1 TIPOS Y CLASIFICACIÓN DE SISTEMAS OPERATIVOS

Existen numerosas formas de clasificar los sistemas operativos, atendiendo a diferentes criterios, las clasificaciones más usuales son las siguientes:

Clasificación de los sistemas operativos según su propósito:

- **Propósito general**. Se caracterizan por tener un gran número de usuarios trabajando con un número variado de aplicaciones.
- **Propósito específico**. Se usan en entornos donde se aceptan y procesan, en poco tiempo, un gran número de sucesos, en su mayoría externos al ordenador. Por ejemplo: control industrial, control de vuelo, equipamiento telefónico conmutado o simulaciones en tiempo real.

Clasificación de los sistemas por el número de usuarios:

- **Monousuario**. Son los que soportan un único usuario a la vez.
- **Multiusuario**. Son los que dan servicio a múltiples usuarios simultáneamente.

Clasificación de los sistemas operativos por el número de tareas:

- **Monotarea**. Son aquellos que solo permiten al usuario realizar una tarea a la vez.
- **Multitarea**. Permiten al usuario realizar múltiples tareas de forma simultánea.

Clasificación de los sistemas operativos por la forma como ofrecen los servicios.

- **Centralizados**. En este modelo un ordenador central se encarga de todo el procesamiento y los usuarios se conectan a él a través de terminales que carecen de memoria y procesador.
- **Distribuidos**. Permiten distribuir los trabajos, tareas o procesos entre un conjunto de procesadores, que pueden estar en el mismo equipo o en equipos distintos.
- **De escritorio**. Es el utilizado habitualmente por los equipos de sobremesa, estaciones de trabajo y portátiles.
- **En red**. Son los que permiten mantener unidad entre dos o más ordenadores a través de algún medio de comunicación para poder compartir los recursos y la información del sistema.

Clasificación de los sistemas operativos según el tiempo de respuesta:

- **Tiempo real**. Son los que permiten que el ordenador dé una respuesta inmediata tras lanzar un proceso.
- **Tiempo compartido**. Son los que permiten que varios usuarios individuales interactúen a la vez, de forma que cada usuario tenga la sensación de que se le está atendiendo en exclusiva, aunque, en realidad, cada tarea tiene un nivel de prioridad y se ejecuta en orden secuencial.

2.1.1.1 LOS SISTEMAS OPERATIVOS MÓVILES

Figura 2.5. *Estructura de los sistemas operativos móviles*

Los sistemas operativos, aunque son distintos entre sí, todos ofrecen una estructura más o menos común con distintas capas. Pasaremos a comentar cada una de ellas:

- **Kernel**. Generalmente los *kernel* o núcleos están basados en Unix. En el caso de Android está basado en el *kernel* de Linux mientras que iOS está basado en el *kernel* de Mac OS. A estos *kernel* se les ha reducido al mínimo para que puedan operar en dispositivos móviles. El *kernel* se encargará de manejar el hardware y controlará los *drivers*, gestionará la memoria, los procesos, los archivos, las comunicaciones, etc.
- **Librerías/Middleware**. Son una serie de librerías que permiten controlar funciones básicas del sistema como librerías multimedia, librerías gráficas, servicios de seguridad, servicios de base de datos como SQLite, librerías comunes, intérprete de páginas web, etc.
- **Interfaz de aplicaciones**. Este interfaz de aplicaciones va a permitir a las *apps* creadas por terceros y a las aplicaciones nativas controlar notificaciones, servicios de localización, servicios de telefonía, gestión de *apps*, etc.
- **Interfaz de usuario y aplicaciones nativas**. Todos los sistemas operativos ofrecen una serie de aplicaciones nativas como pueden ser los contactos, teléfono, mensajes, ajustes, reloj, etc. Además ofrecen una serie de API o librerías para que los desarrolladores de aplicaciones puedan crear y manejar sus interfaces de manera ágil y sencilla. Con estas librerías los desarrolladores pueden crear en sus aplicaciones ventanas, botones, campos de texto, pestañas, etiquetas, etc.

Figura 2.6. *Interfaz del sistema operativo iOS*

Actualmente el sistema operativo para dispositivos móviles más utilizado es Android, seguido a distancia por iOS, de Apple. En el año 2012 la cuota de mercado de Android era de un 75% mientras que en 2011 fue del 57,5%. Eso quiere decir que se consolida como primer sistema operativo móvil del mercado con diferencia.

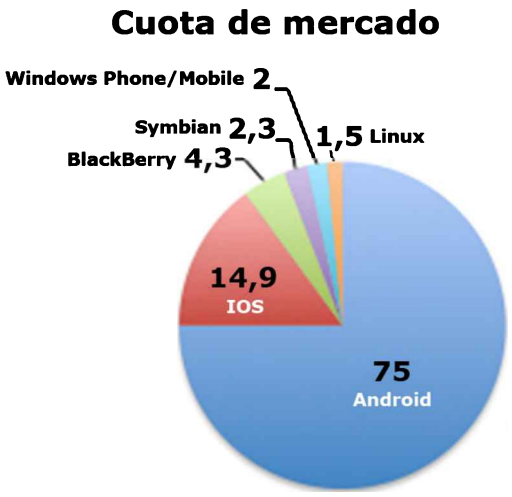

Figura 2.7. *Cuota de mercado de los sistemas operativos móviles en 2012. Fuente IDC*

2.2 PRINCIPALES FUNCIONES DE LOS SISTEMAS OPERATIVOS

A continuación vamos a enumerar las principales funciones de los sistemas operativos:

- Políticas de **reparto de tiempo de proceso**. Controla la ejecución de varios procesos a la vez, repartiendo los recursos del ordenador (procesador, memoria, espacio de almacenamiento…) a los distintos programas que se están ejecutando.
- **Control y manejo de los dispositivos de E/S**. Controla y organiza los dispositivos conectados al sistema.
- **Control de recursos**. Coordinar y manipular el hardware del sistema informático. Se encarga del funcionamiento coordinado de todos los componentes para que funcionen como una sola máquina.
- Administra y mantiene los **sistemas de archivo** de disco permitiendo guardar la información en las unidades de almacenamiento en forma de ficheros y directorios.
- Gestiona y mantiene **usuarios y grupos** de usuarios pudiendo establecer privilegios y restricciones a las diferentes cuentas individuales o grupales.
- Ofrece una **base estándar** sobre la que ejecutar otros programas permitiendo diseñar software de aplicación sin necesidad de tener en cuenta el hardware particular de cada sistema.
- Permite la interacción entre el sistema y los usuarios permitiendo su manejo de forma fácil e intuitiva a través de la **interfaz gráfica** o GUI.
- Detecta e informa de los **errores** que se produzcan.

Un buen sistema operativo aprovechará toda la potencia que ofrece el hardware tratando de que funcione de forma óptima.

Para poder ejecutar cualquier otra aplicación es necesario tener cargado el sistema operativo con el que es compatible. Para ello, el sistema operativo es el primer programa o software que se carga y ejecuta al arrancar o reiniciar el sistema siendo automática su ejecución.

A continuación se pasará a describir en profundidad las funciones más importantes de un sistema operativo:

2.2.1 MANEJO DE LA MEMORIA: MEMORIA VIRTUAL Y PAGINACIÓN

Como ya se sabe, los programas para ser ejecutados tienen que estar en memoria principal. No es posible ejecutar ningún programa desde el disco duro, unidad SSD u otra memoria secundaria. Cuando un programa se traslada a memoria principal para ejecutarse pasa a llamarse proceso. Los procesos cuando se están ejecutando tienen ubicados en memoria tanto el código que se está ejecutando como los datos.

Si siempre fuera mayor la memoria existente que la que se puede necesitar para ejecutar todos los procesos no habría ningún tipo de problema. El caso es que en muchas ocasiones se necesita más memoria de la que verdaderamente existe y la solución a este problema podría ser el no admitir a más procesos de los que se podrían ejecutar en un momento determinado, echar a los procesos de la memoria si esta escasea, dejar de asignar memoria a los procesos que la demandan con lo cual se pueden quedar colgados, que los procesos pidan una memoria determinada y se comprometan a no pedir más, etc. Todas estas soluciones no parecen sensatas, por lo tanto, lo que se ha acordado hacer es utilizar memoria virtual.

La memoria virtual es una técnica que permite hacer ver a un sistema que dispone de más memoria de la que verdaderamente tiene. Los sistemas lo que hacen es utilizar parte de la memoria secundaria (el disco duro) para compensar la falta de memoria RAM. El objetivo del sistema es ubicar en memoria los procesos o parte de los procesos que se están ejecutando en ese mismo momento y llevar a memoria secundaria aquellas partes de procesos que sean menos necesarias.

Toda esta tarea hay que realizarla de forma transparente a los procesos. Los procesos lo único que ven es que hay mucha memoria disponible y es el hardware y el sistema operativo el que internamente gestiona el paso de zonas de memoria RAM a memoria secundaria y viceversa.

Por ejemplo, en un sistema de 32 bits, el máximo de memoria que se podría utilizar es 2^{32} que es equivalente a 4 Gigabytes, si el sistema es de 64 bits el máximo sería mucho mayor, 18,4 Zettabytes.

2.2.1.1 ¿Cómo funciona la memoria virtual?

El sistema operativo y el hardware llevan una correspondencia entre la memoria virtual y la memoria RAM y secundaria. Luego hacen la traducción de direcciones virtuales a direcciones reales. La traducción de estas direcciones virtuales a direcciones reales la hace la Unidad de Manejo de Memoria o **MMU** (*Memory Management Unit*).

Cuando un proceso necesita acceder a una dirección de memoria virtual pueden pasar dos cosas:

- Que al hacer la traducción, la dirección se encuentre en la memoria física. Con lo cual el proceso puede seguir ejecutándose sin problema.

- Que al hacer la traducción, la dirección se encuentre en la memoria secundaria lo que provocaría una **excepción**. En ese caso si se quiere que el programa se siga ejecutando hay que traer a memoria principal el contenido de la dirección de memoria secundaria.

La ventaja de este sistema es el poder ejecutar más procesos de los que se podrían ejecutar sin memoria virtual. Los programas se dividen en fragmentos y se depositan en memoria secundaria aquellos fragmentos que no se utilizan, por lo tanto también se está ahorrando tiempo ya que no se carga en memoria lo que no se necesita. Es el sistema operativo y el hardware los que se encargan de mover estos fragmentos entre memoria secundaria y memoria principal. Concretamente es el sistema operativo el que se encarga de decidir qué partes de la memoria del proceso se mantienen en memoria física y se encarga de gestionar las tablas de traducción de direcciones virtuales a físicas para que la MMU haga su trabajo. También es tarea del sistema operativo localizar una zona de memoria donde alojar el fragmento de proceso cuando se produce una excepción.

A la MMU se le puede indicar de los fragmentos qué propiedades tienen (lectura, escritura y o ejecución) y de esa manera por ejemplo el sistema operativo puede proteger tanto su código como las tablas de traducción de memoria virtual. Este **mecanismo de protección** impide corromper el sistema operativo y proteger también los procesos entre sí.

Otra posibilidad que ofrece la memoria virtual es la **reubicación**. Con este sistema es posible mover los procesos de zona de memoria y seguir ejecutándolos sin problema.

El objetivo de este sistema de memoria virtual es tener en memoria principal aquellos fragmentos que más se utilicen y dejar en memoria secundaria los menos utilizados. De esta forma el sistema es eficiente y efectivo.

2.2.1.2 La paginación

En la paginación, los programas se trocean en partes del mismo tamaño que se llaman páginas. La memoria se divide en trozos del mismo tamaño y en vez de páginas se llaman **marcos**.

Las páginas, al tener el mismo tamaño se pueden cargar en cualquier marco que quede libre (los marcos puede que no estén contiguos).

Las páginas son los bloques unitarios que se van transfiriendo entre memoria secundaria y memoria RAM.

Al no tener que estar necesariamente las páginas contiguas en memoria, el sistema deberá tener un mecanismo para convertir las direcciones de páginas dadas por los programas por direcciones reales de memoria.

El sistema operativo necesita tener en memoria una tabla para cada uno de los procesos que se llama Mapas de página. También deberá tener una lista de marcos libres. En el caso de que se quiera ejecutar un programa con X páginas de tamaño, bastará con encontrar X marcos libres y cargar el programa en ellos.

Este sistema tiene sus inconvenientes puesto que ya se puede deducir que este mecanismo consume muchos recursos de memoria y CPU. No obstante, también tiene muchas ventajas. Por ejemplo, al no tener que estar contiguos los marcos de páginas utilizados por un proceso no hace falta compactar la memoria. Además se puede ejecutar un programa sin que esté cargado completamente en memoria, se puede cargar parte al comienzo y luego se puede ir cargando más programa conforme se vaya necesitando.

2.2.2 POLÍTICAS DE REPARTO DE TIEMPO DE PROCESO

Los procesos tienen un ciclo de vida muy parecido. Una vez creados, se alternan en el uso de la CPU, E/S y otros recursos hasta que terminan voluntariamente, por algún error o por otro motivo y es entonces cuando devuelven los recursos asignados (memoria, archivos, tablas en *kernel*, etc.). Existen procesos en el sistema que nunca terminan como pueden ser los demonios o *daemons* (servidores web, servidores ssh, etc.).

Por lo tanto, dado que muchos procesos van a coexistir, el objetivo de toda política de reparto de tiempos de proceso es el conseguir el máximo aprovechamiento de los recursos del sistema. Para ello, deberá basarse en aquellos algoritmos de planificación que mantengan la CPU activa de forma efectiva el mayor tiempo posible.

Los procesos dentro de su ciclo de vida pueden estar en tres estados:

- **En ejecución**. Es cuando el procesador está ejecutando las instrucciones que componen el programa. La CPU en ese mismo instante le ha concedido el tiempo de uso al proceso.
- **Activo** (también llamado preparado o en espera). Es cuando un proceso está a la espera de ser ejecutado, es decir, está esperando turno para poder utilizar su intervalo de tiempo y poner a ejecutar las instrucciones del programa accediendo a los recursos del sistema.
- **Bloqueado**. Es cuando el proceso está retenido debido a múltiples causas. Si varios procesos intentan acceder al mismo fichero o a la misma unidad de DVD, uno de ellos deberá permanecer bloqueado hasta que el recurso quede disponible.

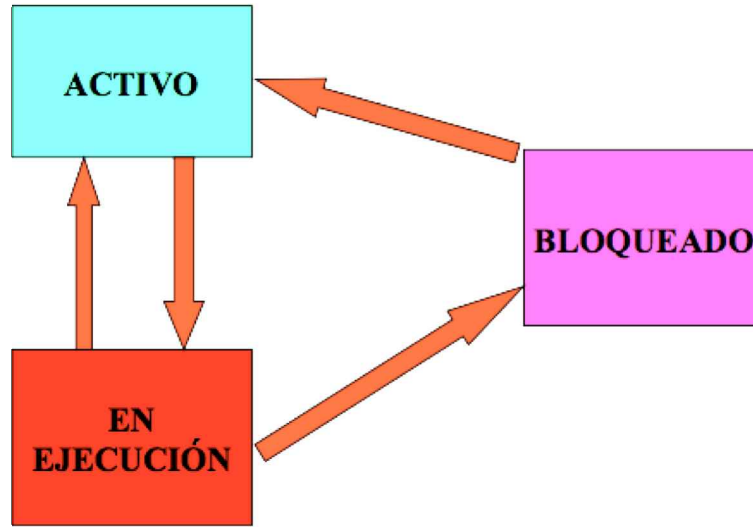

Figura 2.8. *Estados de un proceso y transiciones entre ellos*

Procesos e hilos
Cuando se necesita que un programa haga varias cosas a la vez, existen dos formas diferentes de implementarlo, mediante procesos y mediante hilos. Los procesos son totalmente independientes unos de otros mientras que un hilo no. Un proceso puede tener varios hilos en ejecución y cada hilo realizará una tarea diferente o igual según como haya sido programado. La principal diferencia entre un proceso y un hilo es que los procesos tienen su propia zona de memoria mientras que los hilos de un proceso comparten la misma zona de memoria.
Por temas de seguridad, los procesos pueden ser mejores puesto que si un programa se cuelga, se cuelga solo él, mientras que si un programa tiene varios hilos en ejecución y se cuelga algún hilo, todos los demás hilos se verán afectados. Por el contrario, en cuanto al rendimiento es mejor utilizar hilos puesto que es más ágil para el sistema gestionar hilos que procesos al no tener que asignar recursos diferentes a cada hilo. En resumen, un proceso es más costoso de lanzar mientras que un hilo es más ligero. Por lo tanto, se elegirán hilos cuando se necesite que las tareas que realice el sistema compartan sus datos o cuando se van a generar muchísimas tareas a la vez.
Otro aspecto es la complejidad, los hilos al compartir memoria tienen que programarse de forma más cuidadosa puesto que un hilo puede machacar el trabajo de otro hilo o hacer que se cuelgue. Siempre que los hilos compartan datos hay que tener este punto en cuenta.

En cuanto a la planificación, hay que tener muy en cuenta el concepto de **cambio de contexto**. Un cambio de contexto es un paso de un estado a otro, para ello la CPU tiene que guardar su estado (el valor de los registros de la CPU que tenía cuando salió) y por lo tanto cada cambio de contexto supone el gasto de algunos ciclos de CPU que hace que baje la eficiencia de la misma. El objetivo de toda política de reparto de tiempo de CPU deberá intentar satisfacer a todos los procesos en la medida de lo posible, minimizar los cambios de contexto y mantener la CPU en ejecución el mayor tiempo posible.

Las políticas de reparto se basan en **algoritmos de planificación** que son los que deciden qué procesos tienen que ejecutarse en cada momento y por qué. Estos algoritmos deberán basarse en la equidad, eficiencia, imparcialidad, tiempo de respuesta y rendimiento del sistema.

Algunos algoritmos de planificación son los siguientes:

- **FIFO** o **FCFS** (*First Come First Served*). Primero en entrar, primero en salir. El planificador asignará los ciclos de CPU a cada proceso en función de una cola FIFO en la que el primer proceso que llega al sistema será el primero en ser atendido. Al primer proceso que llega se le asignarán tantos ciclos de CPU como necesite hasta que termine completamente. Una vez terminado con este proceso, se procederá a ejecutar el siguiente y así sucesivamente hasta terminar con el último proceso.
- **Round Robin**. En este algoritmo se asignan tiempos de ejecución de forma rotativa entre los procesos. Este algoritmo se caracteriza por la equidad, dado que se asigna a todos los procesos el mismo quantum o ciclos de CPU. La selección de los procesos se hace mediante una cola FCFS o FIFO.

- **SJF** (*Shortest Job First*) o primero el más corto. En la búsqueda de aumentar la productividad del sistema, el planificador otorgará el procesador al proceso que demande menos ciclos de CPU. En el caso de que existan varios procesos que demanden los mismos ciclos de CPU, se les atenderá por orden de llegada (FCFS).
- **Planificación por prioridades**. Cada proceso tendrá una prioridad determinada que servirá como criterio al planificador para otorgarle el uso de la CPU. En el caso de que varios procesos tengan la misma prioridad se podrá combinar con otro algoritmo como FCFS para elegir el proceso a ejecutar.
- **Planificación con múltiples colas**. Los algoritmos anteriores tenían en cuenta solamente una cola. Si se utilizan múltiples colas se puede utilizar un criterio para asignar los procesos a una cola determinada y otro para elegir qué proceso de qué cola deberá ser ejecutado.

2.2.2.1 Comandos de control de procesos en sistemas UNIX

Todos los comandos que se van a ver a continuación se ejecutan desde la línea de comando.

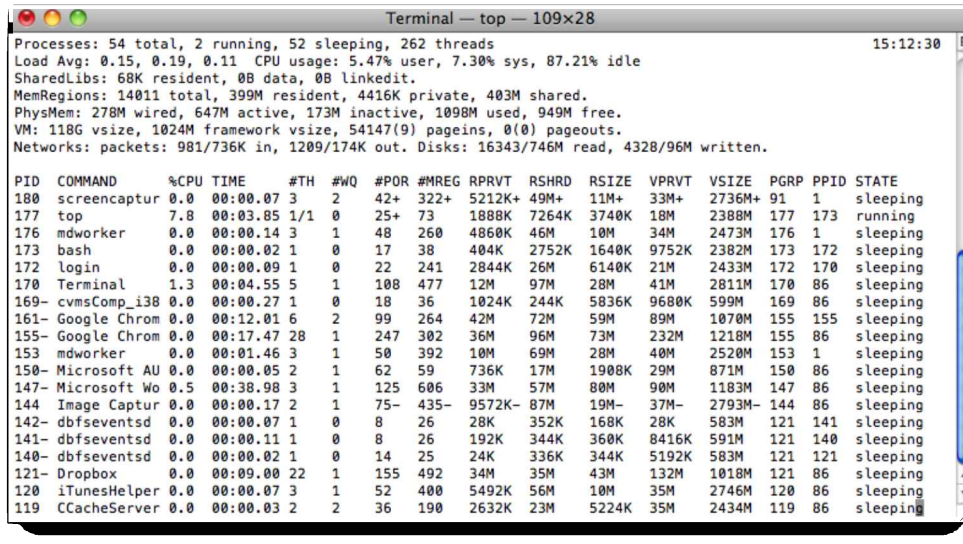

Figura 2.9. *Comando top en Mac OS X*

RECUERDA

El PID es un número que identifica el proceso dentro del sistema. Todos los procesos tienen un PID diferente.

- **top** [opciones]

  El comando top proporciona una visión rápida de los procesos que se están ejecutando en ese mismo instante en el sistema.

- **ps** [opciones] [PID]

  Este comando es muy potente, puesto que permite muchas opciones como parámetro. Si no se especifica ninguna opción, muestra una tabla con todos los programas que el usuario está ejecutando.

- **aux**

  Muestra una lista detallada de todos los procesos independientemente del propietario.

- **kill** [opciones] PID

  Kill envía una señal TERM que indica al programa que tiene que terminar. De esta forma se puede forzar la terminación a procesos que no están actuando de forma correcta. En ocasiones, si tras enviar la señal TERM el proceso no termina, se puede utilizar el parámetro -9 para forzar la salida con la señal KILL, lo que hace que termine el proceso de todas formas.

- **killall** [opciones] nombredeproceso

  Este comando es parecido al comando kill pero utiliza el nombre del proceso en vez del PID como argumento. Eliminará del sistema a todos los procesos que tengan ese nombre.

2.2.3 ENTRADA/SALIDA: MANEJADORES DE INTERRUPCIONES Y DISPOSITIVOS

El sistema de entrada/salida es aquella parte del sistema operativo encargada de abstraerse los dispositivos de entrada/salida conectados al equipo y ofrecer una visión más simplificada de los mismos de tal forma que los demás componentes del SSOO o software puedan hacer uso de ellos de una manera fácil.

Los dispositivos de entrada/salida se pueden clasificar en tres grupos:

- **Dispositivos de almacenamiento**. Como su nombre indica, van a servir al sistema para almacenar la información del mismo de forma no volátil. Son dispositivos de este tipo las unidades SSD, los discos, los *pendrives*, las tarjetas de memoria, etc.

- **Dispositivos de comunicaciones**. Permiten conectar el sistema con otros equipos a través de una red. La tarjeta de red es un dispositivo de este tipo.

- **Dispositivos de interfaz de usuario**. Permiten la interacción entre el usuario y el equipo. Ratones, teclado, pantalla, impresoras, etc., son dispositivos de este tipo.

Como ya se ha explicado, el sistema de entrada/salida deberá ofrecer un interfaz sencillo y de fácil utilización al resto de sistema operativo y demás software. Además, al igual que se hace con el USB, deberá proporcionar dispositivos virtuales de tal manera que no haya que instalar *drivers* si no es necesario. Mediante mecanismos de *plug and play* el sistema permitirá la conexión y uso de dispositivos de E/S de una forma ágil. Además de todas estas funciones, el sistema de E/S deberá velar por la eficiencia del sistema.

El sistema de entrada/salida es el que se encarga de tratar con los controladores de dispositivo o *drivers*. Como ya se ha visto, los *drivers* transforman las órdenes genéricas que le envía el sistema operativo en otras más entendibles por el dispositivo en cuestión.

2.2.4 BLOQUEO DE RECURSOS

Una de las funciones principales de un sistema operativo es la de gestionar de forma adecuada los recursos del sistema. Los procesos van demandando recursos y es función del sistema operativo gestionar la asignación de los mismos. Generalmente los procesos solicitan el recurso, una vez que el sistema operativo se lo asigna, lo utilizan y cuando ya no lo necesitan lo liberan para que cualquier otro proceso pueda utilizarlo.

Si cuando el proceso solicita el recurso, este no está disponible, se bloquea y espera a que quede disponible. En ocasiones, el proceso tiene que reintentar la solicitud él mismo o puede ser que se despierte automáticamente cuando el recurso quede disponible.

Una vez que un recurso es asignado a un proceso, el sistema operativo puede quitárselo sin que pase nada, como puede pasar con la memoria y con el uso del procesador y en otras ocasiones el quitarle el recurso puede hacer que el proceso se cuelgue. Imagínese que un proceso está quemando un DVD y se le retira el recurso. Seguramente el proceso se cuelgue. Puede pasar lo mismo con un proceso que está imprimiendo un documento. Si se le quita el recurso de la impresora seguramente se quede colgado.

- Se llaman **recursos apropiativos** a aquellos que pueden ser retirados sin causar ningún daño.
- Se llaman **recursos no apropiativos** a aquellos que al ser retirados el proceso propietario falla.

El problema con los bloqueos es cuando los procesos llegan a un estado de interbloqueo. Esto quiere decir que un proceso quede bloqueado en la espera de un evento que nunca va a ocurrir. Un ejemplo de interbloqueo, *deadlock* o abrazo mortal es el siguiente:

- El proceso P1 solicita un recurso R1 y el sistema se lo concede.
- El proceso P2 solicita un recurso R2 y el sistema también se lo concede.
- El proceso P1 ahora solicita el recurso R2. El proceso se queda a la espera de que el recurso R2 quede libre.
- El proceso P2 solicita el recurso R1.

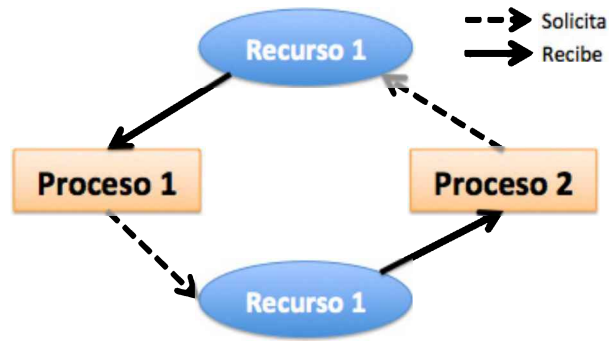

Figura 2.10. *Situación de interbloqueo*

Ambos procesos quedan bloqueados a la espera de que el otro proceso libere el recurso. Como esta circunstancia nunca va a ocurrir ambos procesos quedarán bloqueados para siempre a no ser que el sistema operativo intervenga.

Los sistemas operativos están diseñados para evitar estos interbloqueos implementando técnicas que prevengan los bloqueos, los eviten, los detecten o se recuperen de los mismos.

Los semáforos
Un semáforo se utiliza en sistemas operativos multiproceso para restringir el acceso a un recurso. Los semáforos según sean inicializados permitirán que más o menos procesos utilicen un recurso de forma simultánea.

2.2.5 SISTEMAS DE ARCHIVO

El sistema de archivos es un componente fundamental dentro del sistema operativo. A continuación se pasará a estudiar en profundidad algunas características de los sistemas de archivos.

2.2.5.1 Implementación de sistemas de archivos

El espacio de un disco está dividido en sectores. Físicamente, el sector es la unidad de transferencia que el disco se encarga de leer o escribir. En la mayoría de sistemas, un sector ocupa 512 *bytes* y como un disco es un dispositivo de acceso aleatorio, un sector puede accederse directamente. En el caso de unidades SSD el sistema es igual salvo por la diferencia de que el dispositivo es totalmente electrónico mientras que los discos tienen parte electrónica y parte mecánica.

Los sistemas de ficheros para ser más eficientes leen o escriben en los discos bloques que son un múltiplo de sectores. Un bloque es un grupo de sectores y el tamaño del mismo se establecerá de tal manera que el sistema sea lo más rápido y eficiente. Si se elige un bloque muy grande, el sistema puede ser más rápido al leer o escribir un fichero en disco pero por el contrario puede desaprovechar espacio al almacenar bloques en discos medio vacíos.

La implementación del sistema de ficheros suele realizar varias funciones. Estas funciones se pueden ver como distintas capas de abstracción:

- **Capa lógica**. Mantiene los metadatos del sistema. Los metadatos son estructuras que almacenan todo el sistema de directorios almacenados en el disco y otras estructuras con todos los ficheros que contienen los bloques que ocupan cada fichero en el disco (estas últimas estructuras se llaman inodos). Esta capa lógica es totalmente independiente de cómo sea internamente el disco (disco duro, unidad SSD, etc.).
- **Capa de control de bloques de ficheros**. Esta capa se encarga de traducir los bloques lógicos en bloques físicos (un bloque físico es una agrupación de sectores).
- **Capa de comandos básicos**. Esta capa tiene una serie de rutinas básicas que permitirán a las capas superiores realizar lecturas y escrituras al dispositivo físico.
- **Capa de *driver***. Esta capa es totalmente dependiente del dispositivo. Controla la entrada/salida física. Se encarga de las transferencias de bloques desde el disco y hacia el disco.

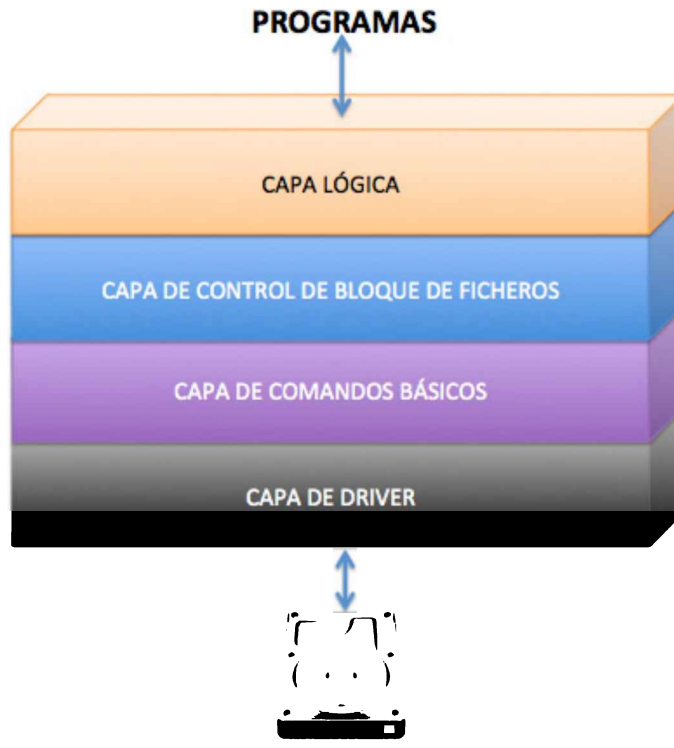

Figura 2.11. *Capas de un sistema de ficheros*

2.2.5.2 Los archivos y directorios en UNIX

En sistemas Unix como Linux, un fichero se implementa en una estructura llamada inodo. Un inodo, aunque no contiene el nombre del archivo porque no le hace falta, contiene todas las características del archivo. Cada inodo está identificado en el sistema con un número entero y todos los inodos tendrán un número diferente.

Los inodos según el estándar POSIX contendrán las siguientes propiedades:

- Información. Formada por:
  - Dentificador del dispositivo. Dispositivo donde se almacena el sistema de archivos.
  - Número de inodo.
  - Longitud del archivo (en *bytes*).
  - Identificador del usuario propietario.
  - Identificador del grupo al que pertenece el fichero.
  - Permisos.
  - Marcas de tiempo (fecha de creación, último acceso y última modificación).

– Número de enlaces asociados a este inodo. Por si existen enlaces en algún directorio a este inodo.
  - Enlaces directos a bloques. Tendrán punteros a bloques donde almacenarán los datos del fichero.
  - Enlaces indirectos a bloques. Tendrán punteros a bloques donde almacenarán otros punteros que apunten a bloques con los datos del fichero.
  - Enlaces dobles indirectos a bloques. Ídem a los anteriores salvo que agregan otro nivel de enlaces para poder de esta manera aumentar el tamaño del fichero.

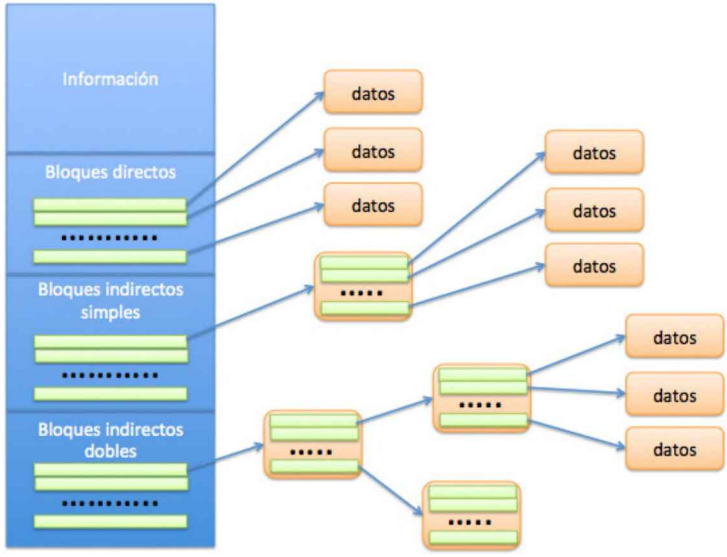

Figura 2.12. *Estructura de un inodo*

Los directorios en Unix son considerados como ficheros también pero con la diferencia de que en vez de acceder a datos directamente, contendrán una estructura en la que almacenarán parejas de nombres de fichero con su número de inodo correspondiente.

2.2.5.3 Comparación de sistemas de archivos

A continuación se verán las características de los sistemas de ficheros más utilizados como son el NTFS, el EXT4 y el HFS+.

■ **NTFS** (*New Technologies File System*) es un sistema de archivos que apareció en 1993 con WIndows NT® y está presente en las últimas versiones del sistema operativo Windows®. Se basó en el sistema HPFS de IBM® y Microsoft® pero también está influido por el sistema de archivos HFS de Apple. Tiene una desventaja y es que ocupa mucho espacio en disco, luego para volúmenes pequeños no es recomendable. Utiliza listas de control de acceso para gestionar el permiso de acceso a los archivos y permite comprimir y cifrar los archivos de forma transparente. Este sistema está en continua evolución y las últimas versiones incorporan puntos de montaje de volúmenes y cuotas de disco. Microsoft®, al contrario que hizo con FAT no ha mostrado cómo funciona internamente este sistema pero gracias a la ingeniería inversa se han desarrollado controladores para que otros sistemas puedan dar soporte de lectura y escritura sobre este tipo de volúmenes.

- **HFS+**. Es el sistema de archivos que utiliza Apple. Su precursor fue el HFS (*Hierarchical File System*). Este sistema se utiliza en los sistemas operativos Mac OS® y también en los iPhone® o iPod®. Al igual que NTFS, utiliza listas de control de acceso para gestionar el permiso de acceso a los archivos. Permite el cifrado transparente de los ficheros salvo en el directorio home. Al contrario que NTFS no permite la compresión transparente.
- **EXT4** (*fouth extended filesystem*). Es el sistema de archivos utilizado en muchas distribuciones Linux. Se desarrolló para mejorar el EXT3 y se caracteriza por demandar un menor uso de CPU y por mejorar la velocidad de lectura y escritura. Al contrario que NTFS no permite la compresión ni el cifrado transparente. Utiliza el estándar POSIX para gestionar el permiso de acceso a los archivos. La ventaja de este sistema es que lo hace retrocompatible con sistemas EXT3. Un sistema EXT3 se puede montar como una partición EXT4 y viceversa.

Seguridad en los sistemas de archivos
Los sistemas de ficheros suelen confiar su seguridad a mecanismos con ACL o listas de control de acceso. Una ACL suele consistir en una tabla que especifica los derechos que tienen usuarios o grupos de usuarios sobre ficheros, programas o procesos. Todo objeto que pueda ser accedido tendrá su entrada en la ACL en la que se especifique los usuarios o grupos de usuarios que pueden acceder a él y qué privilegios tendrán sobre el mismo (lectura, escritura o ejecución).

2.2.5.4 Sistemas de archivos con *journaling*

Una de las principales tareas de un sistema de archivos es asegurarse de que la información que se lee del dispositivo es la misma que se guardó en él. Antiguamente, los sistemas de archivos como EXT2 de Linux carecían de *journaling* mientras que actualmente además de *journaling*, los sistemas de ficheros pueden gestionar particiones más grandes, reducen la fragmentación, tienen mejor rendimiento de entrada/salida y se recuperan mejor ante posibles accidentes.

El problema de sistemas de ficheros antiguos era que una salida inesperada del sistema ya fuera por un corte de luz, fallo software o intencionada obligaba a realizar un chequeo en profundidad del sistema de archivos una vez se iniciaba la máquina la siguiente vez (el famoso fsck de Unix). Dependiendo de lo grande que fueran las particiones, del número de archivos o directorios que contenían, de la velocidad del hardware, etc., este proceso se podía demorar bastante tiempo.

El problema de los sistemas sin *journaling* es que todas las operaciones de escritura en disco se realizan en varios pasos. Si entre medias de alguno de esos pasos el sistema de ficheros se reinicia o para por la razón que sea, una vez reiniciado, deberá examinar todos los metadatos para detectar y reparar todos los problemas de integridad existentes (con fsck). Dado que el tiempo que tarda fsck es proporcional al tamaño de la partición y al número de ficheros y directorios, para los sistemas actuales es imprescindible utilizar un sistema de *journaling*.

Los sistemas de ficheros con *journaling* trabajan con transacciones. Las transacciones son un conjunto de operaciones que se realizan de forma unitaria (o se completan del todo o no se realiza nada). Es algo parecido a una transferencia entre cuentas. Si, por ejemplo, quiero mover dinero entre dos de mis cuentas del banco, el sistema retira

una cantidad de dinero de una y la abona en la cuenta de destino. Si una vez de retirado el dinero de la primera cuenta se produce un fallo eléctrico, la transacción no se habrá completado y el sistema restablecerá el saldo de las cuentas a los valores existentes antes de la transacción. Algo parecido hacen los sistemas con *journaling*.

Cuando un sistema de ficheros con *journaling* se inicia ya no tiene que revisar y verificar los metadatos del mismo, sino que tiene que revisar solamente aquellas transacciones que no se llevaron a cabo y las ejecuta. De esa manera se asegura que la información del disco y los metadatos están sincronizados y son veraces.

El *journal* en un sistema de ficheros consiste básicamente en una lista de transacciones, de esta manera las recuperaciones ante posibles accidentes son mucho más ágiles que en sistemas sin *journaling*.

Existen muchos sistemas de ficheros con *journaling* como por ejemplo EXT4, NTFS, ReiserFS, JFS, etc.

2.2.6 MULTIPROCESO, MULTITAREA Y MULTIUSUARIO

Multiproceso, multitarea y multiusuario son características de los sistemas operativos. A continuación se pasará a estudiar estos conceptos en profundidad:

Multiproceso. Un sistema operativo multiproceso es aquel que puede ejecutar varios procesos de forma concurrente (a la vez). Para que se puedan ejecutar varios procesos a la vez es necesario tener un sistema multiprocesador (con varios procesadores).

El objetivo de utilizar un sistema con varios procesadores no es ni más ni menos que aumentar la potencia de cálculo y por lo tanto rendimiento del mismo. El sistema va repartiendo la carga de trabajo entre los procesadores existentes y también tendrá que gestionar la memoria para poder repartirla entre dichos procesadores. Generalmente los sistemas multiprocesador se utilizan en *workstations*, servidores, etc.

El realizar un sistema operativo que pueda trabajar con varios procesadores de forma concurrente es una tarea complicada y generalmente se diseñan para que trabajen de varias formas:

- **Forma asimétrica**. Se designa un procesador como el procesador *master* (maestro) y los demás serán *slave* (esclavos) de este. En el procesador maestro se ejecuta el sistema operativo y se encargará de repartir el trabajo entre los demás procesadores esclavos. Este sistema tiene la ventaja de que es más simple de implementar ya que se centraliza la gestión en un procesador. Por el contrario, el que existan procesadores que realicen distinto trabajo hace que el sistema no sea tan eficiente como un sistema operativo que trabaje de forma simétrica.
- **Forma simétrica**. En este tipo de sistema operativo todos los procesadores realizan las mismas funciones. Es más difícil de implementar pero son más eficientes que los sistemas operativos multiproceso asimétricos. El poder balancear la carga de trabajo entre todos los procesadores existentes hace que el tiempo de inactividad de los mismos sea mucho menor y por lo tanto la productividad mucho más alta. Otra ventaja es que si un procesador falla, gracias al balanceo de carga, los demás procesadores se hacen cargo de las tareas del procesador que ha fallado.
- **Multitarea**. En un sistema operativo multitarea, varios procesos se pueden estar ejecutando aparentemente al mismo tiempo sin que el usuario lo perciba. La gran mayoría de sistemas operativos actuales son multitarea. Un sistema multitarea permite a la vez estar escuchando música, navegando por Internet y realizando una videoconferencia. El sistema operativo fracciona el tiempo de CPU y lo va repartiendo entre los procesos que lo necesitan de la mejor forma posible. Además de la multitarea aparece el concepto de multihilo, que veremos a continuación.

- **Multihilo**. En ocasiones, un proceso puede tener varios hilos de ejecución de tal manera que un proceso pueda ejecutar varias tareas a la vez. El multihilo se utiliza a veces por eficiencia, debido a que el crear muchos procesos implica la asignación de muchos recursos mientras que muchos hilos pueden compartir los recursos y memoria de un proceso. El multihilo evita además la pérdida de tiempo en cambios de contexto al ejecutarse todos los hilos en el mismo contexto.
- **Multiusuario**. Un sistema operativo multiusuario es un sistema que puede dar servicio a varios usuarios de forma simultánea. Hay que tener en cuenta que un sistema con varias cuentas de usuario no es necesariamente un sistema multiusuario. Por ejemplo, las versiones domésticas de los sistemas Windows® permiten tener varias cuentas de usuario en el mismo sistema operativo pero no permiten que haya varios usuarios trabajando en el sistema al mismo tiempo. Las versiones Server de Windows® sí permiten esta característica a través de *terminal server*. Por el contrario, las versiones Linux sí son multiusuario desde hace mucho tiempo.

2.2.7 ORGANIZACIÓN DE USUARIOS

En esta sección nos vamos a centrar en el sistema de usuarios y permisos de Linux. Las explicaciones sirven prácticamente para cualquier distribución Linux, pero los ejemplos se van a hacer sobre Ubuntu Linux.

En Linux existen tres tipos de usuarios.

- ***Root* o superusuario**. Es el usuario encargado de administrar el sistema, pudiendo realizar cualquier tipo de tarea sin restricción alguna. Es igual al usuario administrador de Windows®.
- **Usuarios normales**. Generalmente son personas físicas teniendo algunos usuarios más privilegios que otros.
- **Usuarios de sistema**. Son usuarios utilizados por procesos o servicios los cuales necesitan una cuenta de usuario para realizar sus funciones. Al listar el fichero */etc/passwd* se pueden ver algunos de ellos (FTP, mail, etc.).

RECUERDA

Los usuarios en un sistema Linux se almacenan en el fichero */etc/passwd*.

2.2.7.1 ¿Cómo funcionan los grupos de usuarios?

Imagínese que se tiene el siguiente problema: existen dos grupos de personas en una empresa, el grupo de comerciales los cuales trabajan con una serie de listados de precios, ventas y clientes y el grupo de técnicos, los cuales trabajan con documentación técnica de la empresa en la que trabajan. Se necesita que las únicas personas que accedan a los listados de precios, ventas y clientes sean los comerciales. Ellos necesitan trabajar con estos documentos pero no deben acceder a la documentación técnica. Sin embargo, el grupo de los técnicos se desea que acceda a la documentación técnica pero no a los listados de precios, clientes y ventas.

Una solución a este problema es crear en el sistema una carpeta por usuario y darle el permiso correspondiente a la carpeta de cada usuario. Dado que son 30 empleados no parece muy operativo, es más, hay archivos como la lista de ventas que tiene que poder actualizarse por cualquier comercial. En ese caso cualquier actualización de un comercial tiene que replicarse en los demás 29 archivos de sus compañeros, lo cual lo hace inviable.

Existe otra solución y es crear un grupo de comerciales, el cual tendrá acceso a la carpeta comerciales y el grupo técnicos el cual podrá acceder a la carpeta *docu_técnica*. Cada usuario estará en el grupo correspondiente y podrá existir un jefe que pertenezca a los dos grupos lo cual permite acceder a toda la información.

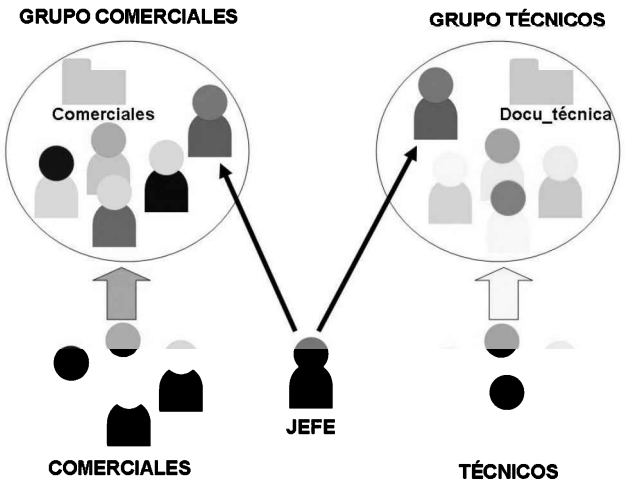

Figura 2.13. *Ejemplo de grupos de usuarios*

2.2.7.2 Gestionar usuarios y grupos en Linux

Gestionar grupos y usuarios en Linux se puede hacer de las siguientes maneras:

- **Mediante comandos**. Lo cual permitirá dar de alta, baja y modificar usuarios en cualquier sistema Linux con los mismos comandos.
- **Editando los ficheros *letc/passwd* o *letc/group***. Este método puede dar algún problema si no se edita el fichero correctamente.
- **Con una herramienta administrativa**. Es más cómodo, pero las herramientas cambian con el tiempo y pueden diferir de una distribución a otra, mientras que no ocurre esto con los comandos.

Consejo sobre contraseñas

- Nunca hay que utilizar contraseñas que sean fáciles de adivinar (sobre todo la del *root*). Generalmente se utilizan contraseñas con los nombres de los hijos, de la novia, con la fecha de nacimiento, etc. Estas contraseñas son fáciles de adivinar. Las contraseñas hay que memorizarlas y no escribirlas en archivos, cuadernos, etc.
- Es buena costumbre cambiar las contraseñas de vez en cuando.
- Mejor elegir contraseñas con mayúsculas, minúsculas y números y con al menos 8 caracteres (si se utiliza algún símbolo de puntuación mejor).

En la siguiente figura se puede observar la herramienta de gestión de usuarios de Ubuntu. Como se puede observar, el interfaz es sencillo e intuitivo y todas las posibilidades que se pueden ejecutar mediante comandos se podrán realizar con el interfaz.

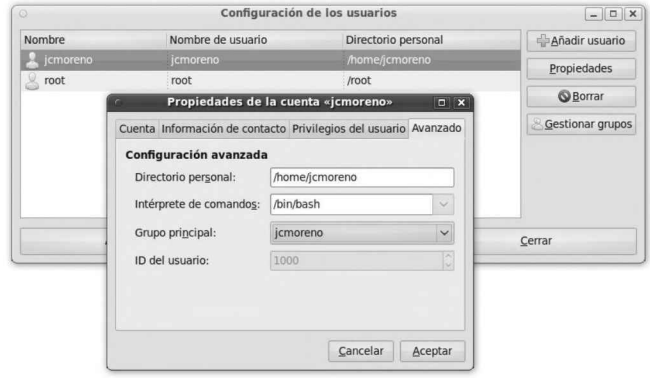

Figura 2.14. *Herramienta de administración de usuarios de Ubuntu*

La gestión de grupos y asignación de usuarios a grupos se puede realizar desde la misma herramienta:

Figura 2.15. *Herramienta de administración de grupos de usuarios de Ubuntu*

2.2.7.3 Los permisos en Linux

Linux, como se vio en el apartado anterior tiene un sistema de seguridad basado en usuarios y grupos. Los ficheros y directorios en Linux tienen seguridad a tres niveles. Se establecerán permisos a los ficheros o directorios al usuario creador del mismo, al grupo al que pertenece el fichero o directorio y a los demás usuarios del sistema.

En Linux existen tres tipos de permisos para archivos y directorios:

- **r. Lectura**. Se puede acceder al contenido del archivo o listar el contenido del directorio.
- **w. Escritura**. Permite borrar y modificar un archivo y en un directorio crear y borrar ficheros dentro de él.
- **x. Ejecución**. A diferencia de Windows®, en Linux los ficheros no hace falta que sean .exe para poder ejecutarse. Los directorios con permiso de ejecución permiten realizar operaciones sobre ellos mediante los otros permisos de lectura y escritura.

Cuando se lista una serie de archivos y directorios (con el comando ls –l) se puede observar algo parecido a la siguiente figura:

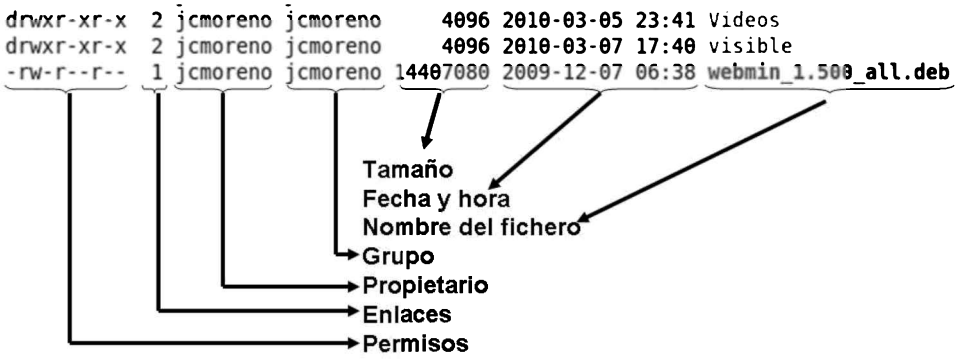

Figura 2.16. *Listado de una serie de archivos*

Como se puede observar en la columna de la izquierda de la figura aparecen los permisos de dichos archivos o directorios. A continuación se comentarán estos permisos:

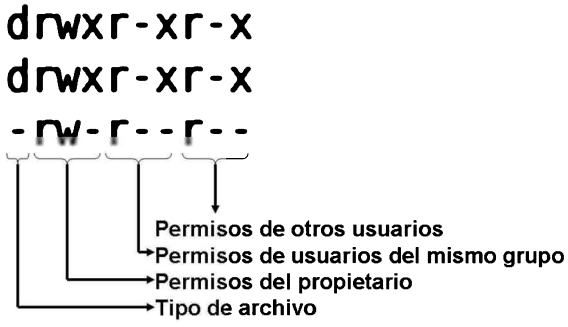

Figura 2.17. *Detalle de los permisos*

El primer dígito de los permisos muestra entre otros si es un fichero regular (-) o un directorio (d). Los permisos (r,w ó x) aparecen en tres grupos y cada permiso tiene una posición (si aparece un guión "–" es ausencia de dicho permiso), el primero son los permisos del usuario seguido de los permisos de los usuarios que pertenezcan al grupo al que pertenece dicho archivo o directorio y por último los permisos que tienen los demás usuarios del sistema sobre dicho archivo o directorio.

2.2.7.4 La cuenta de administrador o superusuario

El superusuario o administrador tiene una serie de privilegios específicos como pueden ser:

- Cambiar propiedad de ficheros.
- Modificar límites de utilización de recursos y prioridades.
- Gestionar los sistemas de ficheros.
- Poner nombre a las máquinas.
- Configuración de la red.
- Configuración de las impresoras.
- Establecimiento de parámetros del sistema.
- Gestionar cuentas de usuario, deshabitar cuentas, etc.
- Otras operaciones.

2.2.7.5 ¿Cuándo deben deshabilitarse las cuentas?

Cuando se cumpla alguna circunstancia de las siguientes:

- Cuando la cuenta no se utiliza durante un tiempo determinado.
- Cuando caduca la misma.
- Cuando el administrador la da de baja.
- Cuando un usuario trata de entrar a la misma e introduce x veces mal la contraseña. Esa x o número de veces podrá ser configurada en el sistema.

Así como para deshabilitar las cuentas, el sistema deberá tener mecanismos para poder desbloquearlas.

2.3 PARTICIONAMIENTO LÓGICO Y NÚCLEOS VIRTUALES

En este apartado se estudiará en profundidad la virtualización y los núcleos virtuales. A continuación se comenzará viendo qué es la virtualización y cómo funciona.

2.3.1 HISTORIA DE LA VIRTUALIZACIÓN

La virtualización comenzó hace bastante tiempo. IBM ya en los años 60 desarrolló un sistema de particiones lógicas que actualmente equivaldrían a máquinas virtuales, las cuales corrían en los *mainframe*s de la época. Al crear estas particiones lógicas se solventaba la necesidad de tener varias máquinas independientes. Este tipo de arquitectura se sigue utilizando durante un tiempo pero con la irrupción de la arquitectura x86 en los 80 se cambia el modelo en muchas empresas. Cambian su estructura de un *mainframe* único y muchos terminales por múltiples máquinas algunas como servidores, con lo cual la necesidad de virtualización se va perdiendo. Ya en los 90 comienza a sentirse la necesidad de la virtualización puesto que se ve absurdo el tener muchas máquinas realizando cada una una tarea pudiendo simplificar más el sistema invirtiendo menos dinero y facilitando la administración.

En esta nueva era de virtualización con la arquitectura x86, los programas que permitían virtualizar sistemas operativos dentro de otro sistema operativo anfitrión tenían un rendimiento bastante malo dado que las máquinas y sistemas de virtualización no estaban tan avanzadas como ahora. Al comienzo empezaron a aparecer sistemas que permitían correr sistemas operativos virtuales *workstation* para luego pasar a correr sistemas servidores, sistemas *hypervisor* y paravirtualización.

Actualmente la virtualización está siendo muy utilizada en el ámbito empresarial por las ventajas que ofrece, puesto que resulta más económica al requerir menos hardware, se consume menos energía y espacio al utilizar menos máquinas y la administración es más fácil y segura. Hoy en día, los fabricantes de hardware ya incluyen en su arquitectura soporte para la virtualización, con lo cual se va consiguiendo que los sistemas virtualizados sean más eficientes y menos complejos.

2.3.2 CONCEPTO DE VIRTUALIZACIÓN. TIPOS

La virtualización en una máquina *host* o anfitriona corre como una capa de virtualización que funcionaría como otra aplicación cualquiera a la cual se le asignan recursos como el procesador, la memoria, espacio de almacenamiento y dispositivos de entrada/salida como una tarjeta de red. Esta capa de virtualización permite instalar y ejecutar sobre ella sistemas operativos huéspedes los cuales tendrán una serie de recursos virtuales (procesador, memoria, almacenamiento y dispositivos de entrada/salida). La capa de virtualización ofrece unos *drivers* virtuales con los cuales se entenderán las aplicaciones y el sistema operativo de la máquina huésped.

Esta simplificación de convertir una máquina en una aplicación tiene las ventajas de poder clonarla, hacer un *backup* (basta simplemente con copiar unos archivos), pausarla, reanudarla, exportarla, añadirle más recursos (memoria, espacio en disco, etc.).

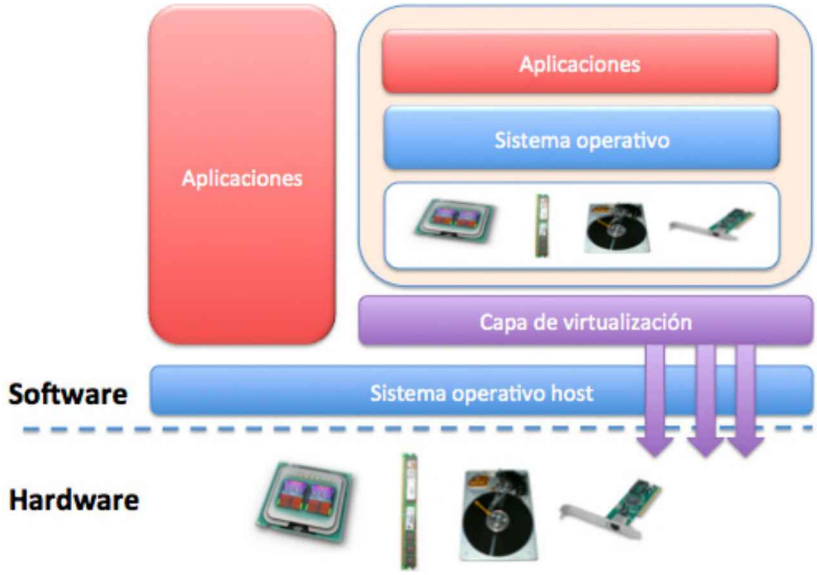

Figura 2.18. *Arquitectura de virtualización clásica*

En sistemas más exigentes se tiende hacia la arquitectura *hypervisor*. La arquitectura *hypervisor* tiene un sistema operativo especial en el que no se pueden instalar aplicaciones, solamente se pueden instalar máquinas virtuales en él. Está reducido al mínimo para ofrecer mejor rendimiento a las máquinas virtuales y ofrece mucha más eficiencia, robustez y escalabilidad que la virtualización clásica. Al no correr aplicaciones en este sistema, los recursos están disponibles enteramente para las máquinas virtuales que corren en él. El sistema solamente se encarga de repartir los recursos entre las disintas máquinas que corren sobre él. Generalmente este tipo de arquitectura ofrece un interfaz web mínimo en el que se pueden gestionar las distintas opciones que tienen las máquinas virtuales (arrancar, parar, clonar, pausar, etc.).

Figura 2.19. *Arquitectura hypervisor*

2.3.3 IMPLEMENTACIONES DE LA VIRTUALIZACIÓN

Existen tres tipos diferentes de implementación de la virtualización:

- **Virtualización total o traducción binaria de órdenes**. La virtualización total o traducción binaria de órdenes consiste en recoger todas las instrucciones de las máquinas virtualizadas y ejecutarlas directamente en la máquina anfitriona. Concretamente se traducirán todas las instrucciones del sistema operativo mientras que las instrucciones de las aplicaciones se ejecutarán tal cual. Todos los recursos de la máquina huésped serán virtualizados (incluso tendrá una BIOS virtual). De esta manera se consigue que un gran número de instrucciones se ejecuten sin modificar de forma nativa.
- **Paravirtualización**. La paravirtualización es una técnica introducida por Xen la cual no necesita virtualización asistida por el hardware. Lo que hace la paravirtualización es modificar el *kernel* del sistema operativo para que se integre con el *hypervisor* y funcionen ambos de una manera más eficiente. Al modificar el sistema operativo se consigue que muchas instrucciones a ejecutar por parte del *hypervisor* se hagan a velocidades casi nativas. Con la paravirtualización las instrucciones de dispositivos virtuales como discos o *nics* se ejecutarán de forma más eficiente que si se utilizase una virtualización total.
- **Virtualización asistida por hardware**. Los fabricantes de hardware, viendo los avances sobre la virtualización y el nuevo mercado que se abre, han optado por ofrecer tecnologías que simplifiquen las técnicas de virtualización y las hagan más eficientes. Concretamente lo que permiten es que las instrucciones de las máquinas virtuales se ejecuten en un modo superusuario con más privilegios que los que puedan tener otro tipo de aplicaciones. De esta manera se consigue que las máquinas virtuales corran de una forma más eficiente.

2.3.4 LOS NÚCLEOS VIRTUALES

Ya desde el año 2002, Intel® puso en práctica su tecnología *Hyper-threading* en procesadores Xeon y Pentium 4. Esta tecnología consiste en crear dos núcleos virtuales a partir de uno real. Esta tecnología consigue un gran aumento del rendimiento utilizando un poco más de recursos. Aunque Intel® dejó de utilizar esta tecnología durante un tiempo, en la tecnología Core i y los Xeon™ vuelve a utilizarla.

Para que esta tecnología sea efectiva, el sistema operativo debe estar preparado para utilizarla. De esa manera al poder ejecutar aplicaciones en varios hilos simultáneos se consigue un mejor tiempo de respuesta y eficiencia por parte del sistema.

La mejora del rendimiento al utilizar esta tecnología de núcleos virtuales se aprecia cuando se ejecutan antivirus en segundo plano o se trabaja con aplicaciones que demandan mucha potencia gráfica.

La ventaja de la utilización de núcleos virtuales permite tener el procesador siempre ejecutando tareas siempre que sea posible. Aunque existen aplicaciones antiguas o poco optimizadas que dificultan la ejecución sobre varios hilos de ejecución, la ventaja de utilizar sistemas operativos modernos permite que dichas aplicaciones no penalicen de forma significativa el rendimiento de todo el sistema.

Es importante recalcar que aunque un procesador tenga 4 núcleos virtuales nunca va a ser igual que un procesador con 4 núcleos reales. Un procesador con 2 núcleos reales y 4 virtuales será mejor que uno con solo 2 núcleos reales pero nunca llegará a tener el rendimiento de un procesador con 4 núcleos reales.

TEST DE CONOCIMIENTOS

1 Elige la respuesta falsa:

a) Se llaman recursos apropiativos a aquellos que pueden ser retirados sin causar ningún daño.

b) En el multiproceso de forma asimétrica se designa un procesador como el procesador *master* (maestro) y los demás serán *slave* (esclavos) de este.

c) La virtualización total o traducción binaria de órdenes consiste en recoger todas las instrucciones de las máquinas virtualizadas y ejecutarlas directamente en la máquina anfitriona.

d) En Linux, se pueden establecer permisos en un fichero o directorio para el usuario, el grupo al que pertenece el fichero y el administrador del sistema.

2 Elige la respuesta falsa:

a) Los algoritmos de planificación deberán basarse en la equidad, eficiencia, imparcialidad, tiempo de respuesta y rendimiento del sistema.

b) En un sistema operativo multitarea, varios procesos se pueden estar ejecutando aparentemente al mismo tiempo sin que el usuario lo perciba.

c) Todos los procesos tienen un PID diferente.

d) Un semáforo se utiliza en sistemas operativos multiproceso para coordinar el flujo de información entre procesos.

3 Elige la respuesta falsa:

a) Un cambio de contexto de un proceso es un paso de un estado a otro.

b) La arquitectura *hypervisor* tiene un sistema operativo especial en el que no se pueden instalar aplicaciones, solamente se pueden instalar máquinas virtuales en él.

c) Físicamente, el sector es la unidad de transferencia que el disco se encarga de leer o escribir.

d) El comando kill envía una señal LOGOUT que indica al programa que tiene que terminar.

4 Elige la respuesta falsa:

a) La virtualización comenzó en los 80 con la irrupción de la arquitectura x86.

b) Una de las funciones principales de un sistema operativo es la de gestionar de forma adecuada los recursos del sistema.

c) La paravirtualización es una técnica introducida por Xen la cual no necesita virtualización asistida por el hardware.

d) En cuanto a la seguridad, los procesos pueden ser más fiables que los hilos de ejecución.

5 Elige la respuesta falsa:

a) Los hilos al igual que los procesos comparten la memoria.

b) El sistema operativo es el primer programa o software que se carga en la máquina y se ejecuta al arrancar o reiniciar el sistema siendo automática su ejecución.

c) En la virtualización asistida por hardware se permite que las instrucciones de las máquinas virtuales se ejecuten en un modo superusuario.

d) Existen procesos en el sistema que nunca terminan como pueden ser los demonios o *daemons*.

6 Elige la respuesta falsa:

a) FIFO es un algoritmo parecido al FCFS pero con algunas variaciones.

b) El *middleware* de un sistema operativo móvil son una serie de librerías que permiten controlar

funciones básicas del sistema como librerías multimedia, librerías gráficas, etc.

c) Los usuarios en un sistema Linux se almacenan en el fichero */etc/passwd*.

d) Un proceso puede tener varios hilos en ejecución.

7 Elige la respuesta falsa:

a) Un semáforo se utiliza en sistemas operativos multiproceso para restringir el acceso a un recurso.

b) El comando aux proporciona de forma interactiva una visión rápida de los procesos que se están ejecutando en ese mismo instante en el sistema.

c) Los sistemas de ficheros suelen confiar su seguridad a mecanismos con ACL o listas de control de acceso.

d) Los hilos tienen que programarse de forma más cuidadosa que los procesos.

8 Elige la respuesta falsa:

a) El sistema operativo Android está basado en el *kernel* de Linux.

b) Los directorios en Unix son considerados como ficheros.

c) Los procesos dentro de su ciclo de vida pueden estar en tres estados, en ejecución, activo y en espera.

d) Se llaman recursos no apropiativos a aquellos que al ser retirados el proceso propietario falla.

9 Elige la respuesta falsa:

a) Los sistemas de ficheros con *journaling* trabajan con transacciones.

b) En el procesamiento de forma simétrica todos los procesadores realizan las mismas funciones.

c) HPFS es un sistema de archivos que apareció en 1993 con Windows NT.

d) En cuanto al rendimiento es mejor utilizar hilos que procesos.

10 Elige la respuesta falsa:

a) El sistema operativo iOS está basado en el *kernel* de Mac OS.

b) Un proceso activo es cuando un proceso está a la espera de ser ejecutado.

c) En la mayoría de sistemas, un sector ocupa 512 bits.

d) El comando kilo envía una señal TERM que indica al programa que tiene que terminar.