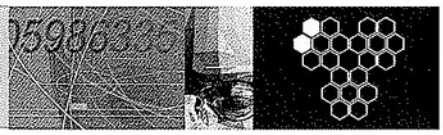


Introducción

El *sistema operativo* es el **software básico** del ordenador. Este software gestiona todos los recursos hardware del sistema informático y proporciona la base para la creación y ejecución del software de aplicación.

Se define como **sistema operativo** (SO) al conjunto de programas, servicios y funciones que gestionan y coordinan el funcionamiento del hardware y del software. El sistema operativo identifica y reconoce al hardware, y el sistema informático comienza a funcionar. Posteriormente, gracias a los programas y aplicaciones del propio sistema operativo, el usuario podrá realizar determinadas funciones. Con el software de aplicaciones funcionando *encima* del sistema operativo, el usuario completará las necesidades de utilización del sistema informático.

Por otro lado, el SO permite al usuario comunicarse con el ordenador, bien mediante el teclado (entorno o interfaz texto), o mediante otros dispositivos, como el ratón (entorno o interfaz gráfica); realiza todo el trabajo dentro del equipo; hace transparente al usuario el hardware del ordenador. El usuario lo utiliza, pero se despreocupa de gestionarlo o administrarlo. Gracias a una **interfaz** sencilla (medio de comunicación entre el usuario y el equipo) proporciona al usuario una comunicación directa, sin que éste tenga que preocuparse de la gestión de cualquier recurso o componente hardware.

Se puede hacer una primera clasificación de los sistemas operativos teniendo en cuenta la gestión que hacen del software y hardware y de cómo el usuario lo puede utilizar:

- *Sistemas operativos monousuario.*
- *Sistemas operativos multiusuario.*

Un **sistema operativo monousuario** (SOMO) permite que los recursos hardware y el software que se esta utilizando, estén a disposición de un único usuario en un único ordenador.

Un **sistema operativo multiusuario** (SOMU) permite que varios usuarios pueden utilizar los recursos software y hardware de un mismo ordenador.

Es evidente que el diseño, eficacia y funciones de un SOMO son inferiores a las de un SOMU. Un SOMO controlará la impresora, por ejemplo, pero para un único usuario, por lo que la gestión de la misma será muy sencilla. En el caso de un SOMU, si varios usuarios pueden utilizar una misma impresora, el sistema operativo, además de controlar el hardware de la propia impresora, tendrá que controlar el orden y prioridad con que se imprimen los trabajos de impresión que los diferentes usuarios han enviado a la misma. Para ello, necesitará funciones de control de trabajos de impresión; funciones de control de prioridades de impresión; funciones de control de seguridad para indicar qué usuarios pueden imprimir y cuándo.

Actualmente, se podría decir que diseñar un sistema operativo monousuario es «fácil». El diseño de un sistema operativo multiusuario implica cientos o miles más de procesos que el mismo sistema tiene que estar controlando en todo momento, ya que ha de ofrecer la posibilidad de compartir, a varios usuarios, el hardware en el que está instalado.

Posteriormente veremos que la ejecución de procesos en los sistemas operativos multiusuario dependerá de varias características y, especialmente, del hardware del ordenador en el que esté instalado.

La ejecución de un programa en un SOMO es relativamente sencilla. En primer lugar, el programa se inicia desde el disco duro o se carga desde un dispositivo de almacenamiento externo. Este programa se ubica en memoria (proceso que realiza la CPU); una vez allí comienzan a trabajar los componentes de la CPU: primero la *unidad de control* para ir ejecutando las instrucciones una a una y, si fuera necesario, la *unidad aritmético lógica* para realizar algún cálculo. Se seguirá estrictamente el **ciclo de ejecución de una instrucción** paso a paso, sin ningún otro tipo de complicación.

La ejecución de un programa en un SOMU es bastante más compleja, dado que trabajan de diferente forma, en función del hardware donde estén instalados. En este punto, se pueden clasificar los SOMU en función del hardware donde estén instalados:

- SOMU instalado en un gran sistema *(mainframe)*.
- SOMU instalado en ordenadores personales.

Un gran sistema *(mainframe)* permite que varios teclados y monitores se conecten a él a modo de terminales; es decir, el usuario que utiliza estos sistemas, no tiene un ordenador propiamente dicho, sino que a través de un teclado y monitor conectados al *mainframe*, utiliza los recursos de éste para realizar su trabajo.

Supongamos ahora que hay cuatro usuarios utilizando un *mainframe*. Cada usuario inicia la ejecución de un programa distinto. El sistema operativo tendrá que trabajar bastante para ubicar en memoria cuatro programas diferentes, e ir ejecutando, de forma intercalada, instrucciones de cada uno de los programas. Es evidente que el tiempo de respuesta que obtendrá cada usuario a la ejecución de su programa será mayor que si lo ejecutara en un SOMO, o si el *mainframe* sólo ejecutara su programa.

El proceso que se sigue en el caso anterior, cuando hay cuatro usuarios utilizando un *mainframe*, es el siguiente. En primer lugar, se ejecutan unas instrucciones del programa iniciado por el primer usuario. A continuación, se detiene la ejecución de este pro-

grama y se concede tiempo de CPU a la ejecución del programa iniciado por el segundo usuario. Este proceso se repetirá secuencialmente para ejecutar todos los programas de todos los usuarios, pero nunca de forma simultánea. La CPU no puede ejecutar dos instrucciones a la vez, pero sí puede conseguir ejecutar primero una y después otra, aunque sea de un programa diferente en un intervalo de tiempo tan breve que los usuarios que están ejecutando los programas no aprecien la detención de los mismos. Este tipo de sistemas operativos sólo se utiliza, como se ha comentado, en grandes sistemas o *mainframes*.

Los SOMU más conocidos son los que se ejecutan en los ordenadores personales actuales de altas prestaciones. En éstos, cada usuario dispondrá de un ordenador personal, con su propio sistema operativo. El usuario encenderá su ordenador y establecerá comunicación con el ordenador principal que dispone del sistema operativo multiusuario, y que realiza las funciones de servidor.

En este caso, si partimos del mismo ejemplo anterior con cuatro usuarios ejecutando un programa diferente cada uno, dispondremos de cinco CPU, cinco bloques de memoria independientes, cinco o más dispositivos de almacenamiento, etcétera. Cuando los cuatro usuarios indiquen al ordenador principal, el que tiene el SOMU, que van a ejecutar un programa, éste puede hacer varias cosas. La primera de ellas es delegar todo el trabajo en el ordenador del usuario que ha iniciado el programa; de esta forma el ordenador principal sólo realiza funciones de *servidor de aplicaciones*, pero ni su procesador ni su memoria se utilizan para procesar el programa del usuario. Si los cuatro solicitan la ejecución de un programa en particular, el ordenador principal se dedicará a iniciar el programa en los ordenadores *cliente* y todo el proceso lo gestionará el sistema operativo de cada usuario, su procesador y su memoria. No obstante, en la mayoría de los casos, los datos que se deben procesar sí estarán físicamente en el servidor y no en los ordenadores cliente. Los clientes ejecutan el programa, pero los datos los almacena y controla el servidor.

Algunos SOMU también controlan y gestionan las aplicaciones de usuario, de tal forma que cuando un usuario inicia un programa, éste se ejecutará en el ordenador principal. El cliente necesitará un pequeño software, denominado **software cliente,** para poder trabajar con la aplicación o programa deseado.

Un sistema operativo, dentro del sistema informático, es el motor de todo; hace de intermediario y controlador entre la parte física del ordenador, el software que se utiliza y el usuario para gestionar y administrar los recursos. Los recursos hardware y software que controla o gestiona el sistema operativo son los siguientes:

- Procesador.
- Memoria interna.
- Periféricos de E/S.
- Información.

2.1 Evolución histórica

Los primeros sistemas operativos se denominaron **monolíticos.** Eran software básico, prácticamente imposible de modificar una vez creado e instalado en un sistema informático. Cuando los diseñadores del propio sistema operativo, o los usuarios, por necesidades específicas, querían introducir modificaciones en él, la labor era realmente complicada, ya que era necesario volver a configurar todo el sistema operativo.

Actualmente, la mayoría de los sistemas operativos son *abiertos* y *segmentados*. Esto permite, en primer lugar, modificarlos fácilmente si fuera necesario, ya que el conjunto de programas que lo componen están divididos por bloques. Están diseñados de tal forma que un bloque de programas controla los periféricos de entrada, por ejemplo; otro bloque los periféricos de salida; otro los dispositivos de almacenamiento, otro las comunicaciones, etcétera.

De esta forma, si es necesario modificar, ampliar o cambiar el sistema operativo por la aparición de nuevos sistemas de almacenamiento, por ejemplo, solamente será necesario modificar el conjunto de programas destinados a tal fin y no modificar nada, o prácticamente nada, del resto.

Los sistemas operativos actuales están estructurados *por niveles*. Cada nivel, o capa del sistema operativo, realiza una función. De esta forma, la modificación o ampliación afectará a un nivel determinado, y no a todo el sistema operativo.

La última generación de sistemas operativos la componen aquellos que se denominan sistemas operativos *de máquina virtual*. Estos sistemas operativos, a diferencia de los anteriores, tienen un núcleo que permite emular el hardware. De esta forma, cada proceso o programa que se inicia se ejecuta en un espacio de memoria totalmente independiente. Además, cada programa o proceso iniciado dispone de una copia «virtual» del hardware. Es necesario tener muy presentes las arquitecturas de los ordenadores, es decir, la evolución del hardware sobre el que se instalan.

Históricamente, se ha hablado de cuatro generaciones de ordenadores, y quedan definidas las características de cada una de ellas, por los componentes hardware de los sistemas informáticos que los componen. Hardware y sistema operativo evolucionan el uno con el otro, y nunca de forma independiente. Si se diseñan sistemas operativos más eficaces, es debido a que el hardware sobre el que van a funcionar también lo es, y a la inversa: se diseña

hardware más eficaz y rápido debido a que los sistemas operativos necesitan cada vez mayores prestaciones hardware.

La primera computadora fue diseñada por el matemático inglés **Charles Babbage,** que ya tenía clara la secuencia a seguir para el tratamiento automático de la información:

Entrada → Procesamiento → Salida

Posteriormente, **George Boole** elaboró la teoría de la lógica matemática y el álgebra que lleva su nombre. Gracias a éste se pudo empezar a pensar en la elaboración de procesos o programas que dependiendo de unas condiciones u otras realizarán unos procesos u otros.

En general, es posible hablar de varias generaciones de computadoras, relacionándolas con la evolución de los sistemas operativos, como se ha comentado.

Primera generación (1945-1955)

Se utilizaban las válvulas de vacío (antiguas resistencias electrónicas). Estas computadoras eran máquinas programadas en lenguaje máquina (lenguaje de bajo nivel), de gran tamaño, elevado consumo de energía y muy lentas a la hora de realizar operaciones, que se reducían a simples cálculos matemáticos. Eran peores que las actuales calculadoras de bolsillo: muchas menos funciones, mucho más lentas, eran máquinas carísimas, etc.

La forma de introducir los datos en estas computadores se hacia a modo de centralita de teléfonos antigua, pinchando clavijas en unos paneles enormes llenos de agujeros. Según se pinchaba la clavija en uno u otro lugar, se indicaba que números se deseaba procesar y qué operación se quería realizar. Posteriormente, a principios de los años cincuenta, para introducir datos en la computadora se utilizaban las tarjetas perforadas. Permitían introducir más datos y de forma más rápida aunque, lo más importante era que se podía repetir el mismo proceso sin tener que volver a introducir de nuevo todos los datos de forma manual.

Segunda generación (1956-1965)

Se integran los *transistores* dentro de la arquitectura de las computadoras. Desaparecen las válvulas de vacío, por lo que las computadoras se hacen más pequeñas, económicas y de menor consumo. Las personas encargadas de la utilización del sistema informático se dividen en categorías: el perforador de tarjetas, el operador de consola, etcétera.

En esta generación aparece lo que se denomina **procesamiento por lotes,** que consiste en que los datos se introducen en la computadora mediante un pequeño componente hardware que previamente ha sido cargado con la información a procesar. Es evidente que la carga de este pequeño componente hardware todavía sigue siendo manual. El procesamiento por lotes implica tres fases:

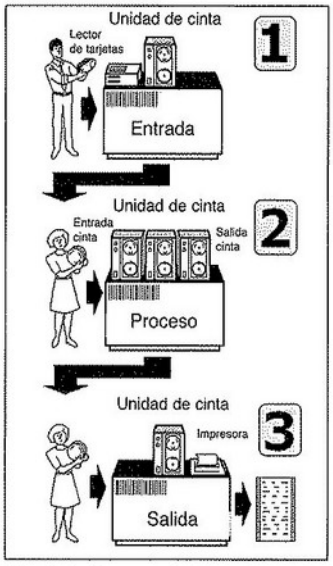


Fig. 2.1. *Procesamiento por lotes.*

- Introducción de los datos a procesar en un componente hardware, que puede ser una tarjeta perforada, un tambor magnético, etc. La introducción de datos se realizaba en un medio físico distinto de la computadora que procesaba la información.
- Introducir el soporte con los datos a la computadora. Se procesaba la información y, a continuación, se almacenaba en otro soporte diferente.
- El soporte donde se almacenan los resultados se lleva a otro dispositivo físico distinto a la computadora para realizar la generación de resultados.
- Aparece el concepto de periférico. En la primera generación todo era una misma cosa.

Tercera generación (1966-1980)

Se reduce considerablemente el tamaño y consumo de energía de las computadoras gracias a la sustitución de los transistores por los *circuitos integrados*. En esta generación, cabe destacar la computadora IBM 360 como máquina capaz de realizar cualquier tipo de cálculo, ya fuera aritmético o lógico; el gran salto es el diseño de hardware y software básico que permite a una máquina o sistema informático realizar varios procesos a la vez (recordemos los SOMU).

Al principio, aunque se podían ejecutar varios programas, esto se hacía en un orden estricto de llegada. Pero también en esta generación aparecen los SOMU, que permiten ejecutar varios programas a la vez, no simultáneamente, pero ofreciendo al usuario un

tiempo de espera medio mayor que si ejecutase el programa él solo.

El software básico que se diseña para gestionar estas computadoras tenía que ser capaz de controlar, gestionar y relacionar los diferentes componentes de un mismo ordenador.

Cuarta generación (1981 hasta la actualidad)

Se utilizan complejas técnicas de integración y miniaturización de componentes electrónicos. Aparecen las memorias de semiconductores, los dispositivos de almacenamiento externo de pequeño tamaño (discos duros actuales), los dispositivos ópticos, etc. Estos componentes son cada vez más rápidos, más económicos y, sobre todo, potencialmente utilizables por personas que no necesitan ser especialistas en informática. Aparecen sistemas operativos mucho más intuitivos y fáciles de utilizar para el usuario. La comunicación entre usuario y computadora se facilita e integra en el sistema operativo.

Estos nuevos sistemas operativos diseñados para este nuevo hardware interactúan con el usuario a través de interfaces sencillas. Nacen sistemas operativos como el DOS, sistemas operativos en red y, actualmente, algunos como Microsoft Windows, en sus diferentes versiones, que permiten un diálogo con la computadora basado en entornos gráficos. Estos sistemas operativos actuales son muy eficaces, sobre todo en la gestión de hardware y en la utilización y distribución de programas y datos en memoria. El sistema operativo **UNIX** se afianza como SOMU, mejora con el tiempo y aparecen cada vez mejores versiones, con menos errores, más fáciles de usar y en entorno gráfico, X-Windows.

2.2 Tipos de sistemas operativos. Sistemas de explotación

Las formas de explotación de un sistema operativo responden a cómo el usuario utiliza los recursos hardware y software que componen el sistema informático. *Explotar* un sistema operativo significa utilizarlo.

En un SOMO, como sólo se ejecuta un programa por vez, la cuestión no tiene mayor importancia a priori. Pero en un SOMU, dado que se pueden ejecutar varios programas a la vez, como ya se ha comentado anteriormente, el procesador no es capaz de ejecutar dos instrucciones en el mismo intervalo de tiempo. Solamente puede atender las peticiones de un solo proceso o programa por vez. El procesador destina **ciclos de CPU** de forma secuencial a cada proceso, para intentar que todos ellos tengan la misma prioridad de ejecución.

Teniendo esto en cuenta, cuando se trabaja con un SOMU, no todos los programas iniciados se encuentran en la misma fase de ejecución. Lo normal es que haya uno en proceso y otros en espera, ya sea por necesidades hardware, software o de procesador. Esto se describe con más detalle posteriormente.

Para hablar de cómo se explota un sistema operativo, hemos de tener en cuenta también las necesidades de los usuarios que lo utilizarán y, por supuesto, todo siempre relacionado con el hardware sobre el que estén montados los sistemas operativos.

Para realizar la clasificación de los diferentes modos de explotación, consideraremos cuestiones tales como el número de usuarios que pueden utilizar el sistema, los procesos que dicho sistema puede ejecutar a la vez, el número de procesadores con los que cuenta el ordenador y el tiempo de respuesta del sistema.

De forma general, un sistema operativo se puede explotar de dos formas:

Procesamiento por lotes. Este sistema de explotación se empezó a utilizar en la segunda generación de ordenadores. El proceso que sigue es el siguiente: en primer lugar se carga el soporte de almacenamiento externo con los datos que se desea procesar; a continuación, éstos se llevan al ordenador que los procesará. Una vez procesada la información, los resultados se vuelven a cargar en otro soporte de almacenamiento externo que, conectado al correspondiente periférico de salida, permite mostrar los resultados. En la Figura 2.1 se ilustraba el procesamiento de datos por lotes.

En este tipo de procesamiento, actualmente en desuso, el tratamiento de la información atraviesa diferentes fases. En primer lugar, se carga toda la información en los soportes; a continuación, se procesa y, posteriormente, se muestra o imprime. Cada fase requiere un tiempo determinado, existe la posibilidad de que se produzcan errores y, por tanto, es posible que se interrumpa el tratamiento de la información si alguna de las fases no se realiza correctamente.

El programa que se encarga de almacenar la información, procesar los datos y mostrar o imprimir el resultado del procesamiento es el sistema operativo. De esta forma se consigue que el procesador sólo se dedique a *procesar información*. Los tres procesos los realiza el sistema operativo de forma secuencial.

Procesamiento en tiempo real. Este tipo de explotación del ordenador es similar al anterior. La diferencia radica en que el usuario que introduce los datos es el que suele iniciar el programa

Por tanto, el resto de usuarios podrán seguir ejecutando sus programas sin más problemas. Es decir, el bloqueo de un proceso que puede estar ejecutando un usuario no afecta al resto de procesos, ya que cada uno de ellos, incluso para el mismo usuario, se ejecuta en una máquina virtual diferente, por lo que el hardware de la máquina no se ve afectado.

Una vez que el usuario termina de ejecutar su aplicación, se produce la interacción real con el hardware. El sistema operativo termina de ejecutar correctamente la aplicación, el resto de usuarios siguen trabajando con sus máquinas virtuales y, evidentemente, el hardware funciona sin problemas. Se descarga el contenido de los archivos que forman la máquina virtual al hardware, momento en que se produce la operación real de entrada y salida sobre el periférico o dispositivo de almacenamiento.

La Figura 2.11 muestra un esquema del funcionamiento de las máquinas virtuales respecto de tres procesos iniciados por un mismo usuario.

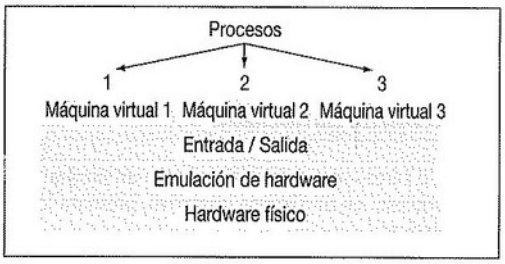


Fig. 2.11. *Máquinas virtuales.*

Fig. 2.12. *Jerarquía de los sistemas.*

2.4 Servicios de sistemas operativos

La tarea principal de un sistema operativo consiste en coordinar el uso del hardware en función de los programas o aplicaciones que se estén utilizando. Los programas que se utilizan en la mayoría de los casos los elige el usuario, pero en otras muchas ocasiones son programas propios del sistema operativo los que tienen que estar funcionando para conseguir que los programas de usuario cumplan con su objetivo.

La comunicación entre los diferentes niveles se realiza mediante las llamadas **interfaces,** que son programas o *servicios* que se ejecutan en el ordenador y que relacionan los niveles para que el usuario final pueda acceder al hardware y ejecutar sus programas. Un **servicio** es un tipo de aplicación que normalmente se ejecuta en segundo plano y es similar a las aplicaciones llamadas *demonios* en entornos UNIX *(daemon)*. Los servicios proporcionan a los usuarios funciones que les permiten utilizar los recursos de SO. Algunos de los servicios iniciados por los sistemas operativos son aplicaciones del tipo cliente-servidor, servidores Web, servidores de bases de datos y otras aplicaciones basadas en servidores, tanto de forma local como a través de una red.

Estos servicios pueden utilizarse para:

- Ejecutar esos programas, proveyendo al sistema de los recursos hardware y software necesarios.
- Acceder de forma controlada a los dispositivos de entrada y salida.
- Acceder de forma controlada y segura a los archivos y a la información.
- Controlar y solucionar errores provocados por el hardware o el software.
- Proporcionar información estadística, de seguridad y registro de lo que se hace en el sistema.
- Etcétera.

En general, los servicios se utilizan para iniciar, detener, pausar, reanudar o deshabilitar programas y aplicaciones (que a su vez pueden ser otros servicios) en equipos locales y remotos. Esta función la realizará normalmente el administrador del sistema informático, que tiene los privilegios adecuados para realizar tales acciones.

En un sistema informático, la mayoría de los servicios se integran al instalar el propio SO. Muchas aplicaciones, especialmente las que utilizan servicios de red, instalan sus propios servicios, que se agregan a los ya integrados en el sistema operativo.

Los servicios son esenciales para el funcionamiento de muchas de las aplicaciones y del propio sistema operativo; son programas que, una vez instalados, se ejecutan automáticamente al iniciar el SO, es decir, al poner en marcha el sistema informático.

2.5 Procesos

A. Conceptos generales

Un proceso, o tarea, se puede definir como un programa en ejecución. Un proceso, en un sistema operativo, presenta las siguientes características:

- Para iniciar su ejecución debe cargarse completamente en memoria y tener asignados todos los recursos que necesita.
- Ha de estar protegido del resto de procesos; ningún otro proceso podrá escribir en la zona de memoria perteneciente a ese proceso.
- Puede pertenecer al usuario o ser propio del sistema operativo. Los procesos pertenecientes a los usuarios se ejecutan en el modo denominado *modo usuario del procesador* (con restricciones de acceso a los recursos hardware). Los procesos que pertenecen al sistema operativo se ejecutan en el *modo núcleo* o *modo privilegiado del procesador* (sin restricciones de acceso a los recursos hardware).
- Cada proceso tendrá una estructura de datos llamada *bloque de control de proceso*, donde se almacenará información acerca del proceso.
- Cada proceso puede comunicar, sincronizarse y colaborar con los demás. Estas operaciones se realizan mediante:
  - **Comunicación:** memoria compartida e intercambio de mensajes.
  - **Sincronización:** semáforos.
  - **Colaboración:** mediante llamadas a procedimientos locales y remotos.

  La razón principal de estas operaciones es que, al residir cada proceso en zonas de memoria independientes, se ha de llamar al sistema para compartir los datos entre los procesos.

Además de las características anteriores, hay que tener en cuenta que a cada proceso se le asigna un espacio de direcciones lógicas en el que reside el proceso (en la parte baja) y las llamadas al sistema (en la parte alta), para tener acceso directo a los recursos del sistema.

Este espacio de memoria es igual al máximo que el sistema operativo instalado en el equipo sea capaz de gestionar (en un sistema operativo de 32 bits es de 4 GB), y aquí entra en juego la memoria virtual o cualquier otra técnica de gestión de memoria.

Los procesos se dividen en partes de igual tamaño, llamados *páginas* o *marcos*. Cuando se carga un proceso, lo que se hace es ubicarlo en memoria y asignarle un número máximo de bloques de memoria. Para esta operación se utilizarán técnicas de gestión de memoria como la memoria virtual, el intercambio de memoria, la paginación, la segmentación, etc. La gestión de memoria se describe posteriormente en esta Unidad.

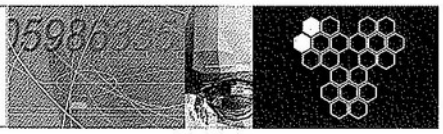
Cuando se ejecuta el proceso, si se desea acceder a una parte del mismo que no está cargada en memoria interna, se busca en la zona de memoria virtual y se carga en memoria interna. Una vez alcanzado el número máximo de espacios de memoria interna utilizados por un proceso, se descargan a memoria virtual las partes de memoria que no se estén utilizando, y se cargan en memoria interna las partes nuevas del proceso que se está ejecutando. Las partes del proceso que se descargan de memoria interna y pasan a la memoria virtual, para dejar paso a la nueva parte del proceso que se está ejecutando, suelen ser las que se utilizan poco o que se cargaron en primer lugar.

El problema de este sistema es que, si un programa está mal programado o construido, pueden ocurrir problemas de *hiperpaginación*. Este fenómeno ocurre cuando el número de zonas de memoria asignadas a un proceso desciende por debajo del número mínimo necesario. En este caso, será necesario suspender la ejecución de ese proceso y descargar los espacios de memoria asignados. En general, cualquier proceso que no cuente con suficiente espacio de memoria provocará *errores de página* muy frecuentemente. Si se reemplazan partes del programa que están activas, estaremos sustituyendo una parte del proceso que casi de inmediato se volverá a necesitar. Esta altísima actividad de paginación se llama **hiperpaginación.** Se dice que un sistema se encuentra en hiperpaginación si utiliza más tiempo paginando que ejecutando.

Para poder ejecutar un proceso, deberá estar siempre cargado en memoria principal, pero no sólo las instrucciones del propio código que lo componen, sino también los datos a los que afecta la ejecución del mismo.

Un programa no es un proceso siempre y cuando no esté en ejecución. Por ejemplo, Microsoft Word será simplemente un archivo almacenado en el disco duro si no se está ejecutando. Una vez que se ejecute, el archivo Microsoft Word seguirá almacenado donde estaba originalmente, pero las instrucciones necesarias para su ejecución se habrán cargado en memoria. Una vez que se está ejecutando el programa, se convierte en proceso.

El reparto de los recursos del sistema entre los distintos procesos y su ejecución concurrente se conoce como **multiprogramación.**



B. Bloque de control de proceso y bloque de control de sistema

Cuando se ejecuta más de un proceso de manera concurrente en un sistema, todos ellos necesitan que el sistema les suministre una serie de recursos. Para ello, el propio sistema operativo se encarga de asignar estos recursos en un orden adecuado y atendiendo a unas prioridades.

Cada vez que un programa se convierte en proceso, es decir, cada vez que se ejecuta, además de ubicarse en memoria las instrucciones que permiten su ejecución y del procesamiento de datos que pueda utilizar, se le asocia una *estructura de datos*. Esta estructura de datos, que es única para cada proceso, identifica el proceso respecto de los demás y permite controlar su correcta ejecución. A la estructura de datos se le denomina **bloque de control de proceso** y contendrá, por cada proceso, lo siguiente:

- **Estado actual del proceso.** Un proceso puede estar en ejecución, detenido o bloqueado. Este último caso es el más conflictivo, ya que los bloqueos pueden provocar que otros procesos también dejen de funcionar, especialmente si el bloqueo es provocado por un componente hardware y varios procesos utilizan el mismo componente.

- **Identificador del proceso.** A cada proceso, dependiendo del sistema operativo, se le asigna un **PID** o código de identificador de proceso. Este código es normalmente un número que el propio SO asigna según unas prioridades y unos parámetros de diseño preestablecidos teniendo en cuenta el uso y funcionalidad de cada proceso.

- **Prioridad del proceso.** La prioridad de cada proceso la asigna de forma automática el SO en función de los parámetros con los que se ha diseñado el mismo. La prioridad de los procesos puede ser modificada por los usuarios con privilegios para ello (normalmente el dueño de un proceso podrá modificar su prioridad, y el administrador del sistema podrá modificar la prioridad de cualquier proceso de todos los usuarios).

- **Ubicación en memoria.** Teniendo en cuenta la técnica utilizada para ubicar los programas en memoria, y dependiendo del tipo de programa de que se trate, el SO tendrá que ubicar cada proceso en una zona independiente de memoria.

- **Recursos utilizados.** Cada proceso debe tener a su disposición determinados recursos hardware y algunos recursos software para poder ejecutarse. Estos recursos se pondrán a disposición del proceso en el mismo momento que comience a ejecutarse. Como es habitual, la asignación de estos recursos la realizará el SO.

A continuación, se enumera lo que debe contener el bloque de control de procesos:

- Identificador del proceso (PID).
- Prioridad.
- Estado del proceso (ejecución, detenido, bloqueado).
- Estado hardware (registros y etiquetas del procesador).
- Información de planificación y estadísticas de uso.
- Información de gestión de memoria.
- Estado de E/S (dispositivos asignados, operaciones pendientes).
- Información de gestión de archivos (archivos abiertos, derechos).
- Información de mantenimiento.

A continuación, se detalla cómo un programa pasa a ser un proceso, es decir, cómo se inicia y finaliza su ejecución. Es necesario suponer que se utiliza un ordenador con un sistema operativo instalado y funcionando correctamente, y que se desea ejecutar un programa cualquiera.

Cuando escribimos en el indicador de comandos (**shell** en UNIX, **símbolo del sistema** en DOS) el nombre de un fichero ejecutable o hacemos **doble clic** con el ratón sobre el icono que representa un programa ejecutable, se carga en memoria un proceso del propio sistema operativo, denominado **cargador.** El cargador prepara el programa para iniciar su ejecución y realiza las siguientes funciones:

- Crea el bloque de control de proceso (BCP). Se le asigna un identificador (PID), **una prioridad base,** así como todos los recursos necesarios, excepto la CPU.
- Se inserta en la tabla de procesos del sistema.
- Se carga en memoria virtual (según la técnica de gestión de memoria utilizada).
- Cuando ya tiene todos los recursos asignados (menos la CPU), el campo de estado del proceso del BCP se cambia a «preparado» y se inserta en la cola de procesos preparados para ser ejecutados por la CPU. El proceso del sistema que controla la cola de la CPU recibe el nombre de **planificador,** y es el encargado de decidir, según las prioridades, qué proceso será el siguiente que ejecutará la CPU.

La prioridad de un proceso no es ni más ni menos que la asignación de ciclos de CPU para que el proceso se ejecute. Si un proceso tiene una prioridad muy baja, puede darse el caso de que nunca se ejecute. Para evitar esto se utiliza el sistema de prioridades dinámicas, es decir, se aumenta la prioridad de un proceso a medida que se incrementa el tiempo de espera en la cola. Los pasos que debe seguir el planificador son los siguientes:

- Asignar nuevas prioridades a los procesos en la cola de la CPU.
- Elegir el proceso con la prioridad más alta.
- Cambiar el campo de estado del proceso elegido a «ejecución».
- Llamar a la rutina de cambio de contexto, que cargará el proceso en la CPU, es decir, copiará el estado hardware en los registros de la CPU; el último registro a actualizar será el registro contador de programa, para que la próxima instrucción a ejecutar sea la siguiente donde se quedó el proceso.

Actualmente, en los sistemas operativos multiusuario y multitarea, se ejecutan varios procesos a la vez. Este paralelismo de tareas o procesos necesita una planificación especial para optimizar el uso de los recursos del sistema, y de ello se encarga el planificador.

Una CPU no puede ejecutar dos o más procesos a la vez. Dada la eficacia actual de los procesadores y la elevada rapidez a la que ejecutan los procesos, el usuario aprecia que los programas se ejecutan simultáneamente, aunque la CPU los ejecute secuencialmente.

Por tanto, cada proceso atraviesa por varias fases, posteriormente estará en espera, mientras la CPU ejecuta otro proceso; otros procesos estarán preparados para iniciar su ejecución, otros pueden estar bloqueados, etc. En estos cambios de proceso, el sistema operativo debe saber qué ficheros están abiertos, qué periféricos se están utilizando, etc. Es importante hablar de la planificación como tarea fundamental a realizar por la CPU en la gestión de procesos.

En general, la mayoría de los ordenadores actuales tienen un único procesador. Eso implica que cuando se están realizando varias tareas a la vez, es necesario compartir el tiempo de trabajo de la CPU. El tiempo compartido resulta de dividir el tiempo de ejecución del procesador en breves intervalos de tiempo **(quantum)** e ir asignando cada uno de esos intervalos de ejecución a cada proceso que se está ejecutando. Así, los ciclos de CPU, sincronizados por el reloj del sistema, se irán asignando a los diferentes procesos. Toda la información referente a esta planificación se almacena en la **tabla de procesos,** que contiene el bloque de control de procesos (estructura en la que se almacena la información de los procesos).

La tabla de procesos contiene las especificaciones, señaladas por el descriptor, de cada uno de los procesos que se están ejecutando, para que cuando vuelvan a ejecutarse se continúe desde el mismo punto en el que se detuvo o pausó la ejecución. El número total de procesos que se ejecutan en el sistema queda determinado por el número de entradas en la tabla de procesos, teniendo en cuenta que los espacios en la tabla de procesos son recursos finitos.

A cada proceso se le asigna un número determinado de *quantums* (unidades de tiempo) de utilización de CPU. Este tiempo podrá ser estático o dinámico dependiendo de la prioridad del proceso. Cuando finaliza este tiempo se produce una interrupción de fin de tiempo de ejecución y se llama al gestor de interrupciones, que a su vez llama al proceso correspondiente para que la gestione. Las principales interrupciones son:

- Interrupción por E/S.
- Interrupción por fin de tiempo de ejecución (en la que se llama al planificador).
- Interrupción por error.

Cuando un proceso solicita una operación de E/S, su estado se cambia a «pausado» y se le coloca en la cola del dispositivo de E/S que desea utilizar. Una vez finalizada la operación de E/S, se vuelve a cambiar su estado a «preparado» y se introduce en la cola de la CPU.

Por ejemplo, si la función de un proceso consiste en escribir una línea por la impresora y le llega el momento de su ejecución, la CPU le concede su tiempo y lo pone en estado de ejecución. En este punto se realiza el cambio de contexto, pasando del estado de «preparado» al estado de «ejecución». Cuando se empieza a ejecutar el proceso con el fin de acceder a la impresora, la CPU le cambia de nuevo el estado a «espera», ya que ahora la CPU está intentando contactar con el periférico deseado. Cuando la CPU obtiene acceso a la impresora, se vuelve a cambiar de contexto; se cambia de nuevo su estado de «espera» a «ejecución». Ahora, la CPU extrae la información de memoria que imprimirá el proceso y realiza la operación.

Por último, finalizada la operación de impresión, la CPU cambiará el estado del proceso a **«zombi»,** que es el aquel en que ha acabado de ejecutarse, pero aún faltan por liberar algunos recursos, por lo que el proceso puede considerarse todavía «vivo» dentro del sistema.

C. Concepto de hebra, multihebra y multiproceso

Una **hebra,** o **subproceso,** es un punto de ejecución de un proceso. Un proceso siempre tendrá, como mínimo, una hebra, en la que se ejecuta el propio programa. Las hebras representan un método software que permite mejorar el rendimiento y eficacia de los sistemas operativos, reduciendo la sobrecarga por el cambio de contexto entre procesos. Las hebras asumirán el papel de los procesos como unidad de planificación. Un proceso será una unidad propietaria de recursos para una serie de hebras.

Un proceso clásico será aquel que sólo posea una hebra. Por ejemplo, si ejecutamos el procesador de textos Microsoft Word, con un

único documento abierto. Microsoft Word convertido en proceso estará ejecutándose en un único espacio de memoria, tendrá acceso a determinados archivos y a determinado hardware.

Si en este momento, sin cerrar Microsoft Word, abrimos un nuevo documento, Word no se vuelve a cargar como proceso, simplemente tendrá a su disposición dos hebras o hilos diferentes, de forma que el proceso sigue siendo el mismo (el original).

En este y otros casos podremos observar, si se queda bloqueado algún documento de texto de los que están abiertos, que no solamente se quedará bloqueado el propio documento; quedarán detenidos todos los hilos mediante los que se ejecutan los diferentes documentos de texto. Por tanto, si no somos capaces de desbloquear el proceso que tiene el problema, no podremos trabajar con ningún documento de texto de todos los que estén abiertos en ese momento. En el peor de los casos, se cerrarán sin darnos opción a grabar las modificaciones que hayamos realizado sobre ellos.

Se inicia la ejecución del proceso, pero la CPU no le asigna sus tiempos para ejecutarlo. La CPU solamente asigna tiempos de ejecución a los hilos de cada proceso. Así, las hebras tendrán un flujo independiente de control para cada una de ellas (punto de ejecución) así como un estado hardware propio. En el caso de sistemas tradicionales multitarea, la asignación de tiempos de la CPU se hace sobre cada proceso en particular.

Todos los recursos (excepto la CPU) son gestionados por el proceso. El proceso es el propietario de la memoria que necesita, del hardware, de los ficheros, etc., a excepción de la CPU. Los tiempos de la CPU son gestionados y asignados a los diferentes hilos de un proceso.

Cuando un hilo se está ejecutando, el resto de hilos estarán en espera, es decir, estarán bloqueados y no se ejecutarán. Habrá solo un hilo ejecutándose y aunque el resto estén detenidos o en espera, el control de todos los componentes hardware, a excepción de la CPU, estarán controlados por el propio proceso y no por los hilos.

De esta forma la comunicación entre las diferentes hebras de un proceso será mucho más rápida y eficiente, porque todas las hebras de un proceso comparten un mismo espacio de memoria.

Actualmente, es normal encontrar sistemas operativos multihebra o multihilo, e incluso procesadores como los últimos **HT** *(HyperThreading)* del fabricante Intel.

Los sistemas operativos que se utilizan habitualmente son multihebra; por ejemplo, Windows NT, Windows 2000, Windows XP, Windows Server 2003, OS/2, NeXTStep, Solaris y varias versiones de UNIX y Linux.

D. Estados, prioridades y planificación de procesos

A continuación, se enumeran los diferentes estados de un proceso (Figura 2.13):

- **En ejecución.** El procesador está ejecutando instrucciones de ese proceso en un instante determinado.
- **Preparado, en espera o activo.** El proceso está preparado para ejecutarse; es decir, espera el turno para poder utilizar su intervalo de tiempo de CPU.
- **Bloqueado.** El proceso está retenido, es decir, está bloqueado por alguna razón. Una de estas razones puede ser que dos procesos utilicen el mismo fichero de datos. Otra puede ser que dos procesos necesiten utilizar la misma unidad de CD-ROM para cargar determinados datos, etcétera.

En general, todos los procesos de cualquier sistema operativo tienen unas características que los identifican. Cada proceso tiene un número asignado por el sistema operativo que sirve precisamente para identificar el proceso, iniciar su ejecución, detenerlo, cancelarlo, reanudarlo, etc. Este identificador de proceso se abrevia como **PID** *(Process IDentificator)*. Cuando se estudien los sistemas operativos, se describirá cómo se puede ver el PID asignado a cada proceso; por ejemplo, con el comando **NETSTAT** es posible ver estos PID en sistemas Windows (familia de servidores y clientes profesionales) y con **PS** se puede hacer lo mismo en versiones UNIX o Linux.

Cada proceso que inicia la ejecución depende, en la mayoría de los casos, de otro proceso denominado *proceso padre*. Así, al nuevo proceso iniciado se le denomina *proceso hijo*.

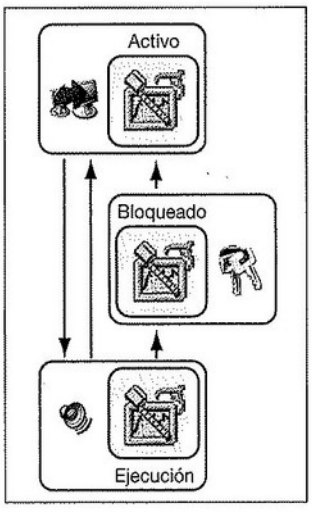


Fig. 2.13. *Estados de los procesos.*

La Figura 2.14 muestra las transiciones que pueden sufrir los procesos entre cada uno de estos tres estados:

- **Transición a.** Tiene lugar cuando el programa que está en ejecución necesita algún elemento, señal, dato, etc., para poder continuar ejecutándose.
- **Transición b.** Tiene lugar cuando un programa o proceso ha utilizado el tiempo asignado por la CPU (procesador) para su ejecución y tiene que dejar paso al siguiente proceso.
- **Transición c.** Tiene lugar cuando el proceso que está preparado pasa a ser ejecutado, es decir, cuando al proceso le llega una nueva disposición del tiempo de la CPU para poder ejecutarse.
- **Transición d.** Tiene lugar cuando el proceso pasa de estar bloqueado a estar preparado, es decir, cuando el proceso recibe una orden o señal que estaba esperando para pasar al proceso de preparado, y posteriormente, tras la transición **c**, pasar a ejecución.

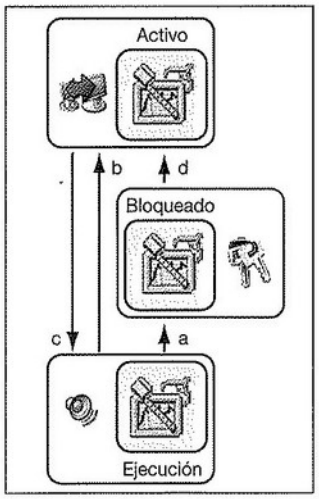


Fig. 2.14. *Transición de los procesos.*

En un sistema multiproceso o multihebra, cuando un proceso o hilo pasa de un estado a otro, por ejemplo de espera a ejecución, lo que se produce es un **cambio de contexto.** El cambio de contexto se puede producir entre diferentes procesos, diferentes hilos de un mismo proceso o entre hilos de diferentes procesos. Así, cuando se realiza un cambio de contexto entre hebras de un mismo proceso, se realizará un **cambio de contexto parcial,** ya que este cambio de contexto parcial no afectará a espacios de memoria, hardware, etc. Si el cambio de contexto se produce entre hilos de diferentes procesos, se producirá un **cambio de contexto completo** ya que el cambio afectará a memoria, hardware, ficheros comunes, etc. La Figura 2.15 muestra un ejemplo de cambio de contexto entre dos procesos.

Mediante el uso de las hebras, los sistemas operativos se hacen mucho más veloces, aunque es necesario tener cuidado a la hora de planificar la ejecución entre las mismas, ya que varias hebras

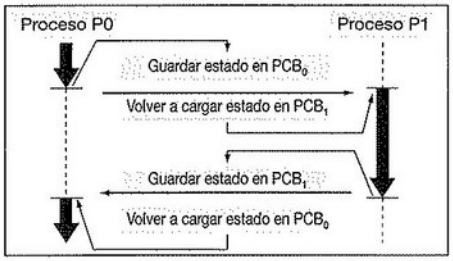


Fig. 2.15. *Cambio de contexto.*

podrán acceder a cualquier variable compartida en memoria y modificarla. En este caso, si una nueva hebra accede a una de estas variables y la modifica, cuando la hebra original vuelva a acceder a la variable, el valor de ésta habrá cambiado y la hebra no podrá seguir ejecutándose. Se pueden producir los denominados problemas de incoherencia de datos y, por consiguiente, provocar inestabilidad en el sistema.

En adelante, cuando se describan los algoritmos de planificación de procesos en la CPU del sistema, siempre se hará referencia a procesos. Hay que tener en cuenta que la gestión de las tareas o procesos se hará igual en un sistema operativo multiproceso tradicional que en un sistema operativo multihebra, excepto que, en vez de planificar procesos, se planificarán hebras de procesos.

La Figura 2.16 muestra cómo se ejecutan tres procesos (o hilos en sistemas operativos multihilo), pasando de estar activos a estar en espera según se asignan tiempos de ejecución de CPU a unos u otros. Las casillas sombreadas indican que el proceso está en ejecución.

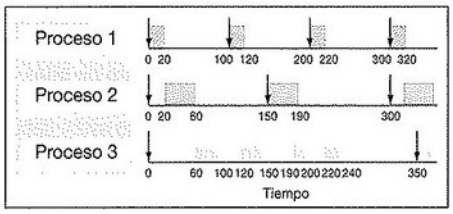


Fig. 2.16. *Esquema de ejecución de tres procesos.*

Antes de analizar y ver cómo se asignan prioridades en la ejecución de procesos o hilos, en sistemas operativos multiproceso o multihilo, se explica cuál es la terminología utilizada para la gestión de procesos. En este caso, se indica el orden con que el sistema operativo gestiona cada proceso, dependiendo de su estado.

Los diferentes estados tienen una relación directa con lo que se denominan **prioridades,** que son aquellas que el administrador del sistema asigna a cada proceso. De ello dependerá que un proceso se ejecute en más o menos tiempo. Se pueden establecer prioridades en función de la necesidad de ejecución de algunos programas. Los programas que más se ejecutan, es decir, los más necesarios, tendrán mayor prioridad de ejecución que aquellos que se ejecutan muy de vez en cuando.

La técnica de **planificación** permite indicar al ordenador los procesos que deben ejecutarse y los estados que deben adoptar. Mediante los **algoritmos de planificación** se consigue decidir, en cada momento, qué proceso debe ejecutarse. Algunas características de estos algoritmos son el equilibrio, la eficiencia y el rendimiento.

Algoritmos de planificación

El concepto de proceso tiene una connotación dinámica y va ligado a la ejecución de un programa. Durante su ejecución, un proceso compite con el resto de los procesos del sistema por el uso de los recursos. El reparto de éstos, entre los distintos procesos, y su ejecución concurrente se conoce como **multiprogramación.** Los sistemas operativos disponen de los servicios necesarios para la gestión de los procesos, tales como su creación, finalización, ejecución periódica, cambio de prioridad, etc. Además, durante su existencia, los procesos pasan por distintos estados cuyas transiciones están controladas por el sistema operativo.

Toda la información de un proceso que el sistema operativo necesita para controlarlo se mantiene en una estructura de datos: **bloque de control de procesos.** En sistemas operativos multiproceso, el sistema operativo mantiene listas de bloques de control de procesos para cada uno de los estados del sistema.

Se denomina **planificador** a la parte del sistema operativo encargada de asignar los recursos del sistema de manera que se consigan unos objetivos de comportamiento determinados. Hay tres tipos de planificadores que pueden coexistir en un sistema operativo: planificadores a largo, medio y corto plazo. En general, el planificador es el encargado de determinar qué proceso pasará al estado activo de entre todos los procesos que están en el estado «preparado».

La elección de los algoritmos de planificación se realiza teniendo en cuenta sus características frente a los criterios de diseño elegido. Las propiedades de los algoritmos se expresan en términos de aspectos tales como la eficacia en el uso del procesador, el rendimiento o número de procesos completados por unidad de medida temporal, el tiempo de espera de un proceso y el tiempo de respuesta a un evento.

Los algoritmos de planificación se diseñan dependiendo de su función y pueden ser de diferentes tipos:

- **Expropiación** *(preemption)*. El planificador de la CPU puede intervenir cuando un proceso pasa voluntariamente a estado de espera, ya que necesita realizar una operación de entrada y salida o se tiene que sincronizar con otro proceso para utilizar espacios comunes, normalmente de memoria. El planificador también puede intervenir cuando se finaliza un proceso. En este caso se dice que la planificación es **sin expropiación.** Cuando interviene el planificador cambiando el estado de un proceso de espera a preparado, se dice que la planificación es **con expropiación.**

- **Intervalos de tiempo.** El proceso puede recibir la atención de la CPU durante un cierto intervalo de tiempo.

- **Prioridades.** A los procesos se les pueden asociar prioridades, que pueden ser estáticas o dinámicas. En este tipo de algoritmos suele ser necesaria la intervención del usuario o administrador del sistema.

- **Tiempos límites** *(deadlines)*. Existe un tiempo límite para que termine un proceso. Cuanto más cerca está ese límite, más urgente se hace su planificación, y más ciclos consecutivos de CPU se le asigna.

A continuación, se incluyen algunos de los algoritmos de planificación puestos en práctica por el planificador a la hora de asignar intervalos de CPU en la ejecución de cada proceso. Recordemos que aunque en este punto se hable de procesos, servirá igualmente para hilos.

Algoritmo de operación por rondas

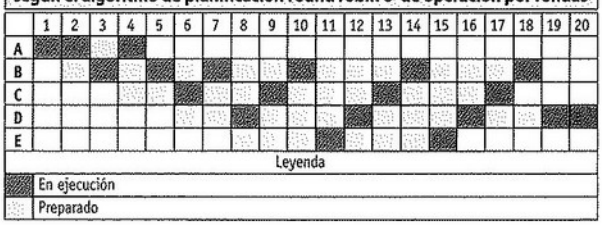
Asigna, por rondas, tiempos de ejecución a los diferentes procesos. Este algoritmo también se denomina **algoritmo de *round-robin,*** y la asignación de tiempos de ejecución a los diferentes procesos es la misma y se realiza de forma secuencial. A cada proceso se le asigna el mismo ***quantum***, es decir, el mismo intervalo de tiempo de ejecución. La selección de entre los procesos se realiza mediante una cola FIFO (*First In First Out*, El primero en entrar es el primero en salir).

Como se puede apreciar, cuando llega un proceso nuevo y hay otro en ejecución, los ciclos de CPU se distribuyen entre ellos, pero ejecutándose antes un ciclo de CPU para el proceso que está

Ejemplo de operación por rondas

Proceso	Ciclo de llegada	Ciclos totales de CPU	Ciclo inicial	Ciclo final
A	0	3	1	4
B	2	6	3	18
C	4	4	6	17
D	6	5	8	20
E	8	2	11	15

Asignación de ciclos de CPU a los procesos según el algoritmo de planificación round robin o de operación por rondas



en activo y no para el recién llegado, al que se le asignará su ciclo inmediatamente después.

Algoritmo FCFS (*First Come, First Serve*, El primero en entrar es el primero en ser servido)

Los ciclos de CPU de cada proceso se asignan en función de una cola FIFO. En este caso, al primer proceso que llega se le asignan tiempos o ciclos de CPU hasta que termina completamente. A continuación, se ejecuta completo el siguiente proceso que hay en la cola FIFO, y así sucesivamente hasta finalizar la ejecución del último proceso.

Algoritmo STR (*Short Time Remainder*, Resto de tiempo breve)

Este algoritmo permite asignar el tiempo de ejecución de forma prioritaria a procesos muy cortos para ejecutarlos en el menor tiempo posible. Si está ejecutando un proceso y llega otro, independientemente de la duración del nuevo, el proceso que está en ejecución finalizará. Una vez finalizado, el siguiente proceso a consumir ciclos de CPU será el más corto de los que haya en la cola de espera.

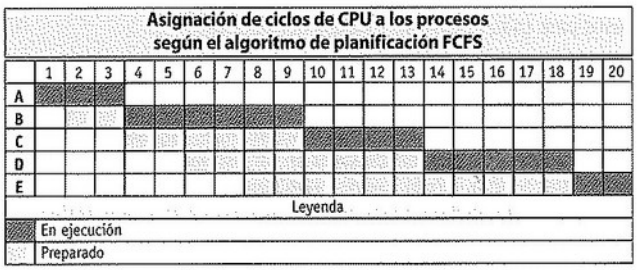
Algoritmo SRTF (*Shortest Remaining Time First*, Primero el de tiempo restante más breve)

Es una variedad del STR, pero en este caso la asignación de ciclos de CPU se hace en función del proceso al que le queden menos ciclos para terminar. De esta forma, cuando llega un proceso nuevo, se estiman los ciclos que le quedan tanto al proceso que hay en ejecución como al que ha llegado. De los que hay en ese momento en la cola de procesos a ejecutar, se terminará aquel al que le queden menos ciclos para su finalización, y así sucesivamente hasta terminar con todos los procesos planificados.

Existen otros muchos algoritmos de planificación, pero los mencionados en estas líneas son los más importantes. De ellos depende, en gran medida, la eficacia del sistema informático, y esta labor la tiene que realizar el administrador del sistema, con los recursos que le ofrezca el sistema operativo.

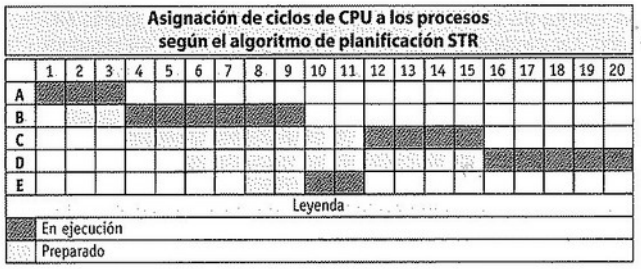
Ejemplo FCFS

Proceso	Ciclo de llegada	Ciclostotales de CPU	Ciclo inicial	Ciclo final
A	0	3	1	3
B	2	6	4	9
C	4	4	10	13
D	6	5	14	18
E	8	2	19	20



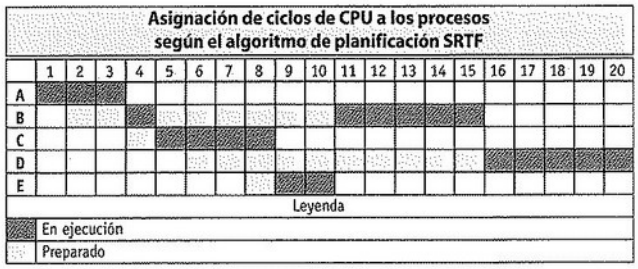
Ejemplo STR

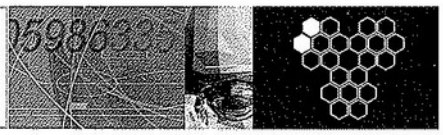
Proceso	Ciclo de llegada	Ciclos totales de CPU	Ciclo inicial	Ciclo final
A	0	3	1	3
B	2	6	4	9
C	4	4	12	15
D	6	5	16	20
E	8	2	10	11



Ejemplo SRTF

Proceso	Ciclo de llegada	Ciclos totales de CPU	Ciclo inicial	Ciclo final
A	0	3	1	3
B	2	6	4	15
C	4	4	5	8
D	6	5	16	20
E	8	2	9	10





F. Sincronización y bloqueo de procesos

Los **procesos de sincronización** consiguen que los programas se ejecuten en el orden adecuado y sin interferencias entre ellos.

Suele ocurrir en multiprogramación, que varios programas necesiten utilizar información de las mismas variables de memoria. Esta fase de ejecución del programa es lo que denominamos **fase crítica**, y es aquella en la que el sistema operativo tiene que sincronizar los procesos para que no puedan acceder a los mismos datos ni modificarlos de forma indiscriminada.

Un ejemplo puede ser un programa que accede a una variable de memoria y modifica su contenido.

Supongamos que ese programa necesitara ese valor posteriormente. Si, entre tanto, la CPU asigna su tiempo de ejecución a un proceso nuevo, y éste necesita tomar el valor de esa misma variable para modificarlo, cuando el primer proceso vuelva a ejecución no encontrará el valor deseado, y se producirán conflictos y errores en la ejecución de los procesos. Mediante la **sincronización implícita**, el sistema operativo solucionará tales problemas. Esto suele ocurrir en sistemas operativos gestores de redes: UNIX, OS/2, Windows NT Server, Windows 2000, etcétera.

También existe la denominada **sincronización externa**, que se realiza mediante software. Se utilizan para ello unos programas, denominados **monitores**, que complementan la sincronización implícita, de forma que la asignación de los recursos es exclusiva para cada proceso.

El hecho de que dos o más procesos se ejecuten de forma concurrente, implica que cada uno de ellos estará en un estado diferente. La sincronización implica que dos o más procesos concurrentes no podrán utilizar los mismos recursos en el mismo instante. Por ello, algunos procesos estarán en ejecución mientras otros quedarán bloqueados.

El bloqueo de los procesos tiene lugar cuando dos o más procesos necesitan utilizar el mismo recurso del sistema. Supongamos que un proceso A necesita grabar información en el disco duro. Llega a ejecución un segundo proceso B que también tiene que utilizar el mismo disco duro para guardar la información que está procesando. En este caso, el primer proceso tendrá acceso al recurso en cuestión: el disco duro. Por el contrario, el segundo proceso estará bloqueado, ya que el recurso hardware está a disposición del proceso anterior.

El bloqueo se utiliza para que no existan conflictos en el uso de componentes hardware como, por ejemplo, memoria, procesador, dispositivos de almacenamiento, etcétera. Es evidente que los bloqueos en los procesos solamente se pueden dar en sistemas operativos multitarea o multihilo, y no en los sistemas operativos monotarea.

El sistema operativo puede utilizar diferentes técnicas para realizar la gestión de bloque de los procesos:

- Señales.
- Tuberías o *pipes*.
- Semáforos.
- Variables condicionales.
- Paso de mensajes.

En cualquier caso, las operaciones de sincronización y bloqueo entre procesos deben ser automáticas.

Los semáforos se utilizan cuando varios procesos utilizan algún recurso o variable compartida. El espacio de memoria en el que están estas variables compartidas se denomina **región crítica**, y es función elemental y básica del sistema operativo proteger los procesos para que ningún otro pueda acceder a esa región crítica.

Es necesario solucionar los problemas que se puedan derivar de la utilización de la región crítica, bien sea mediante los semáforos o mediante cualquier otra técnica de sincronización. La solución al problema de la región crítica tiene que satisfacer tres requisitos.

- **Exclusión mutua.** Si un proceso se está ejecutando en su región crítica, ningún otro proceso se puede estar ejecutando en esa misma región.
- **Progreso.** La asignación de una región crítica a un proceso se puede posponer hasta que se tengan las garantías necesarias.
- **Espera acotada.** Los procesos tienen una cota en cuanto a las veces que pueden acceder a esa región crítica.

Como lo normal es que las tareas de un sistema no sean independientes, es necesario que a veces los procesos cooperen entre sí. Para ello, y gracias a la sincronización, se establecen restricciones en el orden de ejecución de las acciones de cada proceso. De esta forma, cuando un proceso necesita datos generados por otro proceso, se produce un bloqueo del primero para dar prioridad al que acaba de iniciar su ejecución. Estos bloqueos son automáticos.

En resumen, los diferentes procesos que están cargados en un sistema, cooperan, necesitan espacios comunes de memoria, recursos hardware, etc. Así, es necesario que el sistema operativo esté diseñado de tal forma que permita que cada proceso se ejecute en el orden adecuado, en el espacio de memoria adecuado y sin alterar a los otros procesos. Para ello, se utiliza la sincronización y el bloqueo de procesos. El sistema operativo tiene que estar diseñado para detener o iniciar un proceso utilizando las técnicas de bloqueo específicas, como los semáforos.

2.6 Gestión de memoria

La parte del sistema operativo que administra la memoria es el **administrador de memoria.** Se encarga de llevar un registro de las zonas de memoria que se están utilizando. De esta forma, reservará espacio de memoria para los procesos nuevos y liberará el espacio de los procesos que han finalizado.

También se encarga de gestionar el intercambio de datos entre memoria y disco, siempre y cuando los procesos sean tan grandes que no quepan de una sola vez en memoria.

Los sistemas de administración de memoria se pueden clasificar en dos grupos:

- Los que desplazan los procesos de memoria central al disco, y viceversa.
- Los que no realizan dicho desplazamiento.

La gestión de memoria es importante cuando se utilizan sistemas operativos multiproceso y, aún más, con sistemas operativos multihilo, ya que se comparten espacios de memoria en donde están las variables compartidas y a las que acceden varios procesos o los hilos de un mismo proceso. En este caso, el sistema operativo debe controlar y gestionar la memoria de forma que cada proceso utilice un espacio de memoria, sin afectar a otros espacios de memoria, en los que puede haber datos o registros con información para otros procesos o a los hilos de un proceso.

En general, la gestión de memoria es sencilla en sistemas operativos monoproceso. Al introducir la multitarea, la cosa se complica, ya que es necesario disponer de varios procesos residentes simultáneamente en memoria.

La primera opción consiste en dividir la memoria en particiones fijas; el sistema operativo dispone de una cola de los procesos que solicitan entrar en memoria. El planificador tiene en cuenta los requerimientos de memoria de cada uno de los procesos y las particiones de memoria disponibles. Otra organización posible consiste en que cada partición tenga asociada una cola de tareas. El concepto de **intercambio** se encuentra vinculado con la multitarea, dado que los procesos en espera pueden ser llevados al disco y dejar libre la parte de memoria que ocupan para que otros procesos comiencen su ejecución. Los procesos se pueden cargar siempre en la misma posición de memoria o en otra distinta. Éste es el concepto de la **reubicación,** que puede ser estática o dinámica.

La mayor dificultad en el diseño con las particiones fijas es la adecuada selección de los tamaños de las mismas, puesto que puede derivar en un desaprovechamiento o fragmentación de la memoria. Esta fragmentación puede ser interna, que consiste en aquella parte de la memoria que no se está usando, o externa, que tiene lugar cuando una partición disponible no se utiliza porque es muy pequeña para cualquiera de los procesos que esperan.

Con un conjunto dinámico de procesos ejecutándose, no es posible encontrar las particiones de memoria adecuadas, por lo que la otra posibilidad consiste en utilizar particiones variables. El problema que se plantea ahora es disponer de un registro de las particiones libres y ocupadas que sea eficiente tanto en el tiempo de asignación como en el aprovechamiento de la memoria, aunque sigue habiendo problemas de fragmentación externa. Una solución radica en permitir que los procesos puedan utilizar memoria no contigua, lo que se consigue mediante técnicas de paginación. En esta situación, hay un mecanismo de traducción de las direcciones lógicas a las físicas mediante una tabla de páginas. La tabla de páginas presenta dos cuestiones a tener en cuenta: el tamaño de la tabla (que puede ser demasiado grande) y el tiempo de asignación (que debe ser de corta duración).

En contraposición a la visión de la memoria como una matriz o lista unidimensional, está la concepción, por parte del usuario, de la memoria como un conjunto de segmentos de diferentes tamaños, sin ninguna ordenación entre ellos. Este esquema corresponde a la técnica denominada segmentación. En este caso, el espacio de direcciones lógicas es un conjunto de segmentos, con diferentes nombres y tamaños. En el esquema de segmentación no se produce fragmentación interna, pero sí externa, que tiene lugar cuando todos los bloques de memoria libre son muy pequeños para albergar un bloque de proceso.

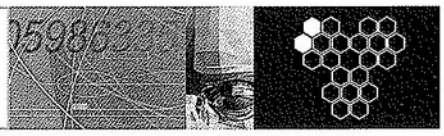
Aunque la segmentación y la paginación son esquemas diferentes de gestión de la memoria, se pueden considerar estrategias combinadas, ya que la única diferencia es que la paginación utiliza bloques de memoria de tamaño fijo. En todos estos esquemas se supone que el proceso que se va a ejecutar está cargado totalmente en memoria. La idea de permitir ejecutar procesos que no están cargados totalmente en memoria, e incluso que su tamaño superen al de la memoria física instalada, da lugar al concepto de *memoria virtual.*

A continuación, se describen las técnicas más usuales aplicadas por los sistemas operativos para la gestión de la memoria.

A. Memoria virtual

El ordenador dispone de memoria central o principal, pero ésta es limitada y, en grandes sistemas, casi siempre insuficiente.

Al principio, para solucionar este problema, se adoptaron técnicas tales como dividir el programa en partes denominadas **capas.** Cada una de las capas se iría ejecutando según fuera necesario; es decir, en primer lugar, se pasaría del disco duro (o soporte de



almacenamiento) a memoria. Cuando fuera necesario utilizar otra parte del programa que no estuviera en memoria central o principal (memoria RAM), se accedería de nuevo al disco para cargar la siguiente capa en memoria principal. Esta labor de dividir el programa en capas la puede realizar el mismo programador, dividiendo el programa en módulos que se irán ejecutando según sea necesario, si bien esto supone un elevado esfuerzo para él.

Con el perfeccionamiento de los sistemas operativos, se ha llegado a la ubicación de las capas del programa en memoria de forma transparente para el programador y para el usuario. De este modo, sólo el diseñador del sistema operativo es el encargado de realizar esta carga y descarga de capas en la memoria.

El método diseñado por Fotheringan se conoce con el nombre de **memoria virtual.** El diseñador consideró que el programa que se iba a ubicar en memoria podría ser excesivamente grande para el tamaño físico de ésta; por tanto, permanecería en memoria la parte del programa que se está ejecutando, mientras el resto seguiría ubicada en el disco. Esta técnica, aplicada hoy en día en la mayoría de los sistemas operativos, considera el espacio en disco libre como si se tratase de memoria RAM (memoria virtual). Así, para el usuario, el programa estará realmente cargado en memoria RAM. Pero sólo se carga en la memoria RAM la parte del programa que en realidad se está ejecutando en ese instante. Entre tanto, el resto del programa en ejecución permanece, temporalmente, almacenado en disco para su posterior utilización, si fuera necesario.

Si, en un momento dado, es necesario ejecutar una parte del programa que está almacenada en memoria virtual (en el disco duro), pasará a memoria RAM para su ejecución, y la parte del programa que estaba en memoria RAM se almacenará en el disco duro. Así, siempre se dispondrá de más memoria RAM liberada para realizar cálculos o ejecutar otros programas, sobre todo en sistemas operativos multiusuario y multitarea. Asimismo, en sistemas operativos multitarea puede utilizarse la técnica de memoria virtual, de forma que de una memoria de 32 Mb pueda asignarse 1 Mb a cada programa. Así, cada programa se aloja en su parte de la memoria, independientemente del tamaño global que tenga.

Con esta técnica, se consigue disponer, casi siempre, de RAM libre, necesaria para el propio procesador. Por el contrario, cuando se cargan demasiados procesos a la vez, el sistema se ralentiza, ya que tiene que estar pasando información continuamente desde el disco duro a la memoria RAM o viceversa.

Los sistemas operativos multiusuario y multitarea son especialistas en esta gestión. Microsoft Windows, en casi todas sus versiones, y especialmente en NT, 2000 y 2003, realiza una gestión muy eficaz de la memoria virtual.

Es evidente que para realizar esta gestión es necesario disponer de un espacio determinado en el disco duro. Concretamente, para sistemas de Microsoft es recomendable asignar un valor comprendido entre 2,5 y 5 veces del tamaño total de la memoria RAM de espacio en disco para la gestión de memoria virtual.

B. Intercambio de memoria *(swapping)*

El *intercambio de memoria* es una técnica parecida a la de memoria virtual. Cuando varios usuarios están ejecutando procesos en un mismo ordenador, éste se ve obligado a cargarlos en memoria RAM. Según el estado en el que se encuentre el proceso de casa usuario, la memoria se irá liberando de su proceso y pasará a la zona de **intercambio** mediante la técnica llamada **intercambio hacia fuera.** De esta forma, la memoria interna queda liberada para que en ella se pueda almacenar otro proceso del mismo usuario o de otro.

Si el usuario vuelve a solicitar su proceso para seguir ejecutándolo, se produce el denominado **intercambio hacia dentro,** que consiste en pasar el programa de la zona de *intercambio* a la memoria interna.

Esta zona de *intercambio* se suele utilizar en sistemas operativos como UNIX y Linux. Está formada por un espacio físico del disco donde se encuentra instalado el sistema operativo y las aplicaciones que se van a ejecutar. Los fabricantes de estos sistemas operativos recomiendan que esta zona sea del 20 %, aproximadamente, del espacio en disco o el doble de la capacidad de memoria RAM del ordenador (la mayor de las dos).

La diferencia entre la gestión de memoria virtual y el *intercambio de memoria* radica en que mediante la primera puede llegar a ocurrir que el disco duro esté tan lleno que la gestión sea difícil o imposible, ya que el espacio destinado al intercambio suele ser espacio del disco duro en el que está instalado tanto el sistema operativo como el software de aplicaciones y los datos del usuario. En el *intercambio de memoria* no puede ocurrir esto, ya que esta zona siempre estará disponible para el intercambio de programas con la memoria principal. Normalmente, al estar esta zona en un dispositivo físico diferente, todo el espacio estará disponible cada vez que se encienda el ordenador.

C. Paginación

La paginación es una técnica que consiste en dividir la memoria interna o RAM en zonas iguales, llamadas ***marcos,*** y los programas en partes del mismo tamaño, denominadas **páginas.**

Para ubicar un programa en memoria, el sistema operativo buscará en memoria física los *marcos* que tenga libres. El tamaño de estos *marcos* se diseña mediante hardware.

Si utilizamos un sistema operativo multiprogramación y sólo hay una tarea en ejecución, éste tendrá asignados todos los *marcos* necesarios para él. Esta asignación de *marcos* la realiza el sistema operativo.

Mediante la **tabla de páginas,** la CPU asigna las direcciones físicas de los *marcos* a las páginas en las que se ha dividido el programa. La asignación de los *marcos* no tiene que ser necesariamente consecutiva. Un proceso se puede ubicar en memoria interna en

marcos no contiguos, ya que éstos pueden estar ocupados por otros procesos.

La técnica de la paginación es similar a la de la memoria virtual. La gran diferencia es que aquí no existe disco duro para intercambiar parte de los procesos. Concretamente, el sistema operativo DOS utiliza una técnica parecida a la paginación. Como ejemplo, consideremos el sistema operativo DOS. Solamente puede ejecutar los programas en la memoria convencional (los primeros 640 Kb). El resto de memoria sólo sirve de almacenamiento para parte del núcleo del sistema operativo, así como para almacenar temporalmente parte de los procesos que tengan un tamaño superior a 640 Kb.

El sistema operativo DOS divide la memoria extendida (por encima del primer Mb) en páginas de 64 Kb de tamaño, para realizar el intercambio de información con la memoria convencional.

Un programa que ocupe 1 Mb albergará lo que pueda de memoria convencional, y el resto se almacenará temporalmente en memoria extendida. Este programa se paginará a través del llamado **marco de página,** que se describe posteriormente. Se intercambian las páginas desde memoria convencional a la memoria extendida y viceversa, dependiendo de la parte del proceso que se vaya a ejecutar. Esta gestión de memoria se conoce como **memoria expandida.**

En resumen, la paginación es una técnica de reasignación o redireccionamiento dinámico; se ha de tener en cuenta que la tabla de páginas se puede almacenar en registros especiales destinados a tal efecto o en una parte de la propia memoria.

La transformación de las direcciones lógicas en físicas la realiza la **unidad de administración de memoria** (MMU, *Management Memory Unit*).

D. Segmentación

Es una técnica similar a la paginación, que permite definir los bloques de memoria de tamaño variable. Cada **segmento** puede variar desde 0 hasta un máximo permitido. Estos segmentos pueden tener distinta longitud. Además, la longitud de un segmento puede variar según las necesidades el programa.

Supongamos que realizamos un programa y, para que se ejecute, necesita utilizar tablas (estructuras de datos) en memoria. Si tenemos en cuenta que una tabla puede asignarse de forma estática o dinámica, según las necesidades del programa, habrá veces que esta tabla necesitará un espacio en memoria determinado, y otras veces más espacio, o menos. Mediante la segmentación, podemos ubicar en memoria estas estructuras de datos, independientemente del tamaño que tengan.

El ordenador, a través del sistema operativo, puede organizar la memoria en bloques concretos, y tener partes de ella destinadas a almacenar las estructuras de datos, que pueden aumentar o disminuir según las necesidades del usuario o del programa. Para ello, se utilizarán las **pilas** de memoria, en las que se gestionan las estructuras de datos necesarias.

La paginación difiere de la segmentación en que mientras que las páginas son de tamaño fijo, los segmentos no.

El uso de la técnica de paginación o segmentación dependerá del sistema operativo utilizado y de la máquina en la que se emplee, así como de las necesidades del software.

E. Programas reubicables, reentrantes, residentes y reutilizables

Los programas o procesos reubicables

Son aquellos que, una vez cargados en memoria RAM para ejecutarse, pueden variar de situación de manera que la parte de memoria RAM que ocupa sea necesaria para ubicar otro proceso. Estos procesos o programas cambian de posición cuando se está realizando una operación sobre el ordenador. Esta operación suele ser de configuración interna del propio ordenador.

Los programas o procesos reentrantes

Son aquellos programas que, si no se están ejecutando, dejan la memoria libre para otros procesos. Estos procesos, cuando se liberan, se suelen almacenar temporalmente en el disco duro: son los gestionados mediante la técnica de memoria virtual.

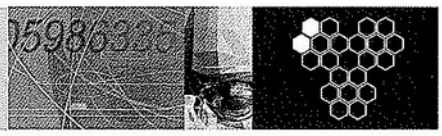
Los programas o procesos residentes

Son aquellos que, una vez cargados en memoria, permanecerán en ella hasta que se apague el ordenador. No cambian su ubicación en ningún momento y pueden situarse entre los 1024 Kb y los 1088 Kb de memoria RAM. Suelen ser programas de antivirus, de análisis del sistema, de supervisión, etc. Los más comunes son los llamados **centinelas,** que incorporan los antivirus para que analicen continuamente lo que se carga en memoria. De esta forma, si se ejecuta un proceso, el programa residente lo analiza y, si detecta algo extraño, generará un mensaje de alerta.

La ubicación de estos programas en memoria dependerá, fundamentalmente, del sistema operativo y de la propia aplicación que inicie el programa residente. Suelen ubicarse en esos 64 Kb de memoria, aunque no necesariamente.

Los programas o procesos reutilizables

Los programas o procesos **reutilizables** son los utilizados por varios usuarios a la vez en memoria, independientemente del



número de usuarios que lo vayan a utilizar. Con ello se consigue un mejor aprovechamiento de la memoria.

F. Protección de memoria

La protección de memoria es una tarea que el sistema operativo tiene que realizar para que en ningún momento se **solapen** unos procesos con otros en la memoria física del ordenador.

Esta protección se realiza normalmente por hardware mediante los **registros frontera** o **dirección frontera.** También puede realizarse por software, si bien este control será bastante más lento que si se utilizarán técnicas de hardware.

Unos sistemas dividen la memoria en bloques de 2 Kb y asignan un código de protección de 4 bits a cada bloque de memoria.

La ubicación de cada parte del software en memoria tiene que responder a la misma combinación de bits que los almacenados en la propia memoria. Si esto es así, el programa, o parte de él, puede almacenarse en ese bloque físico de memoria.

Otra alternativa para la protección de memoria consiste en dotar a la máquina de dos registros especiales de hardware, llamados **registro base** y **registro límite.** Mediante la información contenida en estos registros, pueden cargarse en memoria las instrucciones correspondientes a los programas, sin miedo a que ocupen bloques físicos en los que se encuentran almacenadas otras instrucciones.

El problema de este sistema de protección es su poca flexibilidad, dado que es necesario definir el tamaño de los bloques de memoria en los que ubicar programas.

Si el programa es de tamaño menor o igual que las particiones realizadas, podrá ejecutarse. Si, por el contrario, el programa es de tamaño superior a las participaciones o bloques, no podrá ejecutarse hasta que se reasigne el tamaño adecuado de las particiones.

2.7 Gestión de periféricos

Una de las funciones principales de un sistema operativo es el control de los periféricos de entrada y salida del ordenador. El sistema operativo se encarga de enviar órdenes, determinar el dispositivo que necesita la atención del procesador, eliminar posibles errores, etcétera.

En primer lugar, es necesario clasificar los periféricos en función de si gestionan la información por bloques o por caracteres. La clasificación es la siguiente:

- **Periféricos tipo bloque.** Son aquellos en los que la información que se maneja es de tamaño fijo. La información se escribe o se lee de la memoria en forma de bloque. Un ejemplo son los registros de ficheros de datos almacenados en discos o disquetes, ya que cada registro contiene información referente a un bloque homogéneo.

- **Periféricos tipo carácter.** Son los que sirven para introducir datos en forma de caracteres, sin ningún orden concreto, dentro de la memoria del ordenador, por ejemplo, desde el teclado. También analizaremos la gestión que se realiza de los periféricos que sirven para mostrar los resultados obtenidos de la gestión en forma de cadena de caracteres, como por ejemplo el monitor o la impresora.

Cada periférico consta de componentes mecánicos y electrónicos. Por ejemplo, un disco duro estará compuesto por los propios discos de aluminio recubiertos de material magnético, las cabezas de lectura, el motor que los hace girar, etc., y por la denominada controladora o adaptador, encargado de conectar el dispositivo físico al ordenador.

El sistema operativo se encarga de acceder a la información de la memoria principal, extraerla en forma de impulsos eléctricos y enviarla a los diferentes dispositivos periféricos. Si la información se envía a un disco duro, los impulsos se transformarán en señales de tipo magnético; si se envía a una impresora, se transformará en caracteres, etcétra.

Los dispositivos físicos que el sistema operativo tiene que gestionar para que la información pase de un sitio al otro del ordenador se clasifican según la función que realizan:

- **Soportes de almacenamiento.** Memoria auxiliar del ordenador o memoria externa. Pueden ser discos duros, disquetes, CD-ROM, DVD, *streamer,* cintas DAT, etcétera.

- **Interfaces.** Permiten la comunicación entre el usuario y el sistema operativo. Son el monitor, el teclado, el ratón, la impresora, etcétera.

- **Soportes de transmisión.** Buses y canales encargados de transmitir la información entre los diferentes componentes que integran el ordenador.

Hay que destacar las interfaces como medio de comunicación entre hardware y software a través del sistema operativo. Las interfaces se pueden clasificar en:

- **Interfaz tipo texto.** Si el sistema operativo es de tipo texto, todas las órdenes que el usuario introduzca y las respuestas que el sistema operativo proporcione se introducirán o mos-

trarán, respectivamente, mediante cadenas de caracteres. Un ejemplo de sistemas operativos tipo texto son DOS, UNIX (en versiones inferiores a System V Reléase 4), las primeras versiones de Linux, etc. Todas las órdenes se introducen por teclado y se muestran en la pantalla. La pantalla, cuando se gestiona en tipo texto, tiene un tamaño de 80 columnas por 24 filas, es decir, puede mostrar hasta 1920 caracteres de una sola vez.

- **Interfaz tipo gráfico.** Hoy en día, la mayoría de los sistemas operativos utilizan interfaces de comunicación entre máquina y ordenador de tipo gráfico.

  En este tipo de interfaces, el uso del ratón es casi imprescindible. La información en pantalla se muestra en bloques o en pantallas independientes. A estos bloques se les denomina **ventanas,** y en ellas aparecen una serie de componentes y objetos que sirven para enviar o recibir información sin necesidad de escribir nada.

Para la gestión de periféricos hay que hablar de: *controladoras, canales, interrupciones*. Estos conceptos se describen en la Unidad siguiente.

La **controladora** es un componente hardware que sirve para gestionar el uso de periféricos, puede ser de varios tipos y su función consiste en conectar físicamente el periférico a la placa base del ordenador para establecer la comunicación. En la Unidad siguiente, dedicada al hardware, se describen detenidamente los tipos de controladoras, atendiendo al tipo de periférico, la velocidad, la capacidad, etcétera.

Las controladoras o adaptadores necesitan un pequeño software para que exista comunicación entre el periférico y el microprocesador. Este software, llamado **controlador** (o ***driver***), se encarga de realizar funciones de traducción entre el periférico y el ordenador para que ambos se entiendan. Los controladores suelen suministrarlos los fabricantes de periféricos en disquetes o CD-ROM y suelen estar diseñados para varios sistemas operativos; así, el mismo periférico se puede utilizar en un sistema operativo Windows o en uno UNIX, dependiendo del controlador que se instale. Los buses o líneas, en otros casos llamados **canales,** se describieron en la Unidad anterior. Respecto a los soportes de almacenamiento que se explican en la Unidad 3, se describirá cómo configurarlos e instalarlos y los requisitos hardware y software que necesitan para que el sistema operativo los pueda gestionar.

2.8 Gestión de datos. Sistema de archivos

Cuando trabajamos con sistemas operativos multiusuario, la gestión de datos que se hace dentro del ordenador y su ubicación en la memoria y en los soportes de almacenamiento externo pueden plantear algunos problemas.

Ya hemos visto cómo el sistema operativo pone medios para asignarles una ubicación en la memoria. En cuanto al almacenamiento en soportes externos, la gestión que hace el sistema operativo responde a varias características:

- Que se pueda almacenar una gran cantidad de información.
- Que se almacene de forma correcta una vez terminado el procesamiento.
- Que sea posible que varios procesos o programas accedan a la misma información sin interferencias.

Como ya se ha visto, después de ser procesada la información, tiene que almacenarse de forma permanente en los soportes externos. Cada sistema operativo utiliza su propio **sistema de archivos.** Cada uno de ellos hace una gestión diferente del espacio de almacenamiento, lo cual dependerá de si el sistema es multiusuario o monousuario, multitarea o monotarea, multiproceso o monoproceso, etcétera.

El sistema operativo gestiona cada archivo almacenado en el soporte indicando: el nombre, el tamaño, el tipo, la fecha y hora de grabación, el lugar del soporte en el que se encuentra, etcétera.

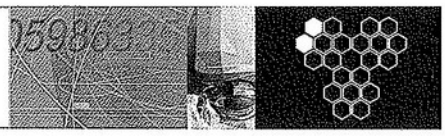
A. Nombre de los archivos

Las características de los nombres de los archivos dependen del sistema operativo.

El sistema operativo DOS y otros muchos sólo administran nombres de ocho caracteres como máximo; UNIX, de más de once; Windows, de hasta 256. Unos diferencian entre mayúsculas y minúsculas (UNIX), y otros no lo hacen (DOS). Algunos sistemas, además del nombre, incluyen una extensión de tres caracteres separados por un punto.

Estas extensiones sirven para que el sistema pueda diferenciar rápidamente el tipo de archivo de que se trata. La omisión de la misma puede provocar que el sistema no reconozca el archivo. Las más usuales en los sistemas operativos actuales son las siguientes:

.TXT, archivos de texto sin formato.
.BAS, archivos de Visual Basic.
.BIN, archivos binarios.
.DOC, archivos de Microsoft Word.
.BMP, archivos de mapa de bits (gráficos).
.JPG, archivos gráficos.
.SYS, archivos de sistema.
.DLL, bibliotecas del sistema.
.OBJ, archivos objeto (de compilación).
.EXE, ficheros ejecutables (aplicaciones).
.COM, ficheros ejecutables (del sistema).
.BAT, ficheros de proceso por lotes.



B. Tipos de archivos

Los archivos que gestiona un sistema operativo se clasifican en dos grandes bloques:

- **Archivos regulares o estándares.** Son los que contienen información del usuario, programas, documentos, texto, gráficos, etcétera.
- **Directorios.** Son archivos que contienen referencias a otros archivos o a otros directorios. Este tipo de archivos se utiliza, únicamente, para albergar estructuras de archivos con el fin de diferenciarlos de otros. Todos los sistemas operativos utilizan la estructura jerárquica para almacenar sus archivos. Por ello se crean bloques (directorios) o compartimentos especiales para tener todos los archivos bien clasificados: directorios para archivos de sistema, directorios para archivos gráficos, etcétera. En casi todos los sistemas operativos existe un directorio principal, llamado **raíz,** del que depende el resto de directorios o subdirectorios y la totalidad de los archivos, si bien hay excepciones, como el sistema operativo OS/400 de IBM, que no dispone de él.
- **Archivos especiales.** Se utilizan para gestionar la entrada y salida de archivos desde o hacia los periféricos. Son los que hemos llamado *controladores*.

C. Acceso a los archivos

El acceso a los archivos es la forma en la que se puede disponer de la información almacenada en ellos. Estará condicionado por el tipo de soporte en el que estén almacenados. Si un archivo está almacenado en un *soporte secuencial* (como cintas de vídeo), el acceso a él solamente se podrá realizar de forma **secuencial.** Así, para acceder a una información determinada, será necesario pasar previamente por la anterior. Por el contrario, si el archivo está almacenado en un *soporte de acceso directo*, el acceso a sus datos también se podrá realizar en forma **directa,** con el consiguiente ahorro de tiempo. Es como si de un CD-ROM de música se tratara: para acceder a una canción determinada no es necesario escuchar ni pasar previamente por las anteriores; se alcanza de forma directa.

D. Atributos de los archivos

Los atributos de los archivos constituyen la información adicional, además de la que ya contienen, con la que cada archivo queda caracterizado. Además, quedan identificadas las operaciones que se pueden realizar sobre él. Los atributos indican cuestiones tales como nombre, hora de creación, fecha de creación, longitud, protección, contraseña para acceder a él, fecha de actualización, etcétera. De ellos, los más importantes, son los que indican qué tipo de operaciones o qué tipo de usuarios pueden usar los archivos. Dependiendo del tipo de sistema operativo utilizado, los atributos de protección son de mayor o menor importancia. Ya veremos cómo cada sistema operativo tiene sus particularidades en este aspecto.

Los atributos de protección pueden ser, de forma genérica, los siguientes:

- **Sólo lectura.** El archivo se puede leer, pero no se puede modificar.
- **Oculto.** El archivo existe, pero no se puede ver.
- **Modificado.** Si el archivo es susceptible de modificarse o no.
- **Sistema.** Si es un archivo de usuario o propio del sistema operativo.

Los atributos de protección dependerán del tipo de operación que se pueda realizar sobre ellos. Las operaciones que se pueden realizar sobre un archivo son varias: crear, eliminar, abrir, cerrar, leer, escribir, agregar, buscar, renombrar, etcétera.

E. Sistemas de archivos

Los sistemas de archivos varían de un sistema operativo a otro. Uno de los sistemas de archivos más entendidos, diseñado por Microsoft, es el sistema **FAT** (*File Allocation Table*, Tabla de asignación de archivos), que se explica en las unidades correspondientes a los sistemas operativos de este fabricante.

El sistema de archivos FAT funciona como el índice de un libro; en la FAT se almacena información sobre dónde comienza el archivo, es decir, en qué posición del disco está la primera parte de éste, y cuánto espacio ocupa, entre otras cosas. Este sistema de archivos ha evolucionado a medida que lo han hecho las versiones de sistemas operativos como DOS y Windows. Actualmente, se utilizan dos tipos de sistemas FAT: FAT 16 y FAT 32. En una Unidad posterior se explica en qué se diferencian.

Otro sistema de archivos importante es el **NTFS** (*New Technology File System*, Sistema de archivos de tecnología nueva), utilizado por Windows NT, Windows 2000, Windows XP y Windows Server 2003. Realiza una gestión diferente de los archivos que el sistema FAT; es más seguro y aprovecha mejor el espacio en disco, es más rápido y de mayor calidad.

El sistema operativo OS/2 utiliza el sistema de archivos **HTFS;** UNIX utiliza **S5;** Linux usa **EXT2.** Los estándares de sistemas de archivos de los diferentes sistemas operativos son UFS y VFS. Son más complejos, pero más seguros y fiables.

Las unidades de CD-ROM utilizan también sistemas de archivos específicos, como, por ejemplo, **CDFS,** que se utiliza exclusivamente para almacenar información en los soportes de tipo óptico.

Estos sistemas de archivos no siempre son compatibles entre sí. Un archivo UNIX puede ver un sistema de archivos FAT, pero no a la inversa. Windows NT, en cualquiera de sus versiones, no es compatible con el sistema FAT 32, y sí lo es con FAT 16.