Introducción
a los
SISTEMAS OPERATIVOS
(MS/DOS, UNIX, OS/2, MVS, VMS, OS/400)

EDUARDO ALCALDE LANCHARRO
Ingeniero Técnico del ICAI
Profesor de Informática en IFP
de Alcobendas. Madrid

JUAN MORERA PASCUAL
Ingeniero de Telecomunicación
Profesor en EU de Informática
Universidad Pontificia Comillas. Madrid.

JUAN A. PEREZ-CAMPANERO ATANASIO
Ingeniero de Telecomunicación
Profesor en E U de Informática
Universidad Pontificia Comillas. Madrid.

McGRAW-HILL

MÉXICO • BUENOS AIRES • CARACAS • GUATEMALA • LISBOA • MADRID • NUEVA YORK
PANAMÁ • SAN JUAN • SANTAFÉ DE BOGOTÁ • SANTIAGO • SÃO PAULO
AUCKLAND • HAMBURGO • LONDRES • MILÁN • MONTREAL • NUEVA DELHI • PARÍS
SAN FRANCISCO • SINGAPUR • ST. LOUIS • SIDNEY • TOKIO • TORONTO

Introducción a los SISTEMAS OPERATIVOS (MS/DOS, UNIX, OS/2, MVS, VMS, OS/400)

No está permitida la reproducción total o parcial de este libro, ni su tratamiento informático, ni la transmisión de ninguna forma o por cualquier medio, ya sea electrónico, mecánico, por fotocopia, por registro u otros métodos, sin el permiso previo y por escrito de los titulares del Copyright.

DERECHOS RESERVADOS © 1992, respecto a la primera edición en español por

McGRAW-HILL/INTERAMERICANA DE ESPAÑA, S. A.

Edificio Oasis-A - 1.ª planta
Basauri, s/n.
28023 ARAVACA (Madrid)

ISBN: 84-7615-767-3
Depósito legal: M. 35.694-1991

Diseño cubierta: Juan García
Ilustraciones: Pudimar
Fotocompuesto en: Pérez Díaz, S. A.

3456789012 LI-94 9087643215

Impreso en Mexico Printed in Mexico

Esta obra se terminó de
imprimir en Diciembre de 1995 en
Litográfica Ingramex
Centeno Núm 162-1
Col. Granjas Esmeralda
Delegación Iztapalapa
09810 México, D.F.

Se tiraron 500 ejemplares

Contenido

Prólogo

En el mundo de la informática actual existe una gran diversidad de disciplinas de las cuales unas son fundamentales y deben ser conocidas y en ocasiones dominadas por aquellas personas que han hecho de este mundo su profesión o afición.

En este sentido, los sistemas operativos como componentes del software de las computadoras son una parte fundamental debido a que a través de ellos se simplifica y rentabiliza el trabajo de una forma asombrosa. Estos, cada día más evolucionados, permiten que un gran número de usuarios estén trabajando con una misma máquina sin que apenas se den cuenta de ello.

Hoy en día es una realidad que cualquier programador pase gran parte de su tiempo dialogando con el sistema operativo, por lo que un buen conocimiento del mismo es un seguro de una utilización eficiente y de alto rendimiento.

En el mercado actual de libros existe una variedad de titulos sobre el tema de los sistemas operativos y sobre alguno de ellos en concreto, en su mayoria suelen ser grandes tratados y libros que para un estudiante como a los que en nuestro caso nos dirigimos tratan los temas con excesivo rigor y extensión. El presente libro consiste en una introducción a la teoría de los sistemas operativos complementada con una breve reseña de caracteristicas de algunos de los sistemas operativos que actualmente se encuentran en todo su esplendor.

Como punto de partida se han tomado los cuestionarios oficiales de la asignatura de nombre «Sistemas Operativos y Compiladores» de la rama Administrativa y Comercial especialidad Informática de Gestión, así como de la asignatura «Sistemas Operativos» del Bachillerato de Administración y Gestión (BAG), sin olvidarnos de la asignatura «Sistemas Operativos» del módulo profesional de nivel 3 denominado Módulo Profesional de Programador de Gestión.

Por ello, el libro va dirigido fundamentalmente a estudiantes de las mencionadas asignaturas y también a aquellas personas que por su trabajo o afición tienen en la informática una buena inversión en su tiempo, por tanto, esquematizando los posibles receptores del presente libro, tendremos:

- Alumnos de Formación Profesional de Informática de Gestión.
- Alumnos del Bachillerato de Administración y Gestión.
- Alumnos del Módulo Profesional de Programador de Gestión.
- Alumnos de enseñanzas de Informática no reglada.
- Alumnos de Escuelas Universitarias de Informática.
- Profesionales y aficionados a la Informática.
- Usuarios en general.

A lo largo del libro, se presentan una serie de conceptos existentes en la totalidad de los sistemas operativos actuales de forma progresiva, para concluir con una visión general de algunos sistemas operativos. La distribución de materias es la siguiente:

En el Capítulo 1 se habla de los conceptos básicos donde se define un sistema operativo desde varios puntos de vista y a continuación se hace un recorrido histórico por la evolución de los sistemas operativos con sus formas y métodos de trabajo. En el Capítulo 2 se definen una serie de conceptos generales de uso frecuente en entornos informáticos que se utilizan en el resto del libro y es preciso que estén suficientemente aclarados desde el principio.

El Capítulo 3 presenta la estructura y prestaciones de un sistema operativo para pasar al Capítulo 4, cuya importancia es vital debido a que en él se tratan los dos elementos principales, el núcleo y los procesos. El Capítulo 5 habla de la planificación del procesador y en él se estudian las distintas políticas y mecanismos de gestión del procesador para así dar un buen servicio a todos los procesos que se encuentren presentes en un determinado momento.

En el Capítulo 6 se analiza el proceso paralelo y la concurrencia entre programas donde además se estudian sus necesidades de sincronización y sus problemas de exclusión mutua e interbloqueo y las formas de corregirlos.

Los Capítulos 7, 8 y 9 hablan de las gestiones que, además de la del procesador, tiene que realizar todo sistema operativo, en primer lugar la gestión de la memoria principal o almacenamiento primario con las técnicas de las particiones, segmentación, paginación y memoria virtual, en segundo lugar la gestión de la entrada/salida y por último la gestión del almacenamiento secundario.

El Capítulo 10 analiza los aspectos referentes a la seguridad interna y externa en el entorno de un sistema informático y en particular de la seguridad del sistema operativo y aquella que debe ofrecer el mismo a los usuarios y a la información.

Por último, en el bloque de introducción a la teoría de los sistemas operativos, el Capítulo 11 habla de los programas traductores que se nos ofrecen en el entorno de todo sistema operativo para realizar la necesaria conversión de programas escritos en lenguajes de alto nivel a sus correspondientes y equivalentes programas escritos en lenguaje máquina para su posterior ejecución.

Los Capítulos 12, 13, 14, 15, 16 y 17 realizan las mencionadas reseñas de características generales de los sistemas operativos MS-DOS, UNIX, OS/2, MVS, VMS y OS/400 respectivamente.

Esperamos que nuestro trabajo se vea gratificado con la confianza que puedan poner en él aquellos lectores que encuentren en este libro un buen primer paso para introducirse en el apasionante mundo del software de sistemas.

Antes de cerrar este prólogo, queremos, como siempre, agradecer a la Editorial McGraw-Hill, a su Director General D. Antonio García-Maroto y al Gerente de la División Técnico-Vocacional D. Wenceslao Ortega la confianza con que han reconocido el esfuerzo, dedicación y coordinación que hemos puesto en la realización de este libro. En particular queremos agradecer la confianza, ánimos y paciencia de nuestro editor, amigo y compañero Teodoro Bartolomé, sin el cual estas palabras se las hubiese llevado el viento.

Madrid, agosto 1991

Los Autores

CAPITULO 1

Conceptos básicos

1.1. INTRODUCCION

En este primer capítulo del libro se tratan los conceptos básicos necesarios para una perfecta comprensión del resto de capítulos. Se define, en primer lugar, el concepto de Sistema Operativo, para pasar luego a una breve reseña histórica y a los modos de trabajo que en la actualidad se están realizando.

1.2. CONCEPTO DE SISTEMA OPERATIVO

En primer lugar definimos el elemento físico en el que se centra toda la actividad informática, es decir, la computadora.

> Una **computadora** es una máquina de origen electrónico con una o más unidades de proceso y equipos periféricos controlados por programas almacenados en su memoria, que pueden realizar una gran variedad de trabajos.

Todo este componente físico es denominado comúnmente **Hardware.** Ahora bien, al ser una máquina programable tiene que contar con información que le indique cómo utilizar todas sus unidades físicas o herramientas que la componen, conocidas como recursos, para llevar a cabo el trabajo. Esta información es lo que denominamos soporte lógico o **Software.**

Una computadora sin software sería una máquina inútil, mientras que con él puede almacenar, procesar y obtener información, editar textos, controlar el entorno, etc.

Un Sistema Operativo es un elemento constitutivo de software que en sus principios fue definido de la siguiente manera:

> Un **Sistema Operativo** es el soporte lógico que controla el funcionamiento del equipo físico.

En la actualidad, no resulta sencilla su definición, de forma que pueden darse varias desde distintos puntos de vista:

■ Punto de vista del USUARIO

> Un **Sistema Operativo** es un conjunto de programas y funciones que ocultan los detalles del hardware, ofreciendo al usuario una vía sencilla y flexible de acceso al mismo.

La ocultación de los detalles del hardware a usuarios y parte del personal informático (Figura 1.1) tiene dos objetivos:

- **Abstracción.** La tendencia actual del software en toda su extensión es la de dar una visión global y abstracta de la computadora haciendo fácil su uso ocultando por completo la gestión interna.
- **Seguridad.** Existen instrucciones en la máquina que pueden parar la computadora, interferir procesos, etc. Por ello, es necesario restringir determinadas operaciones a los usuarios creando varios niveles de privilegio, de tal forma que cada usuario tenga protegida su información y sus procesos.

Figura 1.1. Visión de la máquina por usuarios e informáticos.

■ Punto de vista de GESTOR DE RECURSOS

> Un **Sistema Operativo** es el administrador de recursos ofrecidos por el hardware para alcanzar un eficaz rendimiento de los mismos.

Los recursos fundamentales que administra son (Figura 1.2):

- El procesador.
- La memoria.
- La entrada/salida.
- La información.

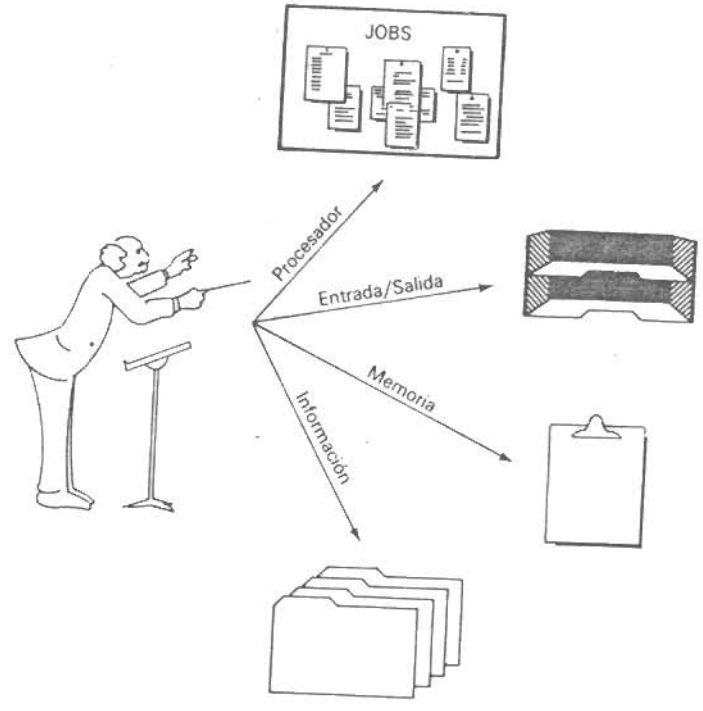


Figura 1.2. El Sistema Operativo como Gestor de Recursos.

Los sistemas operativos construyen recursos de alto nivel, que denominaremos **VIRTUALES,** a base de encubrir los realmente existentes de bajo nivel, que denominaremos **FISICOS.** Por tanto, desde el punto de vista del usuario o de un proceso, la máquina física es convertida por el Sistema Operativo en una **máquina virtual,** también conocida como **máquina extendida,** que, a diferencia de la física, ofrece muchas más funciones y más cómodas de utilizar (Figura 1.3).

El sistema operativo proporciona, además, servicios de los que no dispone el hardware, como pueden ser, por ejemplo, la utilización de la computadora por varios usuarios simultáneamente, interacción entre usuario y programa en ejecución al mismo tiempo, etc.

Para concluir, vamos a dar una definición más académica que las anteriores, basándonos en la definición previa de los términos SISTEMA y OPERATIVO.

- **Sistema.** Conjunto de personas, máquinas y cosas que, ordenadamente relacionadas entre sí, contribuyen a lograr un determinado objetivo.
- **Operativo.** Personas, máquinas y cosas que trabajan conjuntamente y consiguen el objetivo deseado.

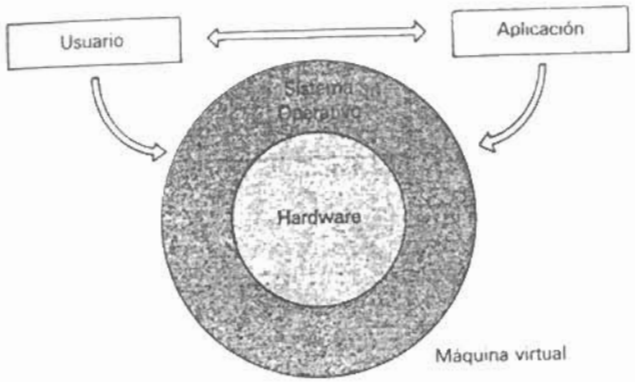


Figura 1.3. Esquema de la máquina virtual o extendida.

De las definiciones anteriores podemos obtener la de sistema operativo de la siguiente forma:

SISTEMA OPERATIVO

Conjunto de programas que ordenadamente relacionados entre si, contribuyen a que la computadora lleve a cabo correctamente su trabajo.

Como resumen, podemos decir que los sistemas operativos cubren dos objetivos fundamentales:

- Facilitar el trabajo al usuario.
- Gestionar de forma eficiente los recursos.

Ahora bien, no todo el software que se ejecuta sobre el hardware se considera como sistema operativo. Existen dos tipos de programas que conviene distinguir:

- **Programas del sistema.** Son los que manejan el hardware, controlan los procesos, hacen más cómodo el entorno de trabajo, etc.
- **Programas de aplicación.** Son los que resuelven un problema concreto de los usuarios y que no son suministrados con el sistema operativo. Son programas diseñados y codificados por analistas y programadores de aplicaciones conjuntamente con los usuarios.

1.3. EVOLUCION DE LOS SISTEMAS OPERATIVOS

Los objetivos mencionados anteriormente, facilidad de uso y gestión eficiente, no son siempre compatibles. En un principio se dio más importancia a la gestión eficiente, mientras que hoy en día se atribuye más importancia al uso sencillo.

Para comprender las razones de este cambio de actitud, haremos un breve resumen de la historia de los sistemas operativos.

1.3.1. Las primeras computadoras

En los comienzos de la historia de las computadoras destacan Howard H. Aiken, que en 1944 construyó la primera computadora electromecánica MARK-I en la Universidad de Harvard, y John W. Mauchly y J. Presper Eckert Jr., que construyeron en la Universidad de Pennsylvania la primera computadora electrónica a base de válvulas de vacío ENIAC (Figura 1.4).

Diversos factores, como el gran tamaño de las máquinas, el precio, la dificultad de uso y la escasez de recursos, hacían necesario buscar la forma de rentabilizar aquello de lo que se disponía.

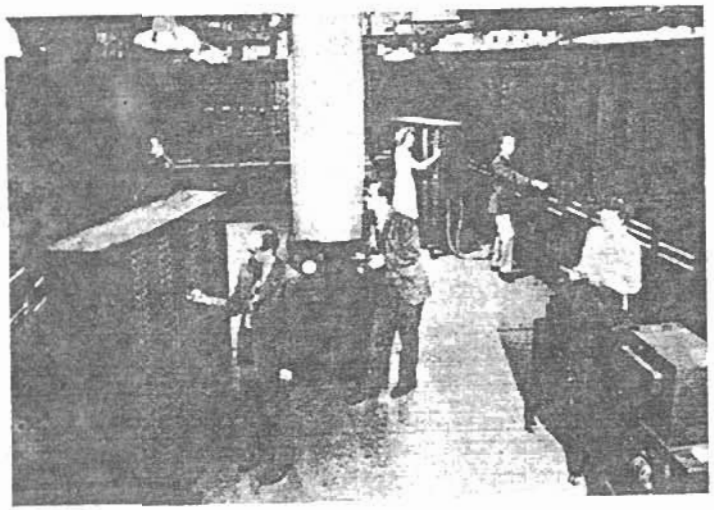


Figura 1.4. La computadora ENIAC (Cortesía de IBM).

Estas enormes máquinas eran gestionadas por el usuario desde un tablero enchufable, donde apenas existía sistema operativo, y el único lenguaje de programación posible era el lenguaje máquina. Más tarde, la gestión de la computadora se realizaba a través de una consola, en la cual cada usuario, y de uno en uno, tenía asignado un periodo de tiempo durante el que se convertía en dueño absoluto de todo el sistema.

En estas primeras computadoras sólo era posible lo que se denomina monoprogramación, es decir, la ejecución de un solo programa que se introducía generalmente a través de una lectora de tarjetas (que aparecieron en 1950), controlándose el proceso desde la consola (Figura 1.5).

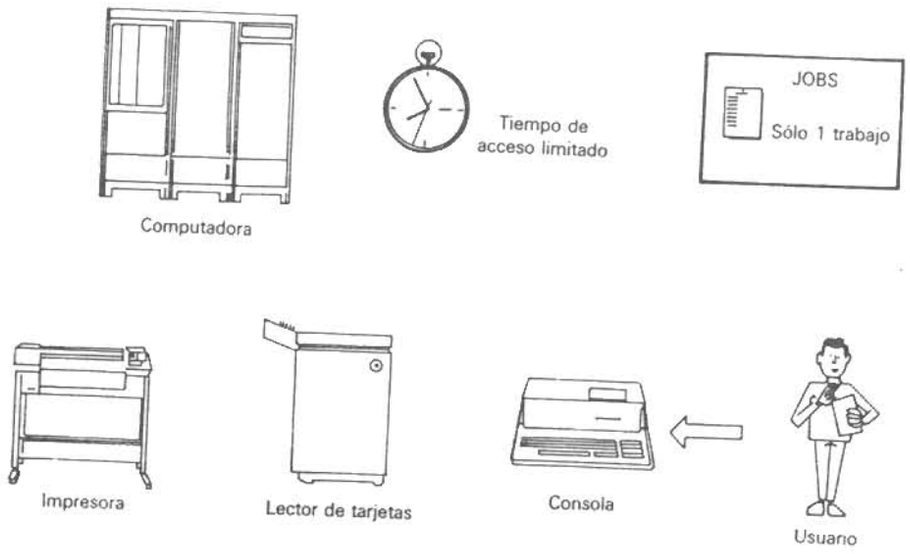


Figura 1.5. Sistema de tiempo asignado.

1.3.2. Accesos por operador

Como hemos podido apreciar en el apartado anterior, la forma de gestión de una computadora a la que acceden los usuarios de uno en uno es de alto coste económico. Para resolver este problema, a partir de 1955 se produjo una separación entre las distintas tareas que se realizaban en entornos informáticos y se establecieron puestos de programadores, operadores y personal de mantenimiento.

Una de las soluciones fue el manejo de la máquina por un especialista en operación (**operador),** cuyas misiones eran las de controlar el sistema, cargar los programas, obtener resultados, etc. A partir de este momento, el programador dejó de tener acceso directo a la computadora.

El procedimiento de trabajo era el siguiente: los programadores daban al operador los trabajos a realizar, éste los reunía y los ejecutaba uno detrás de otro en la computadora, y recogía los resultados obtenidos entregándolos a cada programador (Figura 1.6).

Una segunda solución más eficaz que la primera fue la siguiente: el operador agrupaba los trabajos con una necesidad de recursos físicos y lógicos similares y los ejecutaba como si fuesen un bloque. Por ejemplo, si entre todos los trabajos aparecían varios programas en un mismo lenguaje (FORTRAN o COBOL en aquella época), se reunían uno detrás de otro, con lo cual se conseguía cargar el compilador una sola vez, obteniéndose un considerable ahorro de tiempo.

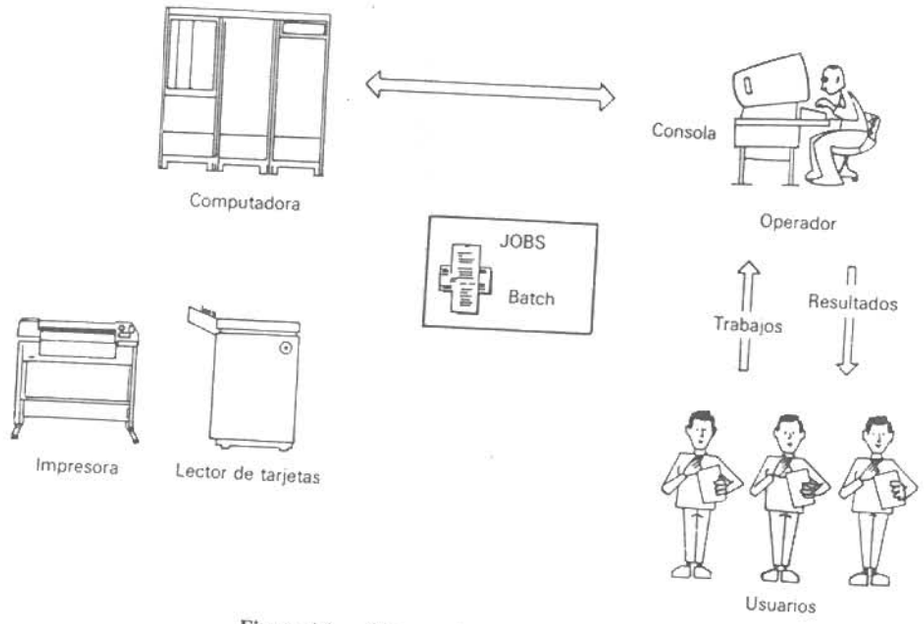


Figura 1.6. Sistemas de acceso por operador.

1.3.3. Secuencia automática de trabajos

Analizando el trabajo que desempeñaba un operador, se observó que era bastante mecánico y que podía ser automatizado en gran parte, con lo que surgió lo que se denomina secuencia automática de trabajos.

Se diseñó un pequeño programa que transfería automáticamente el control de un trabajo a otro. Este programa tomó el nombre de **Monitor Residente,** que puede ser considerado como el primer sistema operativo y que, como su nombre indica, permanecía constantemente en memoria.

En el momento de encender la computadora se daba control al programa monitor, éste a su vez daba control al primer trabajo y cuando terminaba su ejecución tomaba el control de nuevo, dando paso al segundo de los trabajos, y así sucesivamente.

El programa monitor contenía las siguientes partes (Figura 1.7):

- El secuenciador automático de trabajos.
- El intérprete de las tarjetas de control.
- Controladores software de entrada/salida (Drivers).

Para que el monitor supiera qué programa debía ejecutar y qué datos iba a tratar, se añadía al paquete de tarjetas que contenía el programa y los datos, un conjunto de tarjetas de

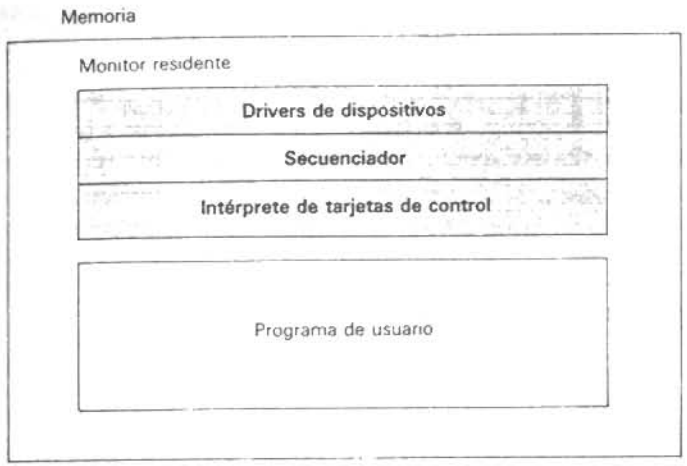


Figura 1.7. Programa monitor.

control con las indicaciones necesarias para que el monitor controlase la situación en cada momento. Estas tarjetas se ceñían a un lenguaje estricto de comandos denominado **Lenguaje de Control de Trabajos (JCL:** *Job Control Language).*

Las tarjetas de control se diferenciaban del resto por el carácter perforado en su primera columna (por ejemplo $); este carácter variaba de un sistema a otro.

En la Figura 1.8 puede observarse un bloque de tarjetas que representan un trabajo con un programa FORTRAN, los datos correspondientes y las tarjetas de control.

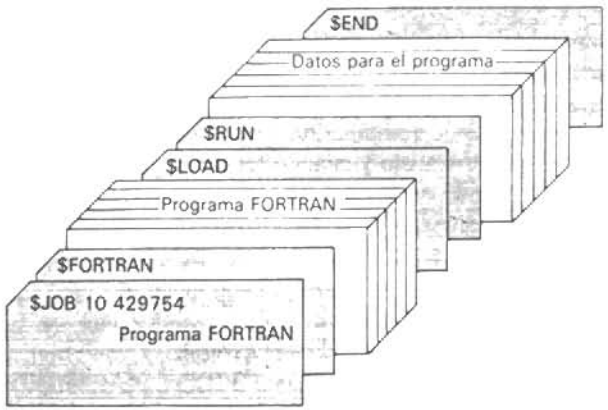


Figura 1.8. Tarjetas de control.

1.3.4. Mejora del rendimiento

Una vez resuelto el problema de la automatización en la ejecución de los trabajos, aún quedaba mucho tiempo en el que el procesador se encontraba ocioso, con la siguiente pérdida económica.

Esta situación se debía fundamentalmente a la diferencia de velocidad entre el procesador y los dispositivos de entrada/salida, cuyo funcionamiento era principalmente mecánico.

- **Off-line**

Con el paso del tiempo, aparecieron dispositivos de entrada/salida más veloces que se utilizaron para resolver el problema existente, aunque también aumentó la velocidad y capacidad de proceso del procesador.

Aparecieron las cintas magnéticas, cuya velocidad era mayor que la de las lectoras de tarjetas, pero el acceso para escribir directamente en ellas era complicado. Por ello, se impuso una técnica consistente en perforar los programas en tarjetas, que eran leídas por una lectora de tarjetas y grabadas en una cinta magnética directamente. Esta cinta se pasaba a la computadora en bloque para la ejecución de los programas, grabándose los resultados en una nueva cinta. Por último, la cinta de resultados se volcaba sobre una impresora.

Las tres operaciones anteriores se hacían en dispositivos distintos, con lo cual la computadora recibía programas y entregaba resultados a una velocidad considerable.

Las acciones de copiado de tarjetas a cinta y de cinta a impresora, al ser lentas y separarse del control de la computadora, permitían a ésta realizar otros trabajos distintos. Este tratamiento es el que se conoce como Off-line (Figura 1.9).

El proceso antes descrito contaba con algunas dificultades que obligaron a dedicar una pequeña computadora para la gestión de las copias. Así surgieron los denominados **Procesadores Satélites,** que fueron la antesala de los actuales sistemas multicomputadoras.

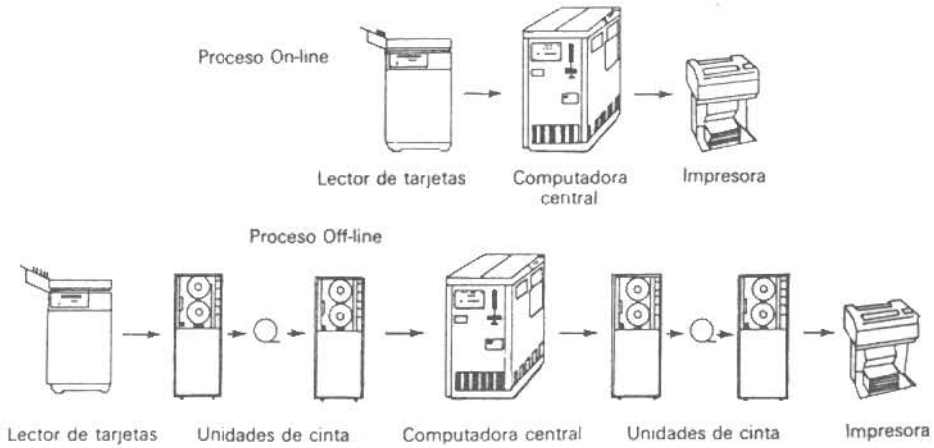


Figura 1.9. Proceso On-line y Off-line.

La ventaja principal de este sistema fue que se podían tener varias lectoras de tarjetas produciendo cintas de entrada, con lo que se mantenía ocupado el procesador la mayor parte del tiempo. Por otra parte, existía la desventaja de que un usuario tenía que esperar a que se llenara la cinta con otros trabajos para que el suyo fuese transferido a la computadora.

■ Buffering

En el caso anterior, el proceso de carga de la cinta a la computadora es relativamente lento con respecto a la velocidad interna de proceso; además se consume un tiempo adicional debido al protocolo o conversación que se establece entre el procesador y la unidad de cinta (Figura 1.10).

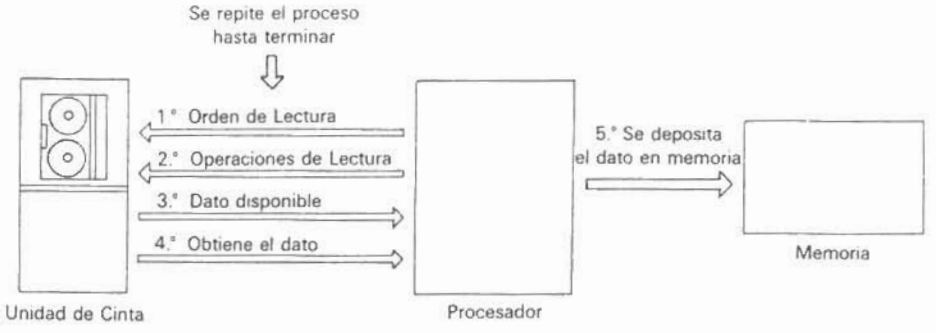


Figura 1.10. Protocolo Procesador-Cinta.

Una forma de mejorar el tiempo que se pierde en los procesos de carga de las cintas es utilizar una memoria intermedia o tampón, también denominada **Buffer,** donde la cinta va grabando datos hasta que se llena, volcándose éstos en la memoria de una vez, y mientras el procesador realiza operaciones con los datos recibidos, en paralelo se vuelve a cargar el buffer. A esta forma de trabajo se le denomina **Buffering** (Figura 1.11).

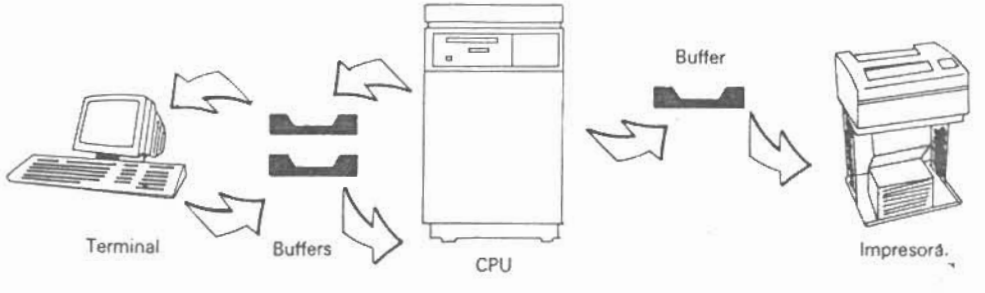


Figura 1.11. Buffering.

Es fácil entender que la misión principal de las técnicas de buffering es mantener tanto el procesador como los dispositivos de entrada/salida ocupados el mayor tiempo posible. Esto se logra solapando la entrada/salida de un trabajo con el proceso del mismo.

Las técnicas de bufering son difíciles de aplicar, ya que necesitan un control muy exhaustivo de cuándo está lleno o no el buffer, y esto sólo es posible a través de interrupciones. Por otro lado, si el dispositivo de entrada/salida es muy rápido, de tal forma que produce la información en menos tiempo que pueda tratarla el procesador, el buffering no reportaría ninguna ventaja; por eso aparecieron las técnicas de **Acceso Directo a Memoria (DMA),** para evitar la intervención del procesador en este tipo de operaciones de carga.

El buffering suele estar soportado por el programa monitor con funciones especiales o dentro de los controladores de dispositivos de entrada/salida *(Drivers).*

■ Spooling

Con la aparición del disco magnético como dispositivo de almacenamiento masivo de información con acceso directo desapareció el problema que representaban las cintas magnéticas, ya que sólo podían ser escritas después de ser leídas por completo, requiriendo además un tratamiento secuencial desde su principio hasta su final, mientras que los discos pueden ser leídos y escritos simultáneamente y en cualquier punto de su superficie.

Las técnicas de **SPOOL (Simultaneous Peripheral Operation On-Line)** permiten que la salida de un programa se escriba en un buffer y posteriormente sea llevada a un disco magnético en espera de poder ser enviada a una impresora o cualquier otro periférico de salida que en ese momento pueda estar ocupado (Figura 1.12). De esta forma, el procesador pue-

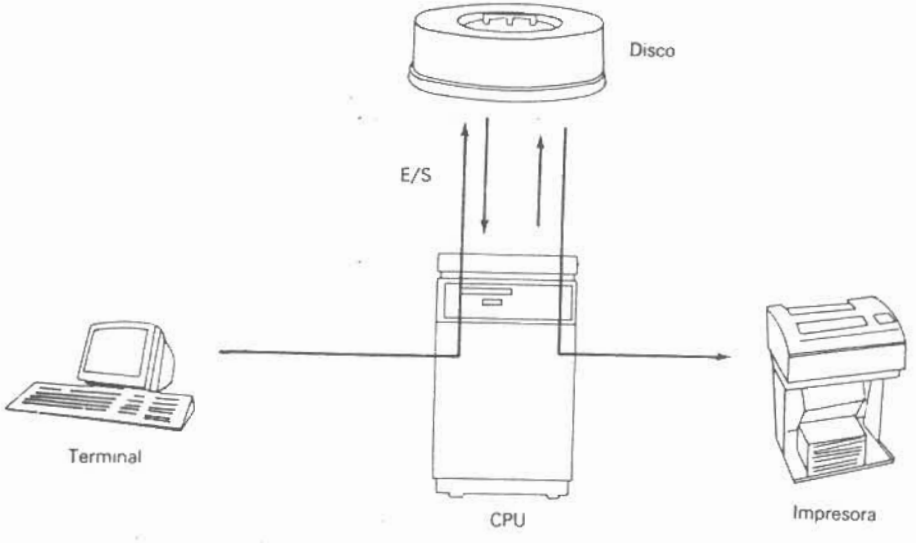


Figura 1.12. Sistema de SPOOL.

de estar ejecutando un trabajo mientras se imprimen, por ejemplo, los resultados de otro proceso anterior que ya hubiera acabado.

Un sistema de SPOOL se podría ver como una cola de archivos en espera de que llegue su turno para ser impresos, grabados en una cinta, etc. La gestión de esta cola puede ser del tipo primero en llegar-primero en salir, en función de la prioridad de cada trabajo, en función de la longitud, etc.

En general, cada dispositivo de entrada/salida tiene su propio sistema de SPOOL.

Al igual que en los sistemas de Buffering, el sistema de Spooling trata de mantener ocupados al procesador y a los dispositivos de entrada/salida el mayor tiempo posible, pero con la diferencia de que ahora se solapan operaciones de entrada/salida de unos trabajos con el proceso de otros.

1.3.5. Multiprogramación

La multiprogramación es un modo de trabajo en el que se pueden ejecutar varios programas simultáneamente con el fin de aprovechar al máximo los recursos de la computadora. Surgió de la imposibilidad, para los sistemas o modos de trabajo anteriores, de que con un solo trabajo se pudiese tener ocupados al procesador y a los dispositivos de entrada/salida durante todo el tiempo (Figura 1.13).

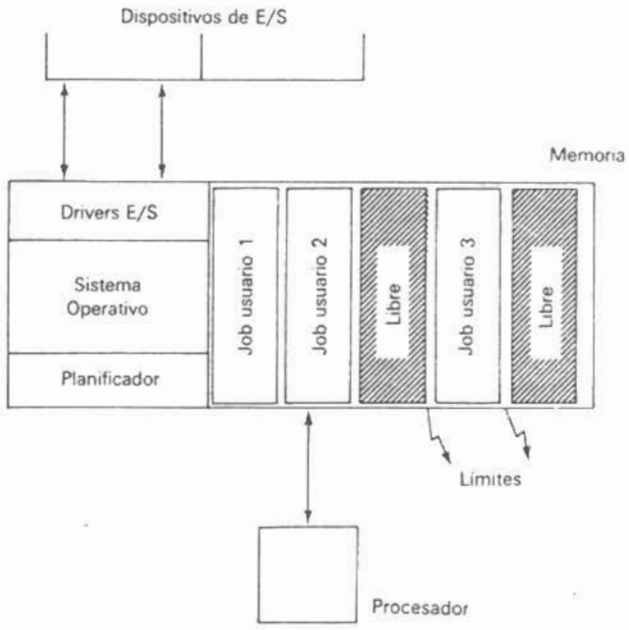


Figura 1.13. Sistema de multiprogramación.

Un trabajo realizado en una computadora, desde el punto de vista de ocupación en tiempo del procesador y los dispositivos periféricos, puede ser de dos tipos (Figura 1.14):

- **Trabajos limitados por proceso.** Son aquellos que consumen la mayor parte de su tiempo en el tratamiento de la información y muy poco en operaciones de entrada/salida.
- **Trabajos limitados por operaciones de entrada/salida.** Son los que dedican la mayor parte de su tiempo en operaciones de entrada/salida, haciendo poco uso del procesador, que se mantiene inactivo durante grandes períodos de tiempo.

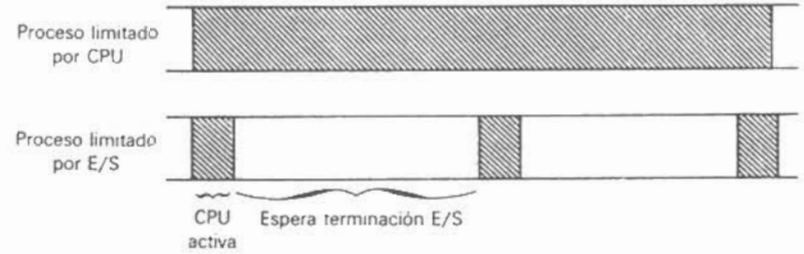


Figura 1.14. Tipos de procesos.

El segundo tipo de procesos dio lugar a una nueva técnica denominada multiprogramación, que consiste en aprovechar la inactividad del procesador durante la ejecución de una operación de entrada/salida de un proceso, en atender a otro proceso.

Desde el punto de vista de cada proceso, todos están siendo atendidos por un procesador, que podemos denominar virtual, aunque en realidad todos están atendidos por el mismo, que conmuta de uno a otro constantemente.

Desde el punto de vista del usuario, se considera que los procesos se están ejecutando en paralelo sin tener en cuenta que en cada momento sólo se atiende a uno de ellos (Figura 1.15).

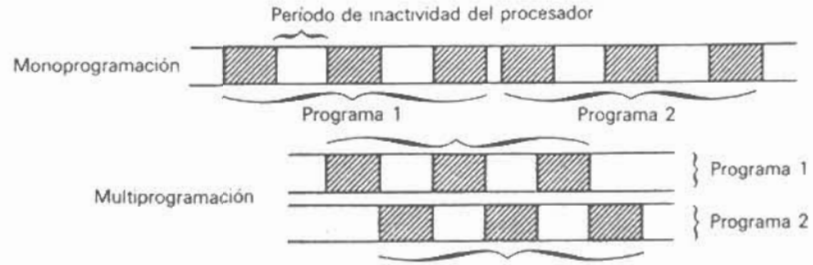


Figura 1.15. Esquema de monoprogramación y multiprogramación.

Este sistema trae consigo diversos problemas:

- El acceso al procesador debe seguir algún tipo de reglas o políticas que permitan la ejecución de todos los trabajos.
- Se hace necesario algún tipo de administración de la memoria, ya que ésta tiene que ser compartida por todos los trabajos.
- Varios trabajos pueden necesitar la utilización de un recurso al mismo tiempo, dando lugar a problemas de concurrencia.

En los sistemas operativos multiprogramados surge el concepto de planificar el procesador y a partir de ellos comienza una nueva estructura interna de los mismos, apareciendo un **núcleo central** (kernel) compuesto de rutinas para la gestión de la memoria central, el procesador, los dispositivos y el resto de recursos disponibles.

■ Proceso por lotes (Batch)

Se denomina proceso por lotes en sistemas multiprogramados al que no precisa intervención del usuario durante la ejecución de los trabajos, tratándose en general de trabajos largos que van solicitándose y entrando en una cola de espera del tipo FIFO (primero en entrar, primero en ser atendido) y que el procesador va tomando en un grupo determinado (por ejemplo, de cuatro en cuatro), realizándolos en paralelo (Figura 1.16).

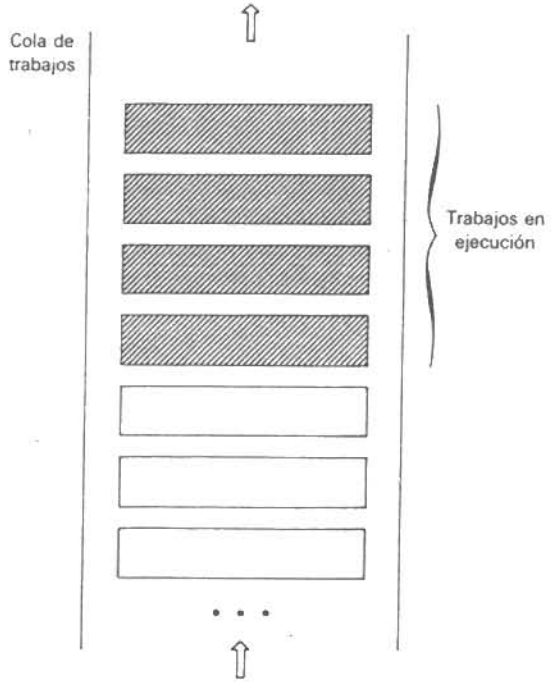


Figura 1.16. Proceso por lotes.

■ Tiempo compartido (Time Sharing)

Dado el inconveniente que tenía la multiprogramación por lotes de no permitir el diálogo entre el usuario y el proceso, el siguiente paso en el desarrollo de los sistemas operativos fue la introducción de la **multiprogramación interactiva.**

Apareció a la vez que los terminales conversacionales o interactivos (teclado-pantalla), en los que el usuario ya no tenía que suministrar todos los datos al principio de la ejecución del proceso, sino que podía ir dándolos a medida que el proceso los iba necesitando, de igual forma que iba recibiendo respuesta inmediata a sus datos.

En este modo de trabajo la organización no se realiza por trabajos, sino por sesiones. Una sesión es todo el conjunto de trabajos que se realizan desde que un usuario se conecta a la computadora hasta que se despide de la misma. Durante estas sesiones se pueden realizar multitud de operaciones controladas por un proceso denominado **intérprete de comandos,** que mantiene el diálogo entre el usuario y el sistema operativo. Este proceso puede dar lugar a otros muchos para realizar todas las demandas de usuario (Figura 1.17).

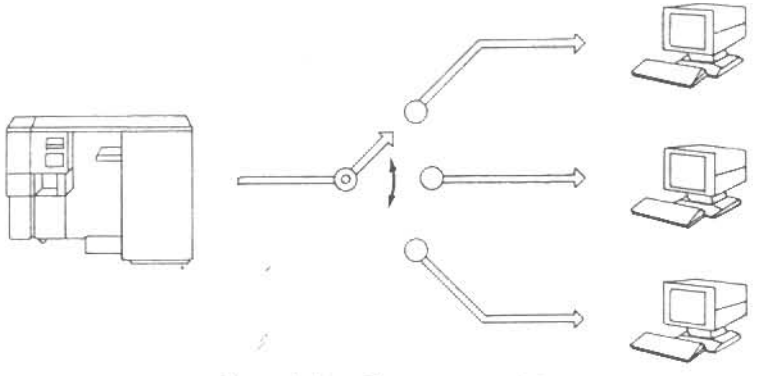


Figura 1.17. Tiempo compartido.

Durante una sesión, el usuario cree tener a su disposición todos los recursos de la computadora, aunque existan otros usuarios con sus sesiones actividadas simultáneamente.

En estos modos de trabajo, la depuración de programas que con anterioridad se realizaba con volcados de memoria, ahora se hace de forma interactiva suspendiendo la ejecución del proceso momentáneamente mientras se estudia el problema que pueda haber aparecido.

Los sistemas de tiempo compartido se caracterizan por:

- Ser muy conversacionales.
- Atender a varios usuarios simultáneamente.
- Ofrecer tiempos de respuesta relativamente cortos (segundos).
- Mantener una interrogación secuencial de peticiones de usuarios (polling).
- Poseer una fuerte gestión de archivos.
- Utilizar técnicas de buffering y spooling.
- Gestionar memoria virtual.

En general, los sistemas operativos suelen tener simultáneamente las técnicas de lotes y tiempo compartido; de esta manera cada usuario elige el sistema a aplicar a sus procesos. Estos sistemas son los más utilizados en la actualidad, y entre ellos podemos citar: el MVS de IBM, el UNIX y el VMS de DEC.

Con este tipo de sistemas operativos surgieron una gran cantidad de utilidades para facilitar el trabajo a los usuarios; entre ellas la más utilizada fue el editor de texto, que permite generar de forma interactiva cualquier archivo de datos (documento o programa).

- **Tiempo real (Real Time)**

El tiempo real es otra modalidad de los sistemas operativos multiprogramados, en que se necesita un tiempo de respuesta pequeño ante cualquier petición.

Suele emplearse en aplicaciones dedicadas a sistemas de control con sensores como elementos de entrada, donde es necesaria una respuesta rápida sobre el sistema a controlar. Podemos decir que un sistema trabaja en tiempo real si el tiempo de respuesta permite controlar y regular al medio sobre el que opera (Figura 1.18).

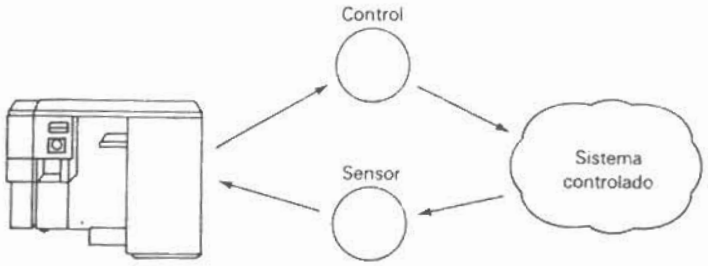


Figura 1.18. Tiempo real.

Las características principales del tiempo real son:

- Fuertes restricciones en el tiempo de respuesta (milisegundos).
- La información debe estar permanentemente actualizada.
- El sistema debe permanecer prácticamente inactivo para atender lo más rápidamente posible cualquier evento en la entrada.
- Manejo eficaz de interrupciones.
- Manejo sencillo de prioridades.
- Gestión de memorial real.

Ejemplos de estos sistemas operativos son los que controlan procesos industriales, reservas de billetes, etc.

1.3.6. Proceso distribuido

El siguiente paso en la evolución de los sistemas operativos fue el proceso distribuido, que consiste en la conexión de computadoras entre sí a través de una gran variedad de dispositivos, existiendo varias modalidades entre las que podemos citar la conexión de varias computadoras compartiendo un mismo almacenamiento principal o aquellos que se conectan a una misma red nacional o internacional para el intercambio de información.

En la Figura 1.19 puede verse la conexión de varias computadoras a través de una red.

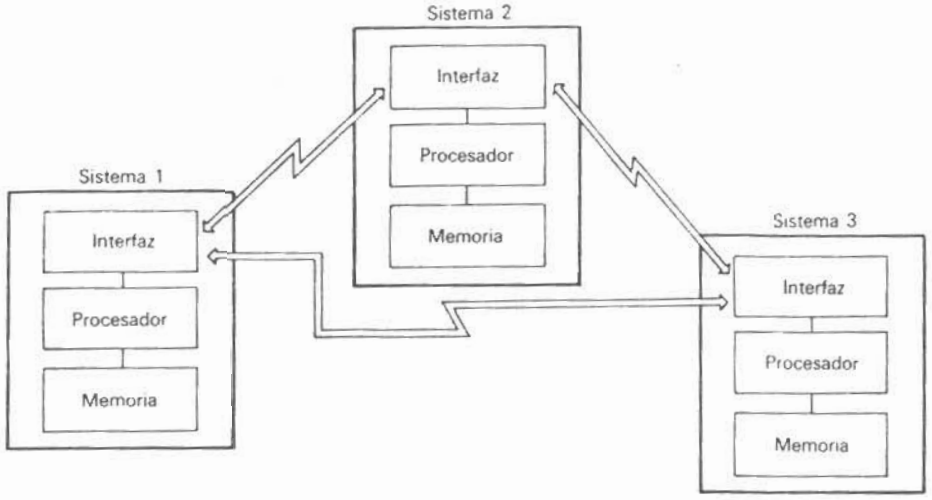


Figura 1.19. Proceso distribuido.

1.3.7. Multiproceso

Uno de los problemas actuales en el proceso de datos ha sido la aparición de aplicaciones que manejan tal cantidad de información, que un solo procesador no es capaz de procesarla en el tiempo requerido. Podemos citar el caso de los sistemas expertos, donde el volumen de datos es extremadamente grande y se necesitan unos tiempos de respuesta extremadamente pequeños; además, los algoritmos que manejan estos datos son complicados y necesitan muchas operaciones para la obtención de los resultados.

En estos casos, las computadoras convencionales no satisfacen las necesidades, por lo que se hizo necesaria la descomposición de algoritmos en subalgoritmos más sencillos, de manera que cada uno pueda tratar un subconjunto de los datos con cierta independencia de los otros. Al final se conjuntan estos datos, obteniendo el resultado final de todo el proceso. Estos subalgoritmos pueden trabajar en paralelo, tratando cada uno su porción de información al mismo tiempo.

Para ello, se están desarrollando en el momento actual diversas máquinas que, siendo una sola computadora, contienen varios procesadores que pueden trabajar conjuntamente. Con ello puede quedar satisfecha la necesidad de proceso de varios algoritmos simultáneamente (cada uno en un procesador) y con un ahorro sustancial en el tiempo de ejecución (Figura 1.20).

En este tipo de computadoras, el sistema operativo es complejo debido a que tiene que administrar varios procesadores de tal manera que la carga y reparto de los trabajos debe equilibrar y optimizar al máximo el proceso global.

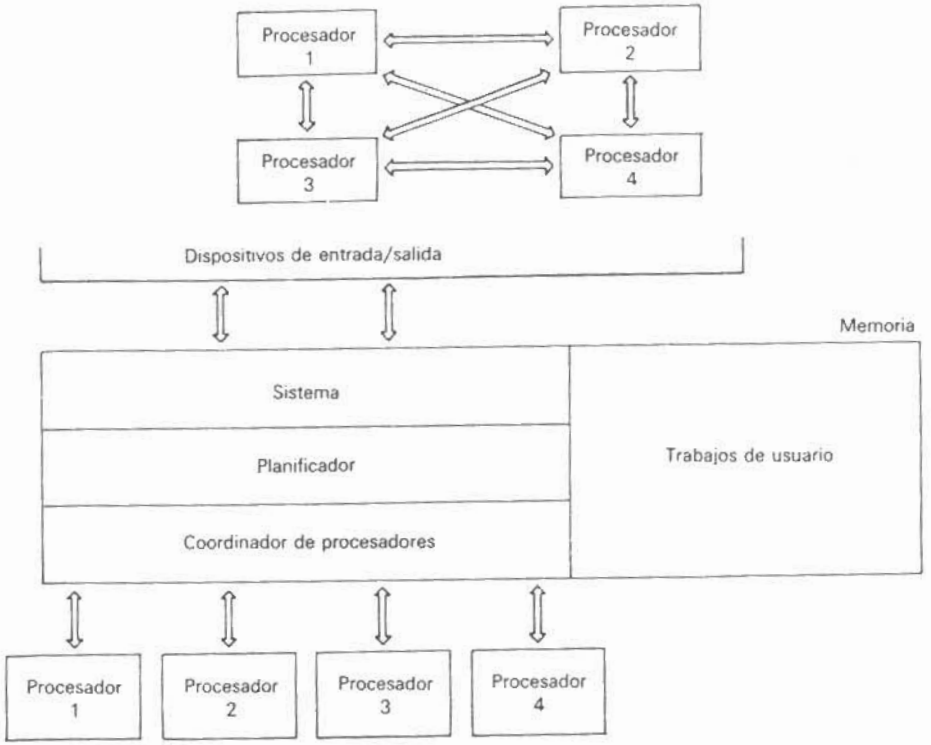


Figura 1.20. Multiproceso.

Para finalizar este recorrido por los diferentes tipos de sistemas operativos y modos de trabajo, veamos en la Figura 1.21 las diferencias existentes en la ejecución de cuatro procesos A, B, C y D en tres sistemas con métodos de monoprogramación, multiprogramación y multiproceso.

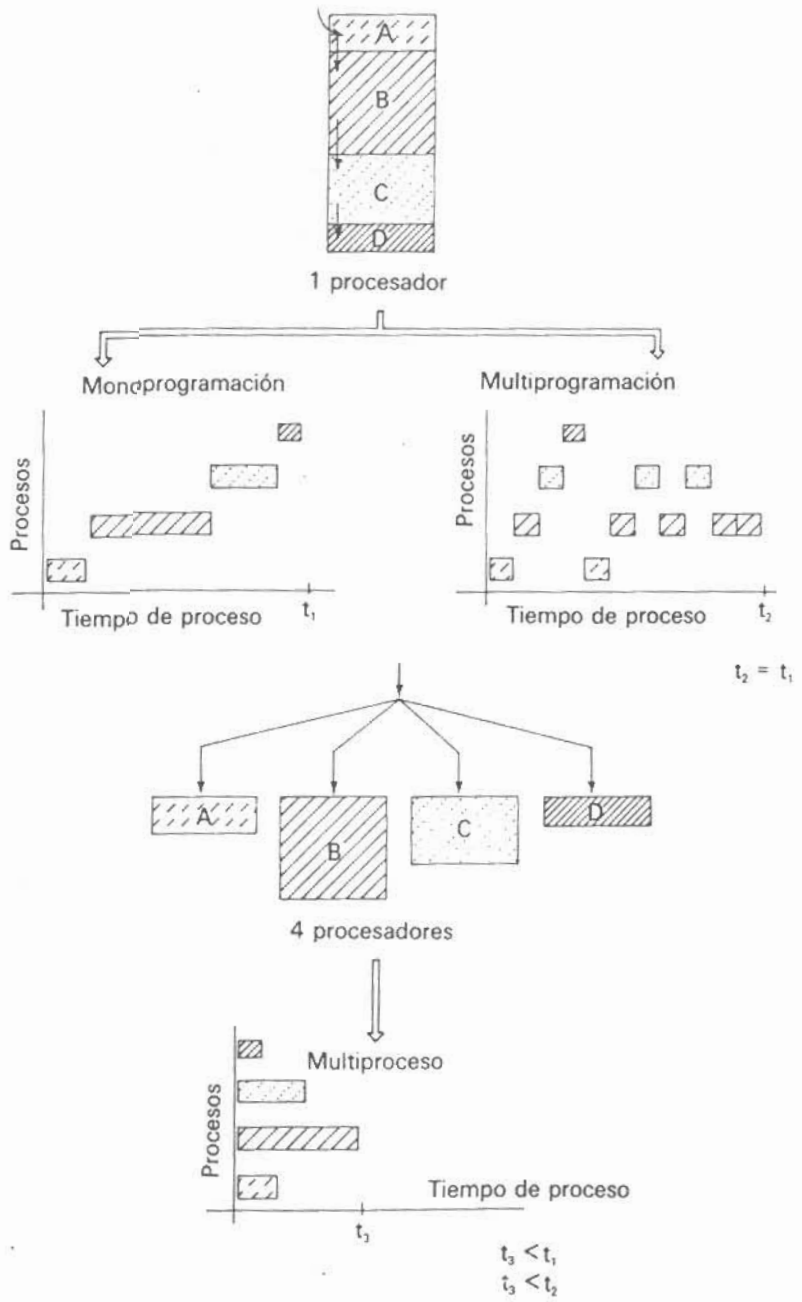


Figura 1.21. Comparación de sistemas de gestión del procesador.

CUESTIONES

1. Definir el concepto de sistema operativo.
2. ¿Qué recursos fundamentales administra un sistema operativo?
3. ¿Qué significado tiene el concepto de máquina virtual con respecto a la máquina física que realmente existe?
4. ¿Qué puestos de trabajo especializados aparecieron a partir de 1955 para realizar las distintas tareas necesarias en un entorno informático?
5. ¿Qué es un programa monitor residente?
6. ¿Para qué se utilizaba el lenguaje de control de trabajos JCL en sistemas con tarjetas perforadas?
7. Comentar en qué consiste el proceso off-line.
8. ¿Qué diferencias hay entre Buffering y Spooling?
9. Indicar las diferencias existentes entre sistemas de monoprogramación y de multiprogramación.
10. Describir brevemente los sistemas multiprogramados por lotes, tiempo compartido y tiempo real.
11. ¿Qué ventajas tiene la utilización de más de un procesador en una misma computadora?
12. Señalar diferencias entre proceso distribuido y multiproceso.
13. Un trabajo que se realiza en computadora, desde el punto de vista de ocupación del procesador y los periféricos, ¿de qué tipos puede ser?
14. ¿Cuáles son las características de los sistemas de tiempo compartido?
15. Realizar un símil de la vida real con un sistema operativo (algo parecido a la representación de la Figura 1.2).

CAPITULO 2

Conceptos generales

2.1. INTRODUCCION

En este capítulo vamos a definir una serie de conceptos generales de uso frecuente en entornos informáticos que se utilizan en el resto del libro y que hemos creído conveniente adelantar para evitar la falta de reconocimiento de los mismos.

La mayor parte de las definiciones están basadas en recomendaciones dadas en los estándares de calidad de software del Institute of Electrical and Electronic Engineering (IEEE).

Estos conceptos se han separado en cuatro grupos que pasamos a exponer seguidamente.

2.2. TERMINOLOGIA GENERAL

En primer lugar, exponemos conceptos y terminología general, común a todas las disciplinas que giran en torno a la informática.

El primer bloque de conceptos se refiere a elementos software muy utilizados, con terminología que en ocasiones depende del lenguaje de programación en el que estén escritos.

Algoritmo *(Algorithm).* Conjunto de acciones correctamente definidas para la solución de un problema en un número finito de pasos.

Programa *(Program).* Conjunto de sentencias escritas en un determinado lenguaje para satisfacer un algoritmo en computadora. Un programa puede ser ejecutado de forma directa.

Subprograma *(Subprogram).* Conjunto de sentencias que realizan parte de un algoritmo y sólo pueden ser ejecutadas por llamada desde un programa o algún otro subprograma.

Subrutina *(Subroutine).* Secuencia de instrucciones que pueden ser invocadas por una instrucción de llamada (call) desde otro programa, recibiendo y devolviendo valores de entrada y salida a través de parámetros.

Función *(Function).* Secuencia de instrucciones que pueden ser llamadas por su nombre para evaluar una expresión o función generalmente matemática, devolviendo un valor al punto de llamada.

Procedimiento *(Procedure)*. Subprograma que realiza parte de un algoritmo recibiendo y entregando valores a través de parámetros y cuya ejecución está supeditada a llamada por su nombre. Es sinónimo de subrutina.

Rutina *(Routines)*. Porción de programa que realiza una tarea específica de uso frecuente. Son rutinas las funciones, procedimientos, subprogramas y subrutinas.

Corrutinas *(Coroutine)*. Dos o más módulos que se pueden llamar entre sí, pero que no están ligados por relaciones de subordinación.

Parámetro *(Parameter)*. Variable que se utiliza para pasar valores entre las rutinas de un programa.

Variable *(Variable)*. Elemento de memoria al que se hace referencia por un nombre simbólico y se utiliza en un programa o rutina para almacenar un dato que puede variar.

Constante *(Constant)*. Elemento de memoria al que se hace referencia por un nombre simbólico, utilizándose para almacenar un dato que no puede variar.

Puntero *(Pointer)*. Identificador que indica la localización de un dato por su posición o dirección.

Código reubicable *(Relocatable code)*. Código máquina en el que se necesita convertir las direcciones relativas en absolutas antes de su ejecución en la computadora.

Interactivo *(Interactive)*. Sistema en el que cada acción del usuario en el terminal provoca una respuesta casi inmediata. También se conoce como conversacional.

Interfaz *(Interface)*. Elemento compartido entre dos partes para que interactúen o se comuniquen entre sí. Se puede considerar como las reglas existentes para establecer dicha comunicación.

En este segundo bloque de terminología general aparecen conceptos referidos a personas que utilizan las computadoras con unas determinadas características.

Usuario *(User)*. Persona que utiliza la computadora para realizar un trabajo y obtener de él unos resultados.

Superusuario *(Superuser)*. Usuario que tiene acceso a cualquier punto del sistema sin restricciones y cuenta con todos los privilegios.

Usuario privilegiado *(Privileged User)*. Usuario que cuenta con ciertos privilegios para realizar algunas funciones o acceder a cierta información no accesible al resto de usuarios.

Usuario no privilegiado *(Non privileged user)*. Usuario que sólo puede acceder a su propia información y a la que le permitan los demás.

Operador *(Operator)*. Persona especializada y dedicada a manipular la computadora con objeto de realizar determinados trabajos de accionamiento y control de los equipos.

Programador *(Programmer)*. Persona especializada y dedicada a codificar programas en lenguajes de alto o bajo nivel para ser ejecutados en computadora. Otras misiones de los programadores son el mantenimiento y documentación de los programas. Podemos diferenciar dos tipos de programadores:

Programador de aplicaciones *(Applications programmer)*. Es el encargado de codificar programas para los usuarios finales. En general suelen desarrollar aplicaciones compuestas por uno o más programas sin privilegios especiales.

Programador de sistemas *(Systems programmer)*. Es el encargado de desarrollar programas y utilidades del sistema operativo. Se necesita mayor cualificación que para la programación de aplicaciones.

Administrador del sistema *(System administrator)*. Persona técnica encargada de gestionar el sistema: contabilidad, usuarios, permisos, etc. Es superusuario y en sistemas operativos sin esta figura es un usuario privilegiado con todo tipo de privilegios.

Otros conceptos generales son:

Política *(Policy)*. Define cuál va a ser la operatividad del sistema operativo. Un ejemplo de la política del sistema operativo es la existencia o no de gestión de prioridades.

Mecanismo *(Mechanism)*. Define cómo se debe llevar a cabo la política del sistema operativo. Un ejemplo en este caso es el algoritmo de planificación del procesador de acuerdo con las prioridades establecidas.

Emulador *(Emulator)*. Hardware, software o firmware que imita todo o parte de un sistema virtual haciendo parecer que existe realmente.

2.3. CONCEPTOS HARDWARE

En este apartado se relacionan y definen todos aquellos elementos hardware que, no siendo objeto de los sistemas operativos, sí es necesario tener al menos un concepto básico de los mismos:

Bit *(Dígito binario - Binary digit)*. Unidad mínima de información tratada por una computadora. Puede tomar dos valores, 0 o 1.

Byte *(Octeto)*. Conjunto de 8 bits con el que la mayoría de los sistemas de codificación pueden tratar un carácter.

Palabra *(Word)*. Grupo de bits que una computadora maneja como una unidad y forman la unidad básica de operación del procesador. En la actualidad los tamaños más corrientes son 16 y 32 bits, existiendo algunas computadoras de 64 bits.

Señalizador *(Flag)*. Indicador que señala la aparición de un error, estado u otra condición especificada. En general se refiere a un bit, aunque puede ser un byte o palabra.

Pila *(Stack)*. Conjunto de datos al que se accede según la política de primero en entrar, último en salir.

Puntero de pila *(Stack pointer).* Registro cuyo contenido es la dirección del último elemento que entró en la pila.

Contador de programa *(Program counter).* Registro que almacena en cada momento la dirección de memoria de la siguiente instrucción a ejecutar por el procesador.

Dirección *(Address).* Conjunto de bits que identifican un registro, una zona de almacenamiento o algún otro tipo de fuente o destino de datos.

Dirección absoluta *(Absolutte address).* Dirección que identifica una posición física de memoria.

Dirección simbólica *(Symbolic address).* Grupo de caracteres que representan la dirección de un dato o instrucción.

Dirección relativa *(Relative address).* Dirección de un dato representada por un valor que lleva implícito un desplazamiento.

Espacio de direccionamiento *(Address space).* Margen de direcciones de memoria que tiene disponibles un programa.

Registro de desplazamiento *(Offset register).* Registro que contiene la dirección base a partir de la cual se reubican todas las direcciones relativas de un programa compilado para obtener las direcciones absolutas.

Almacenamiento primario *(Primary storage).* Memoria real de la computadora (RAM). En ella se almacenan los datos e instrucciones de los programas que van a ser ejecutados.

Almacenamiento virtual *(Virtual storage).* Espacio de almacenamiento que puede ser interpretado como almacenamiento principal direccionable por un programa, donde las direcciones virtuales son traducidas a direcciones reales.

Almacenamiento secundario *(Secondary storage).* Memoria destinada a almacenar información durante largos períodos de tiempo. También se denominan memorias de almacenamiento masivo (cintas y discos magnéticos).

Interrupción *(Interruption).* Suspensión de un proceso para realizar una operación independiente, realizada de tal forma que pueda volver a reanudarse.

Interrogación *(Polling).* Escrutinio periódico que se realiza en espera de la aparición de algún evento con el fin de detectar el momento en que se producen.

Buffer *(Buffer).* En general, zona de memoria que tienen los dispositivos para el intercambio de información. También se conoce como memoria tampón.

Periférico *(Peripheral).* Dispositivo externo conectado a la computadora, cuya misión es la de introducir datos, extraerlos, almacenarlos o alguna combinación de estas funciones.

Ordenador principal *(Host).* Computadora donde se instala un programa. En una red de computadoras es la que ofrece a los usuarios capacidad de proceso.

Ordenador de gran potencia *(Mainframe).* Computadora que por sus características de potencia y velocidad de proceso toma la denominación especial de mainframe.

Consola maestra *(Master console).* Terminal especial para el control directo de la computadora.

2.4. CONCEPTOS FIRMWARE

El **firmware** es el conjunto de programas y datos que vienen en una computadora desde su fabricación, en una memoria que no puede ser modificada por el usuario.

Estos programas pueden venir implantados en el propio cableado del ordenador, denominándose en estos casos lógica cableada, y también pueden venir en memorias de solo lectura (Read only memory-ROM), conociéndose en estos casos como microcódigo o microprograma.

Microcódigo *(Microcode).* Se denomina así a la representación simbólica de un microprograma.

Microprograma *(Microprogram).* Secuencia de instrucciones elementales en código máquina que corresponden a una operación completa de la computadora.

En general, podemos decir que una máquina microprogramada interpreta instrucciones (microinstrucciones) no existentes directamente en el hardware, basándose en otras más elementales que si están directamente disponibles en la máquina.

En la actualidad, la microprogramación se utiliza para realizar funciones de control en una computadora y es facultad de los diseñadores de hardware.

Existen varias modalidades de firmware:

- **Programas cableados** *(Wired-in).* Son secuencias de acciones que configuran determinados algoritmos implantados en el cableado. El ejemplo más corriente es el de las calculadoras (Figura 2.1).
- **Control por programa** *(Program control).* El control se realiza por un programa o conjunto de ellos grabados en una memoria ROM, donde las instrucciones máquina son interpretadas como un juego de señales de control (Figura 2.2).

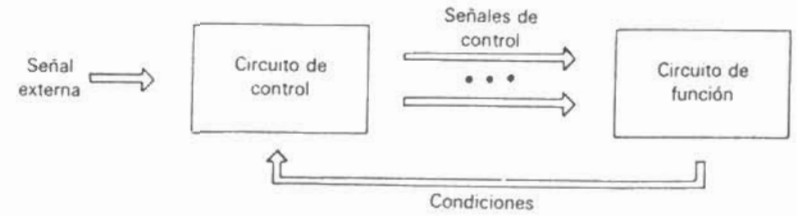


Figura 2.1. Programa cableado.

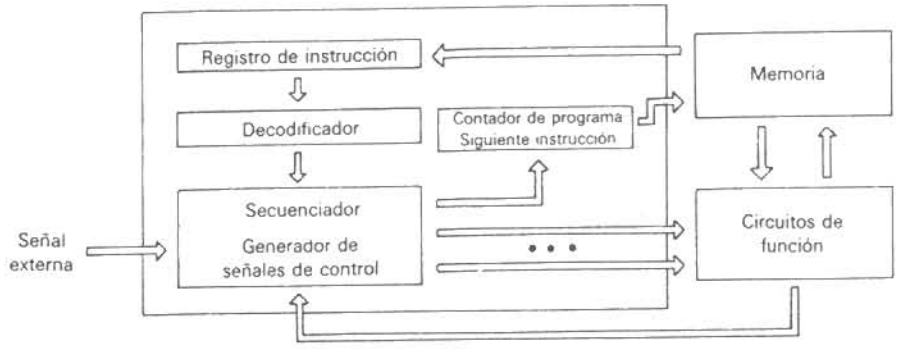


Figura 2.2. Programa almacenado.

- **Control rígido** *(Hard-wired).* Consiste en una mezcla de los dos anteriores, donde la conversión de códigos de operación de las instrucciones del programa almacenado en señales de control se establece por cableado (Figura 2.3).

- **Control por microprograma** *(Microprogramed).* Se realiza el control por medio de un programa (microprograma) almacenado en memoria ROM, cuyas instrucciones son

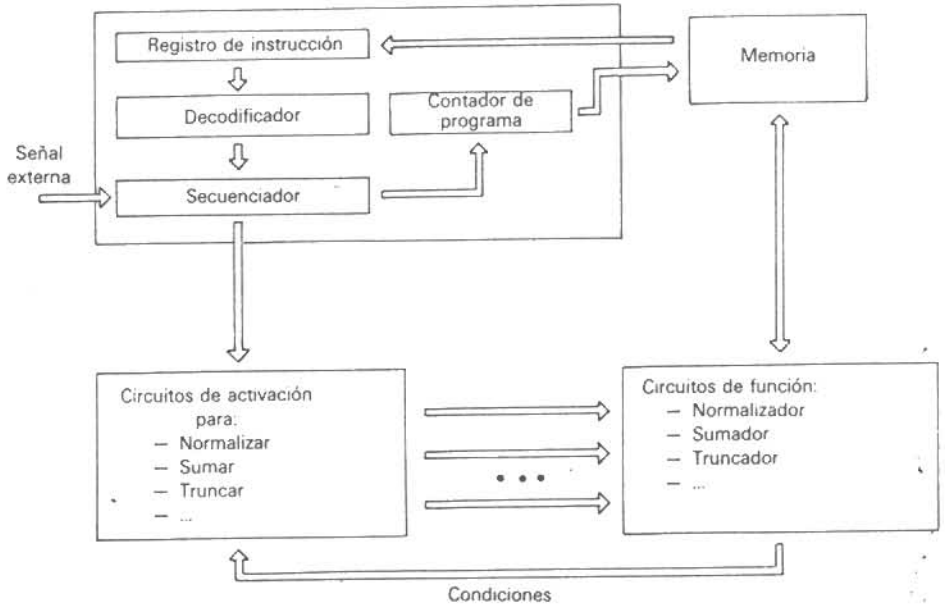


Figura 2.3. Control rígido.

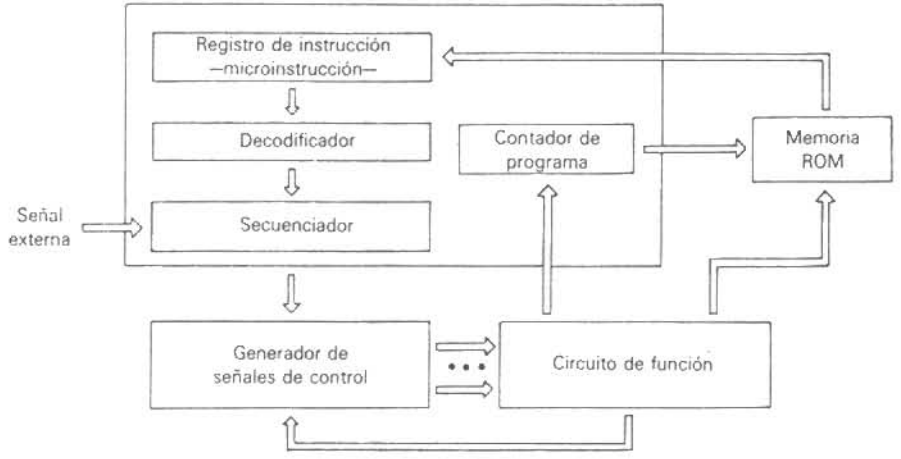


Figura 2.4. Control por microprograma.

menos numerosas que el mismo programa en código máquina, donde además se puede emular una máquina virtual más potente que la real que configura el propio hardware (Figura 2.4).

2.5. CONCEPTOS SOFTWARE

Pasamos a definir una serie de conceptos software que se encuentran relacionados con los elementos que aparecen en la Figura 2.5, donde se esquematiza el proceso de generación de software de usuario a través del denominado proceso de compilación y montaje de un programa.

Instrucción o sentencia *(Sentence).* Línea o líneas de un programa donde se da la orden de realización de una operación o conjunto de operaciones particulares. Además del término general, distinguiremos dos tipos de instrucciones:

Instrucción privilegiada *(Privileged instruction).* Es aquella que para ser ejecutada necesita que el programa o usuario tengan ciertos privilegios.

Macroinstrucción *(Macroinstruction).* Instrucción en el lenguaje fuente que es reemplazada por una secuencia definida de instrucciones en el mismo lenguaje fuente. Todas las macroinstrucciones son expandidas por el compilador o ensamblador al conjunto de instrucciones que representan.

Código máquina *(Machine code).* Representación de instrucciones y datos de un programa ejecutables directamente por una computadora.

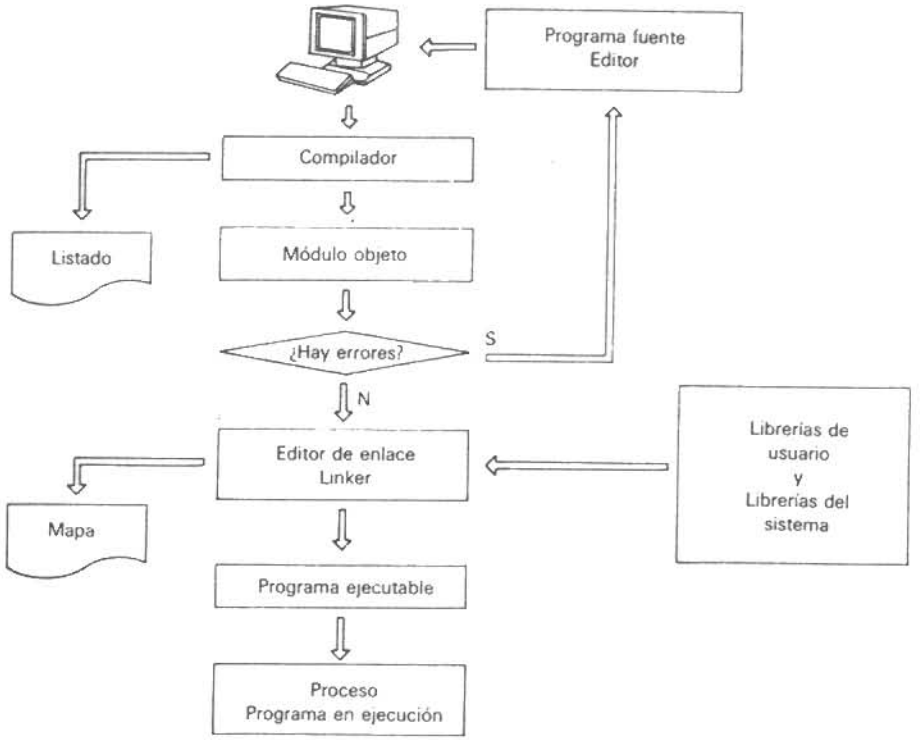


Figura 2.5. Generación de software.

Programa *(Program).* Secuencia de instrucciones que representan la resolución de un algoritmo y que pueden ser ensambladas, compiladas o interpretadas con el fin de obtener un programa ejecutable en código máquina para realizar un trabajo útil para el usuario.

Módulo *(Module).* Unidad de programa que puede ser compilada y unida a otros módulos para formar un programa completo. También lo podemos definir como una parte separable de un programa.

Los programas o módulos, según el proceso de conversión a código máquina, pueden ser:

Programa fuente *(Source Module).* Programa escrito en ensamblador o lenguaje de alto nivel (FORTRAN, COBOL, Pascal, C, etc.) que debe ser ensamblado, compilado o interpretado antes de ejecutarse en la computadora. Normalmente son editados por el usuario o programador por medio de un editor.

Módulo objeto *(Object module).* Es un módulo fuente ensamblado o compilado que está listo para ser unido a otros para formar un programa ejecutable. Si se trata de todo un programa (un solo módulo) se denomina programa objeto.

Programa ejecutable *(Executable code).* Programa construido por el editor de enlace o montador (linker) a partir de uno o más módulos objeto y de rutinas de librería. Este programa puede ser cargado en memoria y ejecutado.

En el proceso de traducción o conversión de un programa fuente a ejecutable entran en juego unos archivos cuya composición es un conjunto de módulos que pueden ser reclamados por los distintos elementos a traducir, denominados librerías. Pueden ser de tres clases:

Librería de programas *(Library).* Archivo que contiene una colección organizada de programas.

Librería objeto *(Object library).* Archivo compuesto de una colección de rutinas que pueden ser solicitadas e incorporadas por los distintos programas al hacer referencia a las mismas.

Librería del sistema *(System library).* Colección controlada de software perteneciente al sistema y que puede ser incorporado a un programa de igual forma que una rutina de librería objeto.

En la traducción de un programa fuente a código máquina se utilizan diversos programas que forman parte del sistema operativo. Estos programas traductores pueden ser:

Ensamblador *(Assembler).* Programa utilizado para traducir un programa escrito en lenguaje ensamblador a lenguaje máquina de tal forma que la traducción se realiza convirtiendo cada sentencia fuente en una instrucción máquina. En la traducción se sustituyen las direcciones simbólicas por direcciones absolutas.

Ensamblador cruzado *(Cross assembler).* Programa traductor de lenguaje ensamblador a lenguaje máquina que se ejecuta en una computadora y traduce para ejecutar en otro distinto.

Compilador *(Compiler).* Programa traductor de un lenguaje de alto nivel a su código máquina absoluto o reubicable equivalente. La traducción se realiza de tal forma que una sentencia fuente se convierte en varias instrucciones máquina, efectuando además un control previo de errores de todo el programa. Si existen errores, la traducción se interrumpe.

Compilador cruzado *(Cross compiler).* Programa traductor de lenguaje de alto nivel a lenguaje máquina que se ejecuta en una computadora, generando el código para ser ejecutado en otra distinta.

Intérprete *(Interpreter).* Programa traductor de lenguaje de alto nivel a código máquina, de tal forma que una sentencia fuente se convierte en varias instrucciones máquina y tras la traducción de cada una de ellas se ejecutan sin esperar a traducir la siguiente.

Los sistemas operativos para la construcción de programas suelen contar, además, con las siguientes utilidades:

Editor *(Editor).* Programa que permite escribir o corregir archivos de texto, generalmente programas fuente.

Editor de enlace *(Linker)*. También denominado montador, es un programa para crear un código ejecutable a partir de uno o más módulos objeto resolviendo las referencias existentes entre los mismos y asignando direcciones definitivas a los elementos reubicables. También extraen las rutinas necesarias de las librerías para incluirlas en el programa ejecutable final.

Cargador *(Loader)*. Es una rutina que lee un programa ejecutable y lo almacena en la memoria principal antes de su ejecución.

Depurador *(Debugger)*. Es un programa de ayuda que permite ejecutar un programa fuente paso a paso investigando la imagen del mismo, que se va creando en la memoria con el fin de analizarlo y corregir posibles errores.

Otras definiciones a tener en cuenta son:

Ejecución *(Execution)*. Proceso de llevar a efecto las instrucciones de un programa ejecutable previamente cargado en la memoria principal.

Proceso *(Process)*. Se utiliza este término para hacer referencia a un programa en ejecución.

Trabajo *(Job)*. Se trata de todos los pasos que ha de realizar una computadora para cubrir los objetivos de un programa.

Recurso *(Resource)*. Elemento hardware disponible en un sistema para su utilización, siendo necesario para llevar a cabo su trabajo.

Palabra clave *(Password)*. Contraseña que permite el acceso a un determinado usuario para trabajar en el sistema.

CUESTIONES

1. ¿Qué se entiende por algoritmo?
2. ¿Qué diferencias hay entre los términos programa, subprograma, subrutina, función, procedimiento, rutina y corrutina?
3. ¿Para qué se utilizan los parámetros?
4. ¿Qué diferencias hay entre un usuario y un superusuario de un sistema?
5. ¿Qué misiones son las asignadas a un programador y qué tipos de programadores existen?
6. ¿Qué se entiende por dirección y qué diferencias esenciales existen entre dirección absoluta, simbólica y relativa?
7. ¿Qué se entiende por firmware y dónde se encuentra ubicado?
8. ¿En qué se diferencian los términos microcódigo y microprograma?
9. ¿Qué se entiende por macroinstrucción?
10. ¿En qué se diferencian un programa fuente de un programa ejecutable?
11. ¿Cuál es la diferencia principal entre la traducción que hace un compilador y la que hace un intérprete?
12. ¿Qué se entiende por depurador de programas?

CAPITULO 3

Estructura y prestaciones de los sistemas operativos

3.1. ESTRUCTURA DE LOS SISTEMAS OPERATIVOS

Lo primero que hay que decidir al diseñar un sistema operativo es su finalidad y el tipo de proceso que se quiere realizar a través de él (proceso por lotes, tiempo compartido, multiproceso, etc.).

Para ello es preciso tener en cuenta las necesidades que pueden plantearse:

- **Requisitos de usuario:** Sistema fácil de usar y de aprender, seguro, rápido y adecuado al uso a que se le quiere destinar.
- **Requisitos del software:** Donde se engloban aspectos como el mantenimiento, forma de operación, restricciones de uso, eficiencia, tolerancia frente a los errores y flexibilidad.

A continuación vamos a describir las distintas estructuras que presentan los actuales sistemas operativos para satisfacer las necesidades que de ellos se quieren obtener.

3.1.1. Estructura monolítica

Es la estructura de los primeros sistemas operativos constituidos fundamentalmente por un solo programa compuesto de un conjunto de rutinas entrelazadas de tal forma que cada una puede llamar a cualquier otra (Figura 3.1).

Las características fundamentales de este tipo de estructura son:

- Construcción del programa final a base de módulos compilados separadamente que se unen a través del editor de enlace.
- Buena definición de parámetros de enlace entre las distintas rutinas existentes.
- Carecen de protecciones y privilegios.
- Generalmente están hechos a medida, por lo que son eficientes y rápidos en su ejecución y gestión.

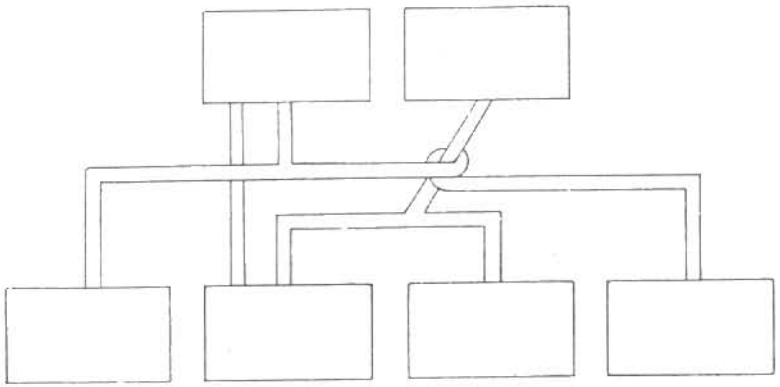


Figura 3.1. Estructura monolítica.

3.1.2. Estructura jerárquica

A medida que fueron creciendo las necesidades de los usuarios y se perfeccionaron los sistemas, se hizo necesaria una mayor organización del software.

Se dividió el sistema operativo en pequeñas partes, de tal forma que cada una de ellas estuviera perfectamente definida y con un claro interface con el resto de elementos.

Se constituyó una estructura jerárquica o de niveles en los sistemas operativos, el primero de los cuales fue el denominado THE *(Technische Hogeschool, Eindhoven)*, de Dijkstra, que se utilizó con fines didácticos (Figura 3.2).

Nivel 5 - Usuario
Nivel 4 - Archivos
Nivel 3 - Entrada/salida
Nivel 2 - Comunicaciones
Nivel 1 - Memoria
Nivel 0 - Gestión CPU
Nivel -1 - Hardware

Figura 3.2. Sistema jerárquico THE.

En este sistema operativo pueden verse las distintas capas en su orden jerárquico:

- Hardware (Nivel −1).
- Planificación del procesador (Nivel 0).
- Gestión de la memoria (Nivel 1).
- Controlador de la consola del operador (Nivel 2).

- Control de las operaciones de entrada/salida (Nivel 3).
- Gestión de archivos (Nivel 4).
- Control de programas de usuario (Nivel 5).

En la estructura anterior se basan prácticamente la mayoría de los sistemas operativos actuales. Otra forma de ver este tipo de sistema es la denominada de **anillos concéntricos** o *«rings»* (Figura 3.3).

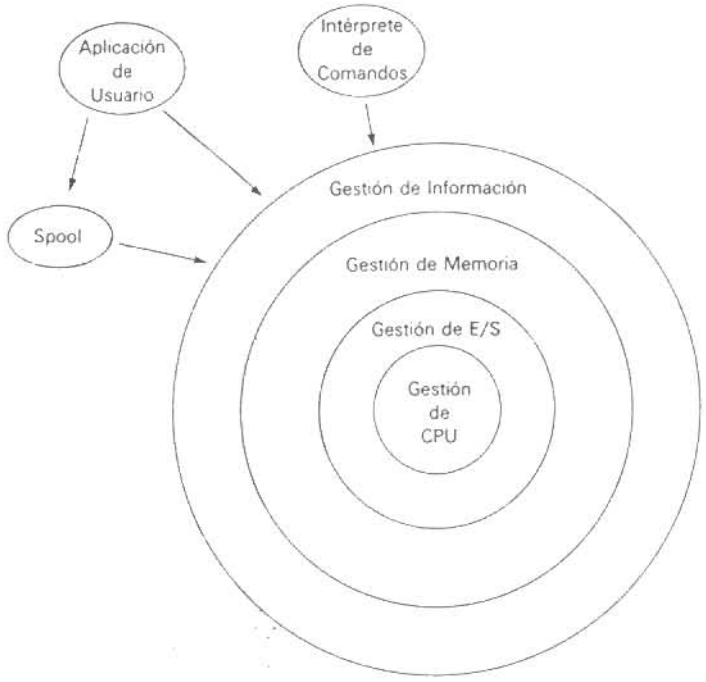


Figura 3.3. Organización jerárquica (anillos).

En el sistema de *rings*, cada anillo tiene una apertura, conocida como puerta o trampa *(trap)*, por donde pueden entrar las llamadas de las capas inferiores.

De esta forma, las zonas más internas del sistema operativo o núcleo del sistema estarán más protegidas de accesos indeseados desde las capas más externas. Las capas más internas serán, por tanto, más privilegiadas que las externas.

3.1.3. Máquina virtual

Se trata de un tipo de sistemas operativos que presentan un interface a cada proceso, mostrando una máquina que parece idéntica a la máquina real subyacente.

Estos sistemas operativos separan dos conceptos que suelen estar unidos en el resto de sistemas:

- La multiprogramación.
- La máquina extendida.

El objetivo de los sistemas operativos de máquina virtual es el de integrar distintos sistemas operativos dando la sensación de ser varias máquinas diferentes.

El núcleo de estos sistemas operativos se denomina **monitor virtual** y tiene como misión llevar a cabo la multiprogramación, presentando a los niveles superiores tantas máquinas virtuales como se soliciten. Estas máquinas virtuales no son máquinas extendidas, sino una réplica de la máquina real, de manera que en cada una de ellas se pueda ejecutar un sistema operativo diferente, que será el que ofrezca la máquina extendida al usuario (Figura 3.4).

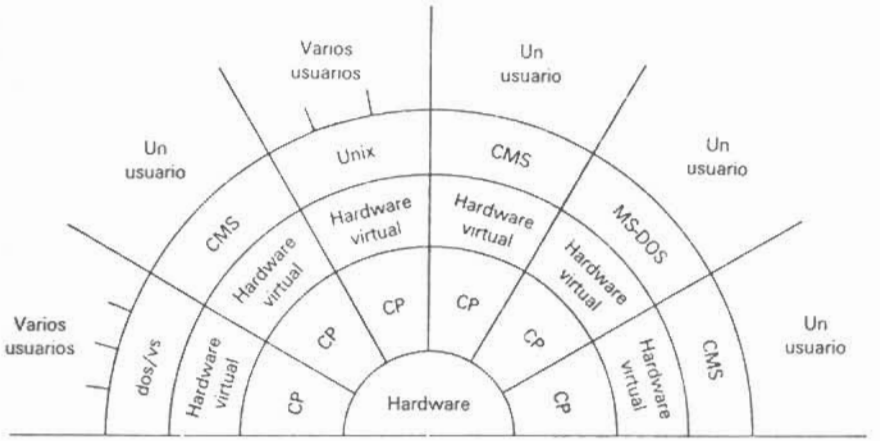


Figura 3.4. Máquina virtual.

3.1.4. Cliente-servidor

El tipo más reciente de sistemas operativos es el denominado Cliente-servidor, que puede ser ejecutado en la mayoría de las computadoras, ya sean grandes o pequeñas.

Este sistema sirve para todo; por tanto, es de propósito general y se basa en lo mismo que el resto de sistemas operativos convencionales: el núcleo y los procesos, presentando grandes diferencias en cuanto a la forma de distribuir los trabajos entre sus distintas partes. Suele suministrar mecanismos adecuados para la gestión de:

- Procesos.
- Memoria.
- Comunicación entre procesos.

El núcleo tiene como misión establecer la comunicación entre los clientes y los servidores. Los procesos pueden ser tanto servidores como clientes. Por ejemplo, un programa de aplicación normal es un cliente que llama al servidor correspondiente para acceder a un archivo o realizar una operación de entrada/salida sobre un dispositivo concreto. A su vez, un proceso cliente puede actuar como servidor para otro (Figura 3.5).

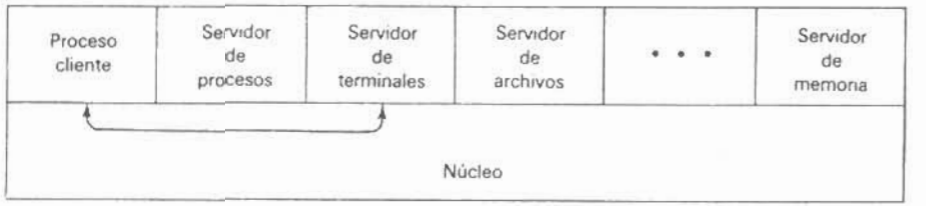


Figura 3.5. Sistema Cliente-servidor.

3.2. PRESTACIONES DE UN SISTEMA OPERATIVO

Como ya se ha dicho, la misión de un sistema operativo es la de ayudar a los usuarios en el manejo de la computadora; para ello deberá proporcionar ciertos servicios que se pueden considerar desde dos puntos de vista distintos:

- **Punto de vista del programador**
  - **Ejecución de programas:** Facilidades para cargar un programa en memoria y ejecutarlo.
  - **Operaciones de entrada/salida:** Facilidades para que un programa pueda tratar un archivo, enviar o recibir datos a un dispositivo, etc.
  - **Gestión de archivos:** Facilidades de uso y organización del sistema de archivos.
- **Punto de vista del sistema**
  - **Asignación de recursos:** Mecanismos de resolución de conflictos de asignación de recursos cuando varios procesos o usuarios están compitiendo por ellos.
  - **Contabilidad:** Control de tiempos de utilización de recursos por los usuarios para su facturación o simplemente para la obtención de estadísticas de utilización.
  - **Protección:** Defensa contra acciones no deseadas.

3.2.1. Servicios de usuario

El sistema operativo ofrece sus servicios a los usuarios de dos formas diferentes: Las llamadas al sistema operativo desde un proceso y la ejecución de programas del propio sistema.

■ **Llamadas al sistema operativo**

Constituyen el interface entre un programa en ejecución y el sistema operativo. Estas llamadas se pueden agrupar de la siguiente forma:

- Gestión de procesos.
- Gestión de operaciones de entrada/salida.
- Gestión del sistema de archivos.
- Protección.

En general, las llamadas al sistema operativo necesitan pasar algún tipo de información para la correcta ejecución del proceso a través de algún registro o bloque de parámetros. En la Figura 3.6 puede verse el esquema de ejecución de una llamada al sistema operativo.

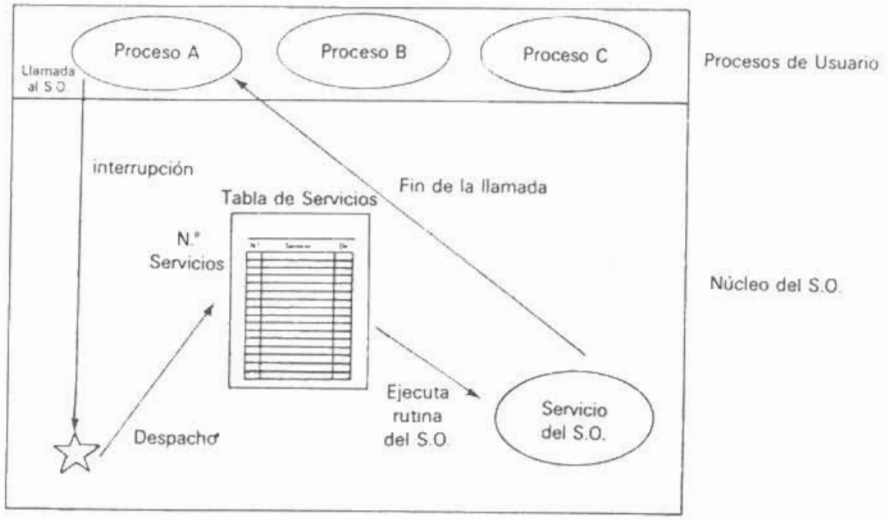


Figura 3.6. Ejecución de una llamada al sistema operativo.

Las llamadas al sistema operativo son como las llamadas a un subprograma desde el punto de vista del programa que llama. Tras la llamada, se ejecuta la rutina del sistema operativo que ha sido invocada, tomando los datos de los parámetros correspondientes; a continuación se devuelve el control al proceso que efectuó la llamada, ejecutándose la siguiente instrucción y, en su caso, tomando los datos de vuelta a los parámetros correspondientes.

■ **Programas del sistema**

En los sistemas operativos actuales, además de las funciones básicas del núcleo que pueden ser ejecutadas a través de llamadas al sistema operativo, existe un conjunto de programas del sistema o de utilidad cuya misión es resolver problemas comunes y frecuentes de los usuarios, ofreciéndolos de forma cómoda y sencilla (Figura 3.7).

Figura 3.7. Programas del sistema.

Estos programas los podemos agrupar de la siguiente forma:

- **Tratamiento de archivos:** Crean, copian, borran, renombran, imprimen, visualizan, vuelcan, ordenan, etc. un archivo, facilitando la gestión de los mismos y los directorios.
- **Información:** Dan cualquier tipo de información relativa al estado del sistema, de la memoria, de los discos, de usuarios, fecha, hora, etc.
- **Editores:** Son programas que facilitan la edición de archivos de texto o de programas fuente.
- **Ejecución:** Son programas para la construcción, depuración y carga de programas ejecutables (Linkers, debuggers, etc.).
- **Programas de utilidad:** Son programas para la gestión de Bases de Datos, compiladores, comunicaciones, etc.
- **Intérprete de comandos:** Es el más importante de todos los programas del sistema operativo, ya que es el que crea el entorno de trabajo de los usuarios.

3.2.2. Servicios del sistema

El intérprete de comandos y los programas del sistema son los que fijan el entorno y la forma de ver el sistema operativo por los usuarios. En cambio, el programador del sistema tiene una visión totalmente diferente; para él todo son recursos físicos y dispositivos que deben ser convertidos en entidades lógicas para ofrecérselas a los usuarios.

Se puede decir que un sistema operativo es un programa activado por eventos; es decir, si no hay programas en ejecución, ni operaciones de entrada/salida pendientes, etc., el sistema estará inactivo hasta que se produzca alguna nueva petición. Normalmente cada evento producirá una interrupción de la ejecución del sistema operativo.

■ **Llamadas al sistema operativo**

En esta ocasión las llamadas al sistema operativo se agrupan por el tipo de llamada y no por la acción que realizan. Pueden ser:

- **Terminación normal:** Se realiza la devolución del control al usuario cuyo proceso ha terminado, a través del intérprete de comandos.
- **Terminación anormal:** Cuando aparece un error en la ejecución de un programa, éste se da por terminado, devolviendo el control al intérprete de comandos, que indicará tal situación de error al usuario.
- **Peticiones de estado:** Se procesa la petición solicitada y se devuelve el control al programa que la solicitó.
- **Peticiones de recursos:** Los programas solicitarán recursos durante su ejecución que serán atendidos de inmediato o se entrará en un estado de espera hasta que puedan ser atendidos.
- **Peticiones de entrada/salida:** De igual forma, los programas las solicitarán y serán atendidas de inmediato o tras un pequeño período de espera.

■ **Interrupciones de los dispositivos de E/S**

Una vez que un programa en ejecución realiza una petición de entrada/salida, se pueden tomar dos tipos de acción por parte del sistema operativo:

- **El proceso queda en espera hasta que se termina la operación de entrada/salida:** En este caso el dispositivo externo, cuando termine la operación, producirá una interrupción que dará control al sistema operativo, el cual activará el proceso que estaba en espera.
- **El proceso seguirá realizando otras operaciones:** En este caso el dispositivo externo también produce una interrupción en el sistema operativo, el cual no activará el proceso puesto que no estaba en espera, pero sí le indicará que la operación solicitada ha terminado.

■ **Gestión de excepciones**

Cuando un programa en ejecución comete un error, se producirá una interrupción; por ejemplo, una división por 0, intento de violación de un archivo protegido, intento de ejecución de una instrucción no permitida o privilegiada, etc. El tratamiento de estos errores se conoce como **manejo de excepciones.**

Los sistemas operativos suelen asociar alguna función para el tratamiento de estos errores. El núcleo da control a esta función en el momento de aparición de un error de este tipo.

3.2.3. Protecciones

Los programas de aplicación de los usuarios no están exentos de errores, así como los sistemas tampoco están libres de usuarios con malas intenciones. Por ello, el sistema operativo debe incluir ciertas funciones de protección con objeto de evitar problemas entre procesos y entre éstos y el propio sistema operativo.

■ **Protección de la entrada/salida**

Todos los dispositivos externos cuentan, por parte del sistema operativo, con rutinas para el control de las operaciones de entrada/salida. Estas rutinas se denominan controladores o drivers de dispositivos y entre otras funciones protegen los accesos incorrectos, devolviendo el control al núcleo del sistema operativo, indicándole la situación errónea que se ha producido.

■ **Protección de la memoria**

En general, cada proceso tiene una zona de memoria asignada para el tratamiento de sus datos denominada espacio de direccionamiento y no puede acceder a zonas asignadas al sistema operativo o a otros procesos. Para evitarlo existen unos registros frontera que indican el límite de memoria asignado a cada proceso (Figura 3.8).

Figura 3.8. Protección de la memoria.

Si un proceso intentase acceder a direcciones que están fuera de la zona de memoria de su espacio de direccionamiento, se producirá una interrupción que dará control al sistema operativo dando cuenta del error que se ha producido.

■ Protección del procesador

Otro tipo de problemas que pueden presentarse es la presencia de bucles infinitos o accesos al procesador que no lo liberan nunca. En estos casos la única solución es la de apagar y volver a encender *(reset)* para volver a comenzar desde el principio.

Para evitarlo, el hardware incluye un temporizador que marca períodos de tiempo, de manera que al terminar un período de tiempo se produzca una interrupción y tome el control el sistema operativo.

CUESTIONES

1. ¿Cuántas y cuáles son las estructuras que presentan los sistemas operativos?
2. ¿En qué consiste la estructura monolítica?
3. ¿Cuáles son las características fundamentales de la estructura monolítica?
4. ¿En qué consiste la estructura jerárquica?
5. ¿Cuál es el objetivo a cubrir por los sistemas operativos de máquina virtual?
6. ¿Qué misión tiene el núcleo en un sistema operativo del tipo cliente-servidor?
7. ¿Qué servicios debe proporcionar un sistema operativo desde el punto de vista del programador?
8. ¿Qué servicios debe proporcionar un sistema operativo desde el punto de vista del sistema?
9. ¿En qué consiste una llamada al sistema operativo y cómo se ejecuta?
10. ¿De qué forma se agrupan los programas del sistema como servicio a los usuarios?
11. ¿Qué es una interrupción?
12. ¿Por qué causas es necesario incluir protecciones en un sistema operativo?

CAPITULO 4

El núcleo y los procesos

4.1. INTRODUCCION

El **núcleo** (kernel) de un sistema operativo es un conjunto de rutinas cuya misión es la de gestionar el procesador, la memoria, la entrada/salida y el resto de recursos disponibles en la instalación. Toda esta gestión la realiza para atender al funcionamiento y peticiones de los trabajos que se ejecutan en el sistema.

En este capítulo se tratan todos los conceptos relacionados con la entidad básica de los sistemas operativos actuales: **Los procesos.** El esquema general del mismo es el siguiente:

- Definición y concepto de proceso.
- El Bloque de Control del Proceso (PCB) como imagen donde el sistema operativo ve el estado del proceso.
- Estados por los que pasa un proceso a lo largo de su existencia en la computadora.
- Operaciones que se pueden realizar sobre un proceso.
- Clasificación de los procesos según su forma de ejecución, de carga, etc.

4.2. PROCESOS

Uno de los conceptos más importantes que gira en torno a un sistema operativo es el de proceso. Este concepto surgió por primera vez con la multiprogramación, donde, como se vio en el Capítulo 1, se puede ejecutar más de un programa simultáneamente con el fin de aprovechar al máximo los recursos de la computadora.

> Un **proceso** es un programa en ejecución junto con el entorno asociado (registros, variables, etc.).

En la Figura 4.1 puede verse gráficamente el concepto de proceso con su entorno (programa en ejecución, datos en variables y pila, etc.).

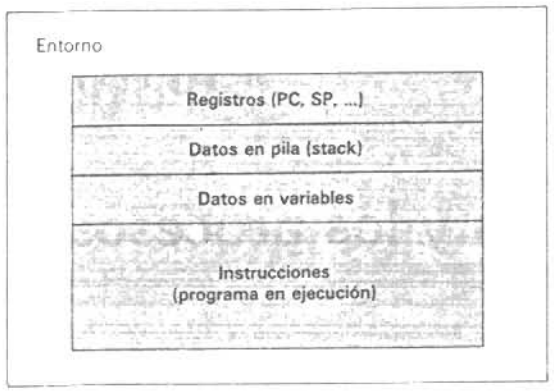


Figura 4.1.
Concepto de proceso.

Ya se ha indicado que el corazón de un sistema operativo es el **núcleo**, un programa de control que reacciona ante cualquier interrupción de eventos externos y que da servicio a los procesos, creándolos, terminándolos y respondiendo a cualquier petición de servicio por parte de los mismos.

4.2.1. Modelo

La diferencia entre un programa (conjunto de instrucciones) y un proceso (instrucciones ejecutándose) es obvia, pero crucial para entender el funcionamiento de los sistemas operativos.

Imaginemos un mecánico de automóviles en un taller donde se reparan vehículos con averías complejas en las que se hace necesario consultar el manual de reparaciones de cada modelo, que contiene instrucciones para todas las posibles averías. Además, se permiten reparaciones rápidas a las que se les da mayor prioridad que a las mencionadas anteriormente. Existe en el taller un almacén de repuestos de todo tipo y herramientas suficientes para las reparaciones.

Comparando esta situación con una computadora, se pueden establecer las siguientes analogías:

- El mecánico sería el procesador que va a realizar el trabajo.
- El manual de cada reparación sería el programa.
- Las herramientas serían los recursos disponibles.
- Las piezas de recambio serían los datos.
- La actividad de utilizar las herramientas para desmontar las piezas defectuosas sustituyéndolas por otras nuevas siguiendo las instrucciones del manual equivaldría al proceso.

Supongamos que en un determinado momento el mecánico está realizando una reparación compleja (de las que llevan tiempo) y aparece un vehículo que solicita una reparación de las rápidas (ha aparecido una interrupción). El mecánico suspende momentáneamente la reparación compleja anotando en qué situación se queda dicha reparación y qué operación estaba realizando en este momento (guarda el estado del proceso). Asimismo, sustituye el manual que estaba utilizando por el de la reparación rápida que se dispone a realizar (cambio de programa). Comienza la nueva reparación (cambio de proceso), en la que las herramientas no serán las mismas que antes (distintos recursos); las indicaciones del usuario, las piezas de repuesto (datos) y las indicaciones del manual (programa) llevarán a feliz término la reparación para que el mecánico vuelva a continuación a la reparación inicial.

Con este ejemplo se desea resaltar que un proceso es una actividad que se apoya en datos, recursos, un estado en cada momento y un programa.

4.2.2. El bloque de control del proceso (PCB)

Un proceso se representa, desde el punto de vista del sistema operativo, por un conjunto de datos donde se incluyen el estado en cada momento, recursos utilizados, registros, etc., denominado **Bloque de Control del Proceso** (PCB).

Los objetivos que se pretenden cubrir con el bloque de control del proceso son los siguientes:

- Localización de la información sobre el proceso por parte del sistema operativo.
- Mantener registrados los datos del proceso en caso de tener que suspender temporalmente su ejecución o reanudarla.

En general, la información contenida en el bloque de control es la siguiente:

- **Estado del proceso.** Información relativa al contenido del contador de programa *(Program Counter,* PC), estado del procesador en cuanto a prioridad del proceso, modo de ejecución, etc., y por último el estado de los registros internos de la computadora.
- **Estadísticas de tiempo y ocupación de recursos** para la gestión de la planificación del procesador.
- **Ocupación de memoria interna y externa** para el intercambio (swapping).
- **Recursos en uso** (normalmente unidades de entrada/salida).
- **Archivos en uso.**
- **Privilegios.**

Estas informaciones se encuentran en memoria principal o en disco y se accede a ellas en los momentos en que se hace necesaria su actualización o consulta. Los datos relativos al estado del proceso siempre se encuentran en memoria principal.

De igual forma existe un **Bloque de Control del Sistema** (SCB), con unos objetivos globales similares al anterior y entre los que se encuentra el enlazado de los bloques de control de los procesos existentes en el sistema (Figura 4.2).

Trataremos de ver a continuación cómo se realiza el cambio de un proceso a otro, para lo cual supondremos que estamos en una computadora con un solo procesador (sólo un proceso puede estar ejecutándose en cada momento), y existen varios procesos activos compitiendo por el acceso al procesador (se está ejecutando un proceso A y el núcleo del sistema operativo decide que debe ejecutarse en un instante dado otro proceso B). Suponemos que los programas de los procesos A y B están ambos en memoria principal.

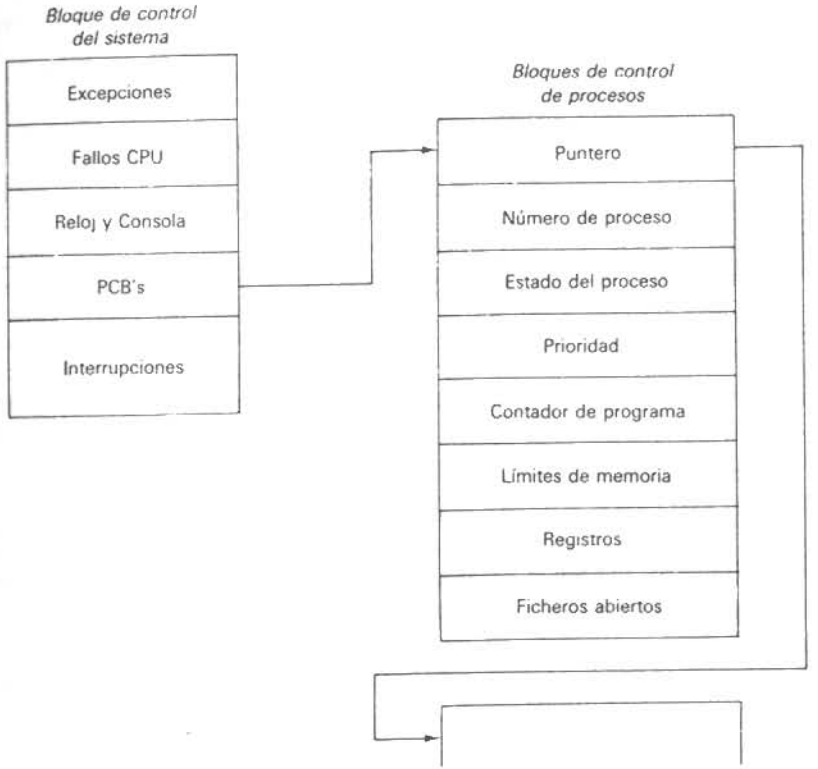


Figura 4.2. Bloques de control de procesos y del sistema.

Las acciones que realiza el sistema operativo para cambiar el proceso A por el B se denominan **cambio de proceso,** y son las siguientes (Figura 4.3):

- Deja de ejecutar el proceso en curso (A), cediéndose el control al núcleo del sistema operativo, y aparece lo que se denomina un **cambio de contexto** pasando del modo usuario al modo supervisor. Antes de realizarse el cambio de contexto se salva el estado del proceso A para su posterior vuelta al punto donde fue interrumpido.
- El núcleo estudia si el proceso B está preparado para su ejecución y, si es así, realiza el cambio de contexto correspondiente pasando del modo supervisor al modo usuario. A continuación repone el estado del proceso B (en caso de haber sido interrumpido con anterioridad) y, por último, pone en ejecución el proceso B.

El cambio de contexto se producirá en caso de ejecución de una instrucción privilegiada, una llamada al sistema operativo o una interrupción, es decir, siempre que se requiera la atención de algún servicio del sistema operativo.

En el cambio de contexto, el núcleo salva el estado del proceso que se estaba ejecutando en su bloque de control y restaura el proceso que va a ejecutarse a partir de los datos almacenados en su bloque de control.

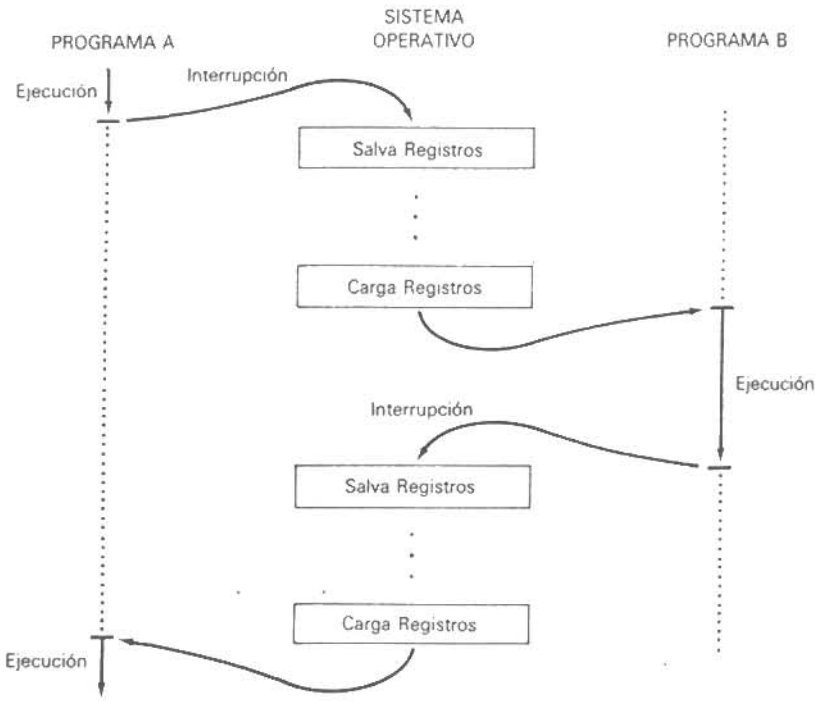


Figura 4.3. Cambio de proceso.

4.2.3. Estado de los procesos

Los bloques de control de los procesos se almacenan en colas, cada una de las cuales representa un estado particular de los procesos, existiendo en cada bloque, entre otras informaciones, tantos campos como colas en las que el proceso se pueda situar, para que a partir de ellos se indique la cola en que se encuentra (Figura 4.4).

Los estados de los procesos son internos del sistema operativo y transparentes al usuario. Para éste, su proceso estará siempre en ejecución independientemente del estado en que se encuentre internamente en el sistema.

Los estados de los procesos se pueden dividir en dos tipos: activos e inactivos.

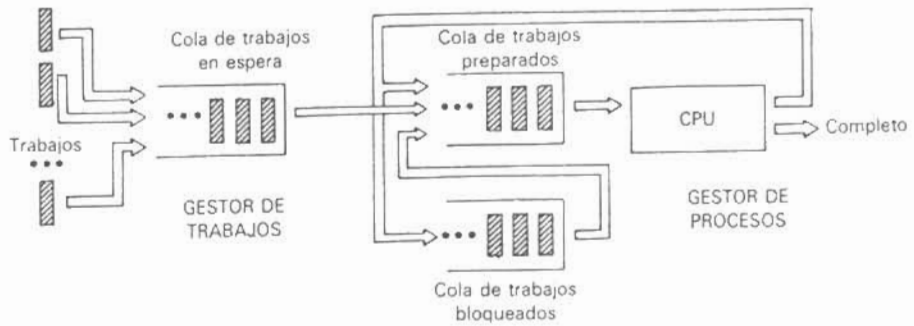


Figura 4.4. Colas de estado.

- **Estados activos**

Son aquellos que compiten con el procesador o están en condiciones de hacerlo. Se dividen en (Figura 4.5):

- **Ejecución.** Estado en el que se encuentra un proceso cuando tiene el control del procesador. En un sistema monoprocesador este estado sólo lo puede tener un proceso.
- **Preparado.** Aquellos procesos que están dispuestos para ser ejecutados, pero no están en ejecución por alguna causa (interrupción, haber entrado en cola estando otro proceso en ejecución, etc.).
- **Bloqueado.** Son los procesos que no pueden ejecutarse de momento por necesitar algún recurso no disponible (generalmente recursos de entrada/salida).

- **Estados inactivos**

Son aquellos que no pueden competir por el procesador, pero que pueden volver a hacerlo por medio de ciertas operaciones. En estos estados se mantiene el bloque de control del proceso aparcado hasta que vuelva a ser activado. Se trata de procesos que no han terminado su trabajo por causas que lo han impedido (por ejemplo, avería en un dispositivo de entrada/salida) y que pueden volver a activarse desde el punto en que se quedaron sin que tengan que volver a ejecutarse desde el principio (se puede pensar en procesos cuya duración es larga). Son de dos tipos (Figura 4.5):

- **Suspendido bloqueado.** Es el proceso que fue suspendido en espera de un evento, sin que hayan desaparecido las causas de su bloqueo.
- **Suspendido preparado.** Es el proceso que ha sido suspendido, pero no tiene causa para estar bloqueado.

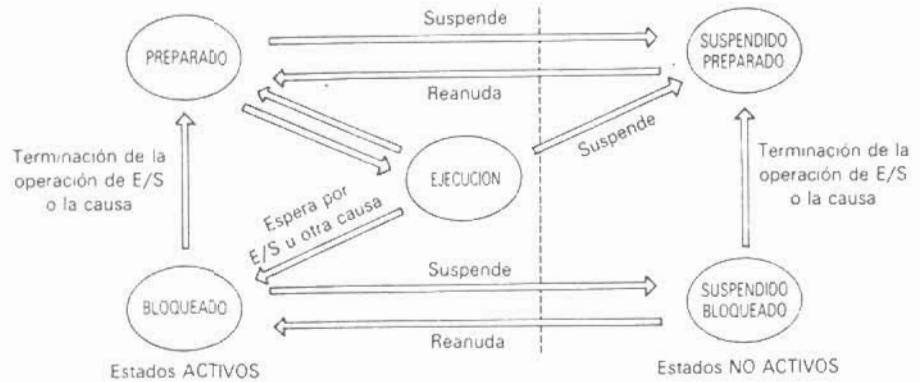


Figura 4.5. Estados de un proceso y sus transiciones.

4.2.4. Transiciones de estado

Todo proceso a lo largo de su existencia puede cambiar de estado varias veces. Cada uno de estos cambios se denomina **transición de estado.** Estas transiciones son las siguientes (Figura 4.5):

- **Comienzo de la ejecución.** Todo proceso comienza al ser dada la orden de ejecución del programa insertándose en la cola de preparados. El encolamiento dependerá de la política de gestión de dicha cola.
- **Paso a estado de ejecución.** Cuando el procesador se encuentre inactivo y en la cola de preparados exista algún proceso en espera de ser ejecutado, se pondrá en ejecución el primero de ellos.
- **Paso a estado bloqueado.** Un proceso que se encuentre en ejecución y que solicite una operación a un dispositivo externo (unidad de entrada/salida), teniendo que esperar a que dicha operación finalice, será pasado de estado de ejecución a estado bloqueado insertándose su PCB en la cola correspondiente de bloqueados. A partir de ese momento el procesador pone en ejecución el siguiente proceso, que será el primero de la cola de preparados.
- **Paso a estado preparado.** Este paso puede ser producido por alguna de las siguientes causas:
  - Orden de ejecución de un programa, con lo cual, como ya hemos dicho, el proceso pasa a la cola de preparados.
  - Si un proceso está en estado bloqueado por causa de una operación de entrada/salida y ésta finaliza, pasará de la cola de bloqueados a la de preparados.
  - Si un proceso está en ejecución y aparece una interrupción que fuerza al sistema operativo a ejecutar otro proceso, el primero pasará al estado preparado y su PCB a la cola de preparados.

– Activación. Un proceso suspendido previamente sin estar bloqueado pasará al estado preparado al ser activado nuevamente.

- **Paso a estado suspendido bloqueado.** Si un proceso está bloqueado y el sistema operativo recibe la orden de suspenderlo, su PCB entrará en la cola de procesos suspendidos bloqueados.
- **Paso a estado suspendido preparado.** Este paso se puede producir bajo tres circunstancias:
  – Suspensión de un proceso preparado pasando éste de la cola de procesos preparados a la de suspendidos preparados.
  – Suspensión de un proceso en ejecución, con lo cual el proceso pasa a la cola de suspendidos preparados.
  – Desbloqueo de un proceso suspendido bloqueado por desaparecer la causa que impedía el ser activado de nuevo.

4.2.5. Operaciones sobre procesos

Los sistemas operativos actuales poseen una serie de funciones cuyo objetivo es el de la manipulación de los procesos. En general, las operaciones que se pueden hacer sobre un proceso son las siguientes:

- **Crear el proceso.** Se produce con la orden de ejecución del programa y suele necesitar varios argumentos, como el nombre y la prioridad del proceso. Aparece en este momento el PCB, que será insertado en la cola de procesos preparados (Figura 4.6).

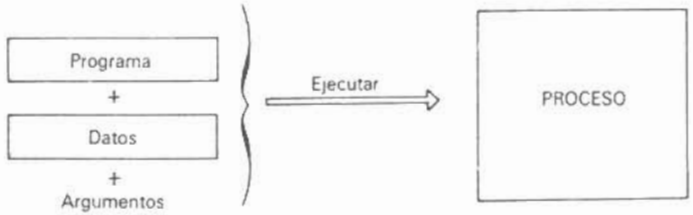


Figura 4.6. Creación de un proceso.

La creación de un proceso puede ser de dos tipos:

– **Jerárquica.** En ella, cada proceso que se crea es hijo del proceso creador y hereda el entorno de ejecución de su padre. El primer proceso que ejecuta un usuario será hijo del intérprete de comandos con el que interactúa (Figura 4.7).
– **No jerárquica.** Cada proceso creado por otro proceso se ejecuta independientemente de su creador con un entorno diferente. Es un tipo de creación que no suele darse en los sistemas operativos actuales.

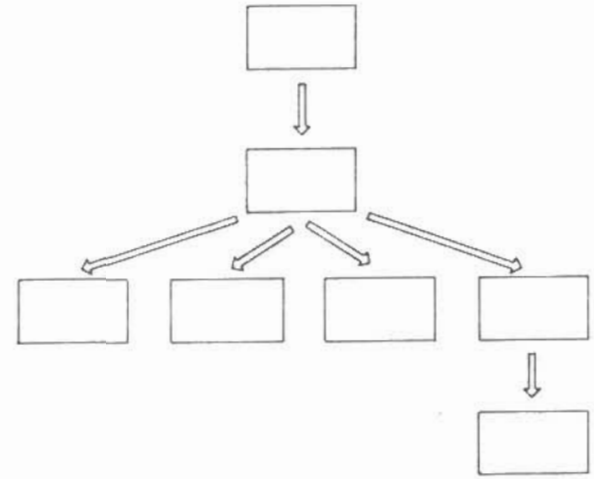


Figura 4.7. Jerarquía de procesos.

- **Destruir un proceso.** Se trata de la orden de eliminación del proceso con la cual el sistema operativo destruye su PCB.
- **Suspender un proceso.** Es una operación de alta prioridad que paraliza un proceso que puede ser reanudado posteriormente. Suele utilizarse en ocasiones de mal funcionamiento o sobrecarga del sistema.
- **Reanudar un proceso.** Trata de activar un proceso que ha sido previamente suspendido.
- **Cambiar la prioridad de un proceso.**
- **Temporizar la ejecución de un proceso.** Hace que un determinado proceso se ejecute cada cierto tiempo (segundos, minutos, horas...) por etapas o de una sola vez, pero transcurrido un periodo de tiempo fijo.
- **Despertar un proceso.** Es una forma de desbloquear un proceso que habrá sido bloqueado previamente por temporización o cualquier otra causa.

4.2.6. Prioridades

En general, todo proceso por sus características e importancia lleva aparejadas unas determinadas necesidades de ejecución en cuanto a urgencia y asignación de recursos. Esto hace que los distintos procesos presentes en un sistema no accedan de igual forma y con igual frecuencia al procesador debido a la prioridad que cada uno de ellos tiene asignada.

Las prioridades según los sistemas operativos se pueden clasificar del siguiente modo:

- **Asignadas por el sistema operativo.** Se trata de prioridades que son asignadas a un proceso en el momento de comenzar su ejecución y dependen fundamentalmente de los privilegios de su propietario y del modo de ejecución.

- **Asignadas por el propietario.** En este caso es el propio usuario el que asigna a cada proceso la prioridad con que éste debe ejecutarse. Esta modalidad de asignación de prioridades es muy utilizada en sistemas de tiempo real, ya que algunos de sus procesos necesitan atender rápidamente algún evento sin que tengan que interrumpirse.

Otra clasificación de prioridades atendiendo a la posibilidad de variación de las mismas es la siguiente:

- **Estáticas.** Son aquellas prioridades que no pueden ser modificadas durante la ejecución del proceso. Pueden ser utilizadas en sistemas de tiempo compartido, pero no en los de tiempo real
- **Dinámicas.** La prioridad de un proceso puede ser modificada con el fin de atender cualquier evento que se produzca.

4.2.7. Tipos de procesos

Un proceso puede clasificarse en dos grandes grupos según el uso que vaya a tener y la forma como se haya construido el código ejecutable de su programa. Estos grupos son:

- **Reutilizables.** Son aquellos que pueden cambiar los datos que utilizan, pero si vuelven a ejecutarse necesitan comenzar en su estado inicial y procesar nuevos datos. Es el caso de los programas normales de usuario.
- **Reentrantes.** Se caracterizan por no tener asociados datos, es decir, sólo tienen código puro. Los datos que utilizan se encuentran en la pila o en registros internos y no pueden ser modificados durante su uso. Este es el caso de programas compartidos por varios usuarios a la vez, como es el de los editores cuyo código se encuentra una sola vez en memoria, siendo utilizados por usuarios cuyos datos particulares estarán en una zona del disco asignada a cada uno de ellos.

Otra clasificación de procesos según la capacidad que tienen de acceso al procesador y al resto de recursos es:

- **Apropiativos.** Son aquellos que al tener asignado un recurso no permiten que otro proceso pueda acceder a él hasta que hayan terminado de utilizarlo.
- **No apropiativos.** Permiten que otros procesos puedan acceder a un recurso que esté siendo utilizado por ellos.

Por último, los procesos según su forma de ejecución se clasifican en:

- **Residentes.** Son los que permanecen en memoria todo el tiempo que dure su ejecución.
- **Intercambiables.** Son aquellos que pueden ser llevados de memoria principal a disco mientras estén bloqueados. La memoria principal liberada por ellos puede ser utilizada por otro proceso que en ese momento la necesite.

4.2.8. Excepciones

A lo largo de la ejecución de un proceso pueden aparecer una serie de irregularidades o fallos que de alguna forma un sistema operativo debe tratar de controlar y en su caso corregir. Estos pueden ser de distinta naturaleza y afectar en mayor o menor medida al proceso; entre ellos podemos citar:

- Fallos hardware.
- Fallos software.
- Entrada de datos incorrectos.
- Eventos anómalos.
- Etc.

Para atender a este tipo de eventos los sistemas operativos incorporan lo que se denomina gestor de excepciones, cuya misión es la de tratar el software que controla este tipo de eventos o excepciones.

Según la gravedad de los eventos que pueden presentarse, se establecen tres categorías de errores:

- **Catastróficos.** Son aquellos que imposibilitan el funcionamiento del sistema y no hay modo de recuperarlo; por ejemplo, un fallo en la tensión de alimentación.

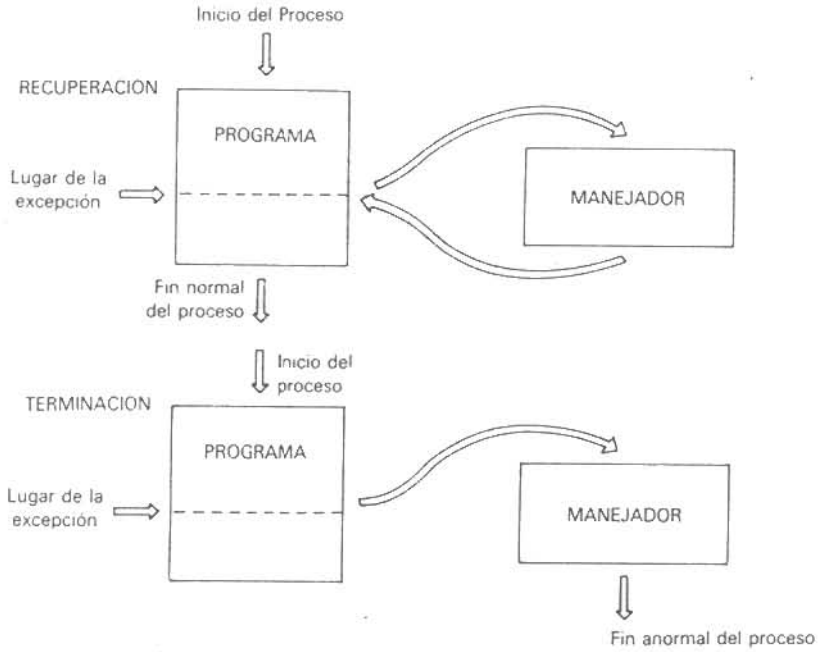


Figura 4.8. Modelos de gestor de excepciones.

- **No recuperables.** Son los que sin afectar al sistema, hacen que el proceso no pueda continuar su ejecución; por ejemplo, la aparición de una división por 0.
- **Recuperables.** Son los que con ciertos ajustes permiten que el proceso continúe su ejecución normal; por ejemplo, datos con formato indebido.

El tratamiento de una excepción puede seguir dos modelos diferentes (Figura 4.8):

– Tratamiento de la excepción y continuación del proceso.
– Tratamiento de la excepción y finalización del proceso.

CUESTIONES

1. Definir el concepto de proceso.
2. Indicar qué se entiende por bloque de control de un proceso y qué misión realiza.
3. ¿Cuál es la información contenida en el bloque de control de un proceso?
4. ¿Qué es un cambio de proceso? ¿Qué acciones realiza el sistema operativo para llevarlo a cabo?
5. ¿A qué se denomina estado activo de un proceso? ¿En qué tres grupos se divide?
6. ¿Qué es una transición de estado?
7. Comentar alguna transición de estado de un proceso.
8. ¿Cuáles son las operaciones que en general pueden hacerse sobre un proceso para su manipulación a través del sistema operativo?
9. Comentar los dos tipos de creación de un archivo que existen.
10. Qué son las prioridades y cómo se clasifican.
11. Según el uso a que se destine y la forma de construcción de su código ejecutable, ¿cómo se clasifica un proceso?
12. ¿Qué es una excepción? ¿Qué incorpora un sistema operativo para su control?

CAPITULO 5

Planificación del procesador

5.1. INTRODUCCION

En este capitulo se estudian las distintas políticas y mecanismos más comunes que poseen los sistemas operativos actuales para realizar la gestión del procesador que se conoce con el nombre de **planificación**, cuyo objetivo principal es el de dar un buen servicio a todos los procesos que existan en un momento dado en el sistema.

En general, se distinguen varios niveles de planificación, como se refleja en la Figura 5.1.

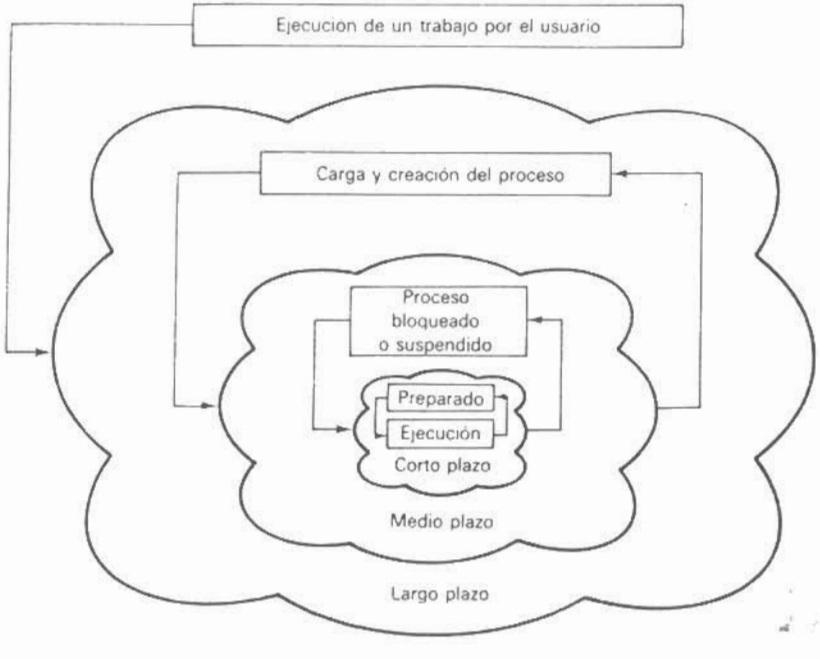


Figura 5.1. Niveles de planificación del procesador.

- **Planificación a largo plazo (planificador de trabajos).** Decide cuál será el próximo trabajo que se va a ejecutar. Este nivel sólo existe en los sistemas de proceso por lotes, donde la decisión se basa en las necesidades de recursos y su disponibilidad. En los sistemas de tiempo compartido tiene como única misión cargar los programas que se desean ejecutar en memoria. Este nivel es, por tanto, el encargado de crear los procesos.
- **Planificación a medio plazo (planificador de swapping).** Decide si un proceso que está en ejecución en estado bloqueado o suspendido debe ser extraído de la memoria temporalmente. Posteriormente, cuando el sistema se encuentre más descargado, devolverá dicho proceso a la memoria y al estado de ejecución. Esta técnica se conoce con el nombre de *swapping* y será estudiada al tratar la gestión de la memoria. Sólo existe en sistemas de tiempo compartido y en aquellos que tienen gestión de memoria virtual. Este nivel, por tanto, gestiona los procesos suspendidos en espera de algún recurso no disponible en el momento de la suspensión.
- **Planificación a corto plazo (planificador del procesador).** Es el encargado de decidir cómo y cuándo tendrá acceso al procesador un proceso que está preparado para utilizarlo. Por tanto, lleva a cabo las funciones de la multiprogramación, estando siempre residente en memoria y ejecutándose con mucha frecuencia; por ello, debe ser de ejecución muy rápida. En este nivel, que es el que vamos a estudiar en este capítulo, es donde se debe dar un buen servicio a los procesos interactivos para que el usuario no perciba, o lo haga en pequeño grado, que está compitiendo por el procesador junto con otros usuarios.

  Sólo queda añadir que toda política de gestión debe ser justa al tiempo de ofrecer un rendimiento del sistema lo más aceptable posible.

5.2. OBJETIVOS

Las políticas de planificación intentan cubrir los siguientes objetivos:

- **Justicia.** La política debe ser lo más justa posible con todo tipo de procesos, sin favorecer a unos y perjudicar a otros.
- **Máxima capacidad de ejecución.** Debe dar un servicio aceptable para que todos los trabajos se realicen lo más rápidamente posible. Esto se logra disminuyendo el número de cambios de proceso.
- **Máximo número de usuarios interactivos.** En los sistemas de tiempo compartido se tratará de que puedan estar trabajando el mayor número de usuarios simultáneamente.
- **Predecibilidad.** La política de planificación debe concebirse de tal forma que en todo momento pueda saberse cómo será su ejecución.
- **Minimización de la sobrecarga.** La computadora debe tener poca sobrecarga ya que ésta incide directamente sobre el rendimiento final del sistema: a menor sobrecarga, mayor velocidad de proceso. Por ello, los cambios de contexto deben minimizarse.

- **Equilibrio en el uso de recursos.** Para obtener un buen rendimiento en el uso de los recursos y que éstos estén ocupados equitativamente el mayor tiempo posible.
- **Seguridad de las prioridades.** Si un proceso tiene mayor prioridad que otro, éste debe ejecutarse más rápidamente.

Los objetivos enunciados pueden entrar en ocasiones en contradicción; por ello es necesario llegar a una situación de compromiso entre todos los objetivos para conseguir del sistema operativo un buen rendimiento y un buen servicio.

5.3. CRITERIOS

Los criterios que se deben tener en cuenta a la hora de elegir o diseñar un algoritmo de planificación son los siguientes:

- **Tiempo de respuesta.** Velocidad con que el ordenador da respuesta a una petición. Depende mucho de la velocidad de los dispositivos de entrada/salida.
- **Tiempo de servicio.** Es el tiempo que tarda en ejecutarse un proceso, donde se incluye el tiempo de carga del programa en memoria, el tiempo de espera en la cola de procesos preparados, el tiempo de ejecución en el procesador y el tiempo consumido en operaciones de entrada/salida.
- **Tiempo de ejecución.** Es idéntico al tiempo de servicio menos el tiempo de espera en la cola de procesos preparados; es decir, es el tiempo teórico que necesitaría el proceso para ser ejecutado si fuera el único presente en el sistema.
- **Tiempo de procesador.** Es el tiempo que un proceso está utilizando el procesador sin contar el tiempo que se encuentra bloqueado por operaciones de entrada/salida.
- **Tiempo de espera.** Es el tiempo en que los procesos están activos pero sin ser ejecutados, es decir, los tiempos de espera en las distintas colas.
- **Eficiencia.** Se refiere a la utilización del recurso más caro en un sistema, el procesador, que debe estar el mayor tiempo posible ocupado para lograr así un gran rendimiento.
- **Rendimiento.** Es el número de trabajos o procesos realizados por unidad de tiempo, que debe ser lo mayor posible.

Ahora bien, ¿qué algoritmo de planificación se debe elegir para un sistema determinado? Será misión del diseñador del sistema operativo la elección de los mecanismos apropiados para que la política elegida partiendo de los criterios anteriores sea satisfactoria y ofrezca un alto rendimiento global.

5.4. MEDIDAS

Para estudiar el comportamiento de las distintas políticas de planificación, definiremos dos medidas relacionadas entre sí que nos indiquen cómo estamos tratando un proceso concreto.

Consideremos **t** como el tiempo que un proceso **P** necesita estar en ejecución para llevar a cabo su trabajo, t_i el instante en que el usuario da la orden de ejecución del proceso y t_f el instante en que el proceso termina su ejecución. En función de estos datos, tendremos las siguientes medidas para cada proceso:

- **Tiempo de servicio** (T): $T = t_f - t_i$
- **Tiempo de espera** (E): $E = T - t$

A partir de los dos valores anteriores, podemos establecer una relación que nos permite evaluar la actuación de la política establecida en lo que se denomina **indice de servicio** (I).

$$I = \frac{t}{T}$$

Este índice representa el tanto por uno de tiempo que el proceso está en ejecución respecto al tiempo de vida del mismo en el sistema.

En caso de que sólo exista un proceso ejecutándose en el sistema, según el valor del índice de servicio, podemos decir que:

- Cuando I sea próximo a la unidad, el proceso está limitado por proceso.
- Si I tiene un valor bajo próximo a 0, el proceso estará limitado por entrada/salida.

En los casos más usuales en los que existe más de un proceso en el sistema, no podemos hacer las consideraciones anteriores puesto que puede desvirtuarse el verdadero comportamiento del sistema. Por este motivo se establecen las mismas medidas, pero con valores medios obtenidos al considerar el conjunto de procesos presentes. Estas serán las medidas que nos reflejarán el verdadero comportamiento del sistema. Las medidas a las que nos referimos son:

- **Tiempo medio de servicio.**
- **Tiempo medio de espera.**
- **Eficiencia** (indice medio de servicio).

Además de las anteriores, en la planificación del procesador suelen emplearse otras dos medidas de interés:

- **Tiempo del núcleo.** Es el tiempo consumido por el núcleo del sistema operativo para tomar las decisiones de planificación del procesador, donde se incluyen los tiempos de cambio de contexto y de proceso.
- **Tiempo de inactividad (Idle).** Es el tiempo consumido cuando la cola de procesos preparados está vacía y por tanto no puede realizarse ningún trabajo productivo.

5.5. ALGORITMOS DE PLANIFICACION

Como ya hemos visto, el planificador del procesador tiene como misión la asignación del mismo a los procesos que están en la cola de procesos preparados. Esta cola es alimentada desde dos puntos distintos:

- Cada vez que un usuario inicie la ejecución de un programa, el planificador a largo plazo recibe la orden de ejecución, crea el proceso y lo pasa al planificador a corto plazo, colocándose en la cola de procesos preparados.
- Cuando un proceso deja de estar en estado de ejecución y no existen causas para su bloqueo, o deja de estar bloqueado, pasa nuevamente a la cola de procesos preparados.

Por otro lado, cuando un proceso termina su ejecución, deja de existir para el planificador.

Las políticas de planificación se agrupan en:

- **Políticas apropiativas.** Son las que producen un cambio de proceso con cada cambio de contexto; es decir, el proceso que está haciendo uso del procesador puede ser temporalmente suspendido y permitir que otro proceso se apropie del procesador. Se utilizan en sistemas operativos con tiempo compartido y tiempo real.
- **Políticas no apropiativas.** Son aquellas en las que un proceso no abandona nunca el procesador desde su comienzo hasta su fin. Se utilizan en sistemas de proceso por lotes.

En los sistemas operativos comerciales existen diversas políticas de planificación, de las cuales veremos algunas a continuación, puestas en práctica a través de algoritmos que en ningún caso son perfectos. Recordemos que los objetivos y criterios pueden ser contradictorios entre sí, de manera que si favorecemos a un tipo de procesos, normalmente perjudicaremos a otros.

Para el estudio de las diferentes políticas nos basaremos en la situación de un grupo de procesos existentes en un sistema, cuyos datos se encuentran en la Tabla 5.1 y su representación gráfica en la Figura 5.2.

Tabla 5.1

Nombre proceso	Instante llegada	Tiempo ejecución	Prioridad
A	0	3	0
B	1	5	1
C	4	2	0
D	5	6	2
E	8	4	1

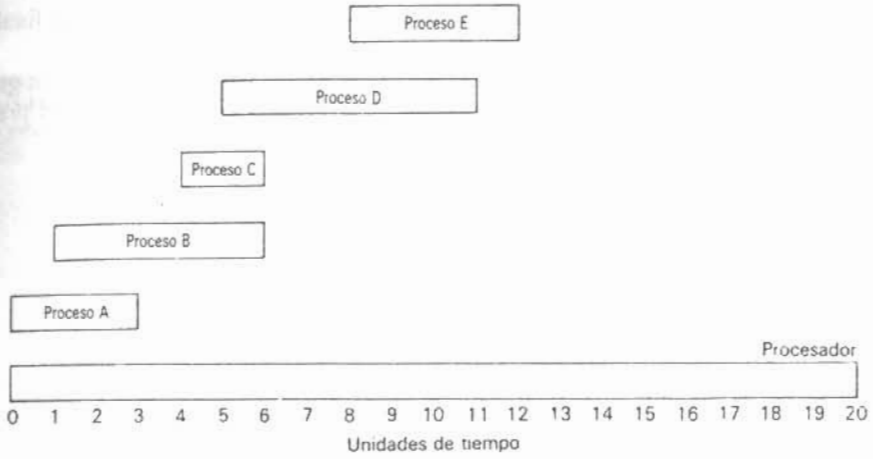


Figura 5.2. Representación gráfica del ejemplo.

Además de figurar el instante de entrada en el sistema, también se ha representado el tiempo de servicio de cada proceso, es decir, el tiempo que necesita cada uno de ellos para ejecutarse si fuera el único presente en el sistema. Por otro lado, supondremos que estos procesos no necesitan realizar operaciones de entrada/salida (aunque esto sea una utopía) para facilitar el estudio.

La unidad de tiempo que utilizamos en el ejemplo es abstracta, es decir, puede ser cualquier unidad (milisegundos, segundos, etc.). No debemos olvidar que el valor más representativo del comportamiento de una política de planificación es el índice medio de servicio o eficiencia que es un valor adimensional y suele expresarse en tanto por ciento (%).

5.5.1. Primero en llegar, primero en ser servido (FCFS)

En esta política de planificación FCFS *(First Come, First Served)*, el procesador ejecuta cada proceso hasta que termina; por tanto, los procesos que entren en cola de procesos preparados permanecerán encolados en el orden en que lleguen hasta que les toque su ejecución (Figura 5.3). Este método se conoce también como «primero en entrar, primero en salir» *(First Input, First Output - FIFO)*.

Figura 5.3. Planificación FCFS.

Se trata de una política muy simple y sencilla de llevar a la práctica, pero muy pobre en cuanto a su comportamiento. En la Tabla 5.2 se muestran los datos de los procesos para esta planificación, y en la Figura 5.4 puede verse cómo despacha FCFS los procesos del ejemplo propuesto, donde las partes sombreadas indican el tiempo que ha estado cada proceso en espera de acceder al procesador.

Tabla 5.2

Nombre proceso	Instante llegada	Tiempo ejecución	Instante finalización	*T*	*E*	*I*
A	0	3	3	3	0	1.00
B	1	5	8	7	2	0.71
C	4	2	10	6	4	0.33
D	5	6	16	11	5	0.54
E	8	4	20	12	8	0.33
Media				7.8	3.8	0.58

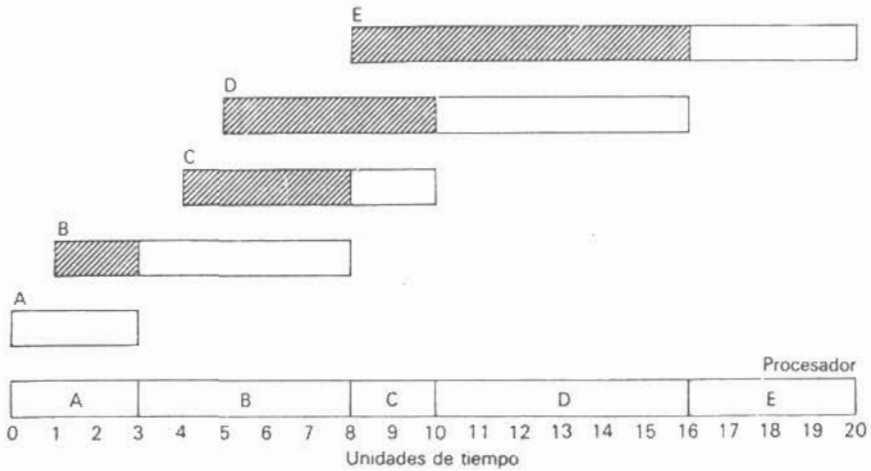


Figura 5.4. Politica FCFS en el ejemplo.

Podemos observar que el indice de servicio mejora cuanto más largos son los procesos. Es decir, los procesos cortos que entren en el sistema después de uno o varios largos tendrán que esperar un periodo de tiempo relativamente largo hasta su ejecución.

La cantidad de tiempo de espera de cada proceso depende del número de procesos que se encuentren en cola en el momento de su petición de ejecución y del tiempo que cada uno de ellos tenga en uso al procesador, y es independiente de las necesidades de ejecución del propio proceso.

Las características de esta política son las siguientes:

- No es apropiativa.
- Es justa, aunque los procesos largos hacen esperar mucho a los cortos.
- Es una política predecible.
- El tiempo medio de servicio es muy variable en función del número de procesos y su duración.

5.5.2. Round-Robin (RR)

Esta política, cuya traducción podría ser *asignación cíclica* o *planificación en rueda*, es una mejora de la FCFS. Trata de ser más justa en cuanto a la respuesta tanto de los procesos cortos como de los largos.

Consiste en conceder a cada proceso en ejecución un determinado período de tiempo q *(quantum)*, transcurrido el cual, si el proceso no ha terminado, se le devuelve al final de la cola de procesos preparados, concediéndose el procesador al siguiente proceso por su correspondiente quantum (Figura 5.5).

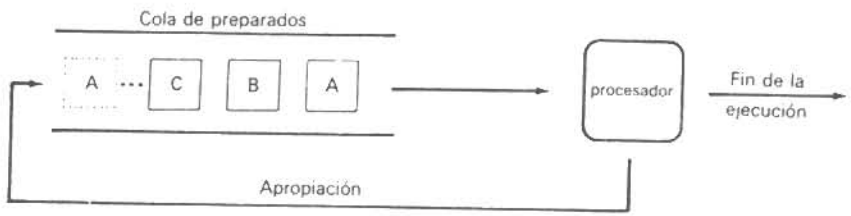


Figura 5.5. Planificación Round-Robin.

Esta interrupción periódica continúa hasta que el proceso termine su ejecución, formando una rueda de procesos que serán ejecutados cíclicamente hasta que terminen.

La gestión de la cola de procesos preparados se puede realizar de muy diversas maneras, siendo las más comunes la FIFO o por prioridades, donde los procesos se ordenan según su prioridad.

Variando el parámetro q lograremos tener diferentes comportamientos de esta política, de tal forma que si q es mayor que el tiempo que necesita para su ejecución el proceso más largo, se convertiría en una política FCFS. En cambio, si se aproxima a 0, la sobrecarga del sistema será muy grande puesto que la mayor parte del tiempo se consumiría en cambios de contexto.

Los valores de q varían entre 10 y 100 milisegundos, siendo recomendable que el 80 por 100 de los tiempos de respuesta de los procesos sean inferiores al quantum.

En la Figura 5.6 puede verse esta planificación para el ejemplo propuesto, con un valor de quantum $q = 1$, con las siguientes condiciones:

- Si un proceso finaliza durante su quantum, inmediatamente se le concede el procesador a otro proceso, al que se le asigna el quantum completo.

- Al crearse un proceso y pasar a la lista de procesos preparados, se coloca al final de la lista.
- Si un proceso comienza su ejecución (creación) en el mismo momento en que un quantum finaliza, se supondrá que dicho proceso ha llegado a la cola de procesos preparados antes de la finalización del mencionado quantum.

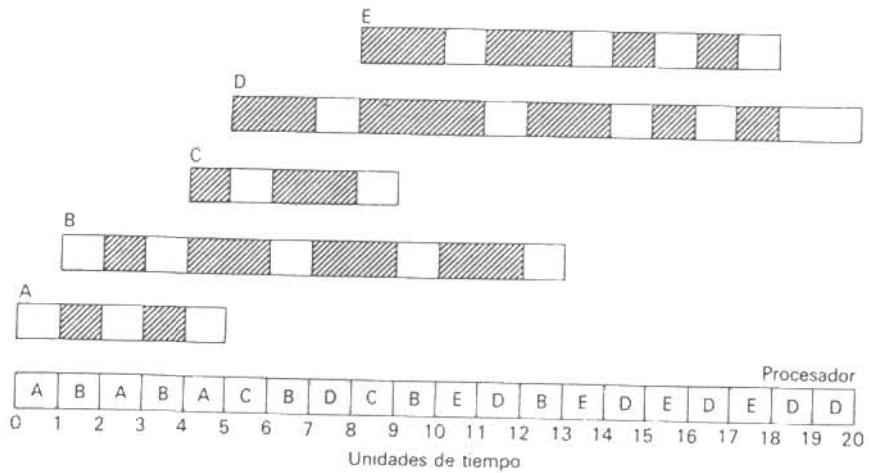


Figura 5.6. Política Round-Robin para $q = 1$.

La Tabla 5.3 representa los valores del ejemplo para $q = 1$.

Tabla 5.3

Nombre proceso	Instante llegada	Tiempo ejecución	Instante finalización	T	E	I
A	0	3	5	5	2	0.60
B	1	5	13	12	7	0.42
C	4	2	9	5	3	0.40
D	5	6	20	15	9	0.40
E	8	4	18	10	6	0.40
Media				9.4	5.4	0.44

A continuación se repite la planificación RR para un valor de quantum 3 y con las mismas condiciones que en el caso anterior. La Tabla 5.4 y la Figura 5.7 representan los datos y el gráfico, respectivamente, del ejemplo propuesto para este valor.

Tabla 5.4

Nombre proceso	Instrucción llegada	Tiempo ejecución	Instante finalización	*T*	*E*	*I*
A	0	3	3	3	0	1.00
B	1	5	13	12	7	0.42
C	4	2	8	4	2	0.50
D	5	6	19	14	8	0.43
E	8	4	20	12	8	0.33
Media				9.0	5.0	0.54

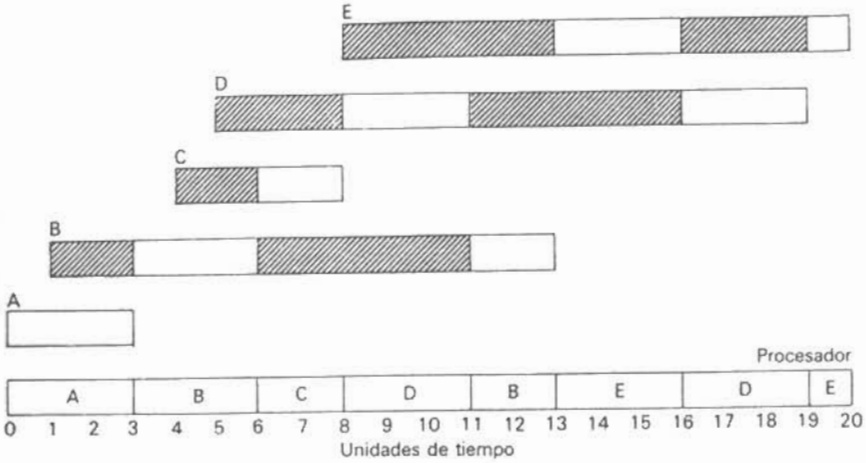


Figura 5.7. Politica Round-Robin para $q = 3$.

En este caso el tiempo de servicio T se mantiene prácticamente constante. Se puede observar que el tiempo de espera E crece de acuerdo con el tiempo de ejecución de cada proceso.

Las características de esta politica de planificación son:

- Baja sobrecarga si el cambio de contexto es eficiente y los procesos siempre están en la memoria principal.
- El tamaño óptimo del quantum depende de:
  - El tipo de sistema.
  - Las cargas que vaya a soportar el sistema.
  - El número de procesos en el sistema y su tipo.
  - Es la política más utilizada para tiempo compartido.
  - Ofrece un índice de servicio uniforme para todos los procesos.
  - Es una política apropiativa.

5.5.3. El siguiente proceso, el más corto (SJN)

Hemos visto que la política RR mantiene constante el índice de servicio de los procesos cortos basándose en la apropiación del procesador. El método SJN *(Shortest Job Next)* es una política de planificación no apropiativa que trata de cubrir los mismos objetivos que la RR.

Esta política toma de la cola de procesos preparados el que necesite menos tiempo de ejecución para realizar su trabajo. Para ello debe saber el tiempo de procesador que necesita cada proceso, lo cual es tarea nada fácil, pero posible a través de diversos métodos como pueden ser la información suministrada por el propio usuario, por el propio programa, basándose en la historia anterior (heurística), etc.

El tiempo de servicio T en esta política es bueno para los procesos cortos, saliendo perjudicados los procesos largos.

En la Figura 5.8 y la Tabla 5.5 pueden verse el gráfico y los datos, respectivamente, del ejemplo propuesto en esta política.

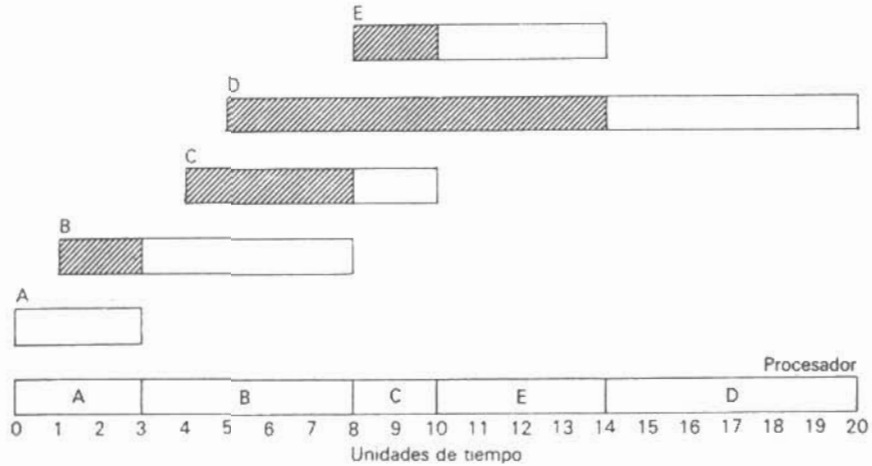


Figura 5.8. Política SJN.

Tabla 5.5

Nombre proceso	Instante llegada	Tiempo ejecución	Instante finalización	*T*	*E*	*I*
A	0	3	3	3	0	1.00
B	1	5	8	7	2	0.71
C	4	2	10	6	4	0.33
D	5	6	20	15	9	0.40
E	8	4	14	6	2	0.67
Media				7.4	3.4	0.62

Las características de esta política de planificación son las siguientes:

- No es apropiativa.
- El tiempo de espera aumenta de acuerdo con la longitud de los procesos, pero el tiempo medio de espera con respecto a otras políticas es óptimo.
- Es poco predecible.
- No es justa con los procesos largos.
- Buen tiempo de servicio.
- Resulta difícil de poner en práctica por los datos que necesita para realizarse la planificación.

5.5.4. Próximo proceso, el de tiempo restante más corto (SRT)

La política SRT *(Shortest Remaining Time)* es una mezcla de los dos métodos anteriores y trata de obtener las ventajas de ambos. Para ello esta técnica cambia el proceso que está en ejecución cuando se ejecuta un proceso (paso del planificador de largo plazo al de corto plazo), con una exigencia de tiempo de ejecución total menor que el que se está ejecutando en el procesador. El valor del tiempo de respuesta medio de los procesos largos mejora con respecto a SJN.

Presenta un excelente índice de servicio I y el tiempo de espera E es bastante corto para la mayoría de los procesos. SRT consigue una buena eficiencia, ya que logra que la lista de procesos preparados sea lo más corta posible.

En la Tabla 5.6 y Figura 5.9 pueden verse los datos y el gráfico, respectivamente, del ejercicio propuesto para esta política.

Tabla 5.6

Nombre proceso	Instante llegada	Tiempo ejecución	Instante finalización	*T*	*E*	*I*
A	0	3	3	3	0	1.00
B	1	5	10	9	4	0.55
C	4	2	6	2	0	1.00
D	5	6	20	15	9	0.40
E	8	4	14	6	2	0.67
		Media		7.0	3.0	0.72

Esta política presenta las siguientes características:

- Es una variante de SJN, para hacerla apropiativa.
- Puede ser injusta, ya que un proceso corto puede echar a uno largo que esté haciendo uso del procesador y que además esté terminando.
- Presenta una mayor sobrecarga.
- Excelente tiempo medio de servicio.
- Es muy eficiente.

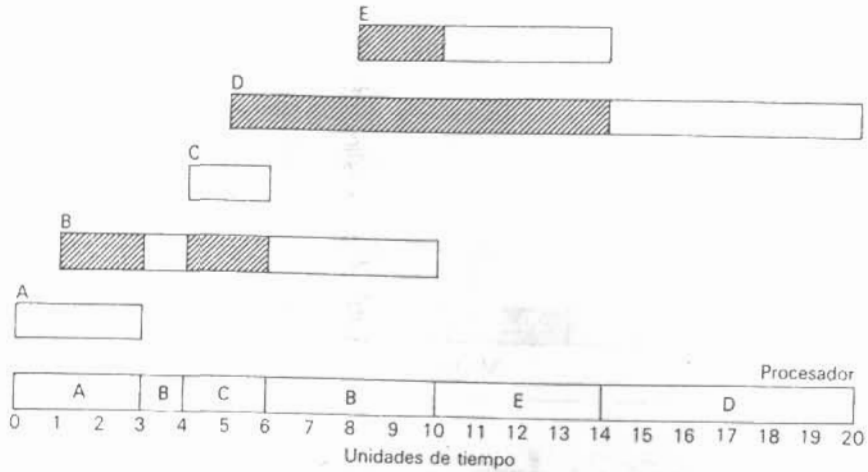


Figura 5.9. Política SRT.

5.5.5. Prioridad

En esta política se asocia a cada proceso una prioridad, de manera que el procesador se asigna al proceso de mayor prioridad.

Las prioridades pueden ser definidas interna o externamente. En el primer caso, el sistema operativo se basa en una serie de informaciones medibles para el cálculo y asignación de dichas prioridades (tiempo necesitado de procesador, necesidad de memoria, etc.).

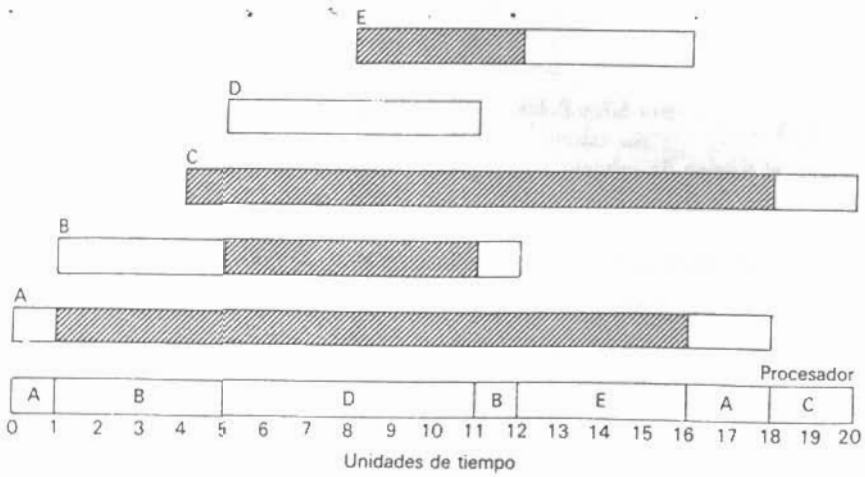


Figura 5.10. Política de prioridad.

El principal problema de esta política es el bloqueo o postergación indefinida, ya que un proceso de baja prioridad puede estar esperando su turno indefinidamente. Para evitarlo se suele emplear lo que se denomina envejecimiento de las prioridades, que aumenta gradualmente las prioridades de los procesos que están a la espera de utilizar el procesador.

Cualquier algoritmo basado en esta política puede ser apropiativo o no apropiativo. En el primer caso, un proceso puede ser retirado del procesador si aparece otro de mayor prioridad en la cola de procesos preparados.

El comportamiento de esta política como apropiativa puede verse gráficamente, para el ejemplo propuesto, en la Figura 5.10, y los datos en la Tabla 5.7.

Tabla 5.7

Nombre proceso	Instante llegada	Tiempo ejecución	Instante finalización	*T*	*E*	*I*
A	0	3	18	18	15	0.17
B	1	5	12	11	6	0.45
C	4	2	20	16	14	0.13
D	5	6	11	6	0	1.00
E	8	4	16	8	4	0.50
Media				11.8	7.8	0.45

5.5.6. Próximo el de más alto índice de respuesta (HRN)

HRN *(High Response Next)* es una política que trata de corregir las posibles injusticias de la política SJN con los procesos largos y las de la política FCFS con los procesos cortos.

Se basa en hacer variable la prioridad de un proceso, calculándola constantemente por medio de la expresión:

$$P = \frac{w + t}{t}$$

donde:

P es la prioridad del proceso.
w es el tiempo de espera en la cola de procesos preparados.
t es el tiempo de ejecución del proceso.

En un principio *P* valdrá 1 e irá aumentando paulatinamente a medida que el proceso permanezca en la cola de procesos preparados *(w* favorece a los procesos largos), e irá disminuyendo cuanto más tiempo esté en ejecución *(t* favorece a los procesos cortos).

Cuando un proceso que está en ejecución abandona el procesador, ya sea por que ha terminado o por una operación de entrada/salida, el proceso preparado que tenga mayor prioridad (que más haya esperado en la cola), será el que se seleccione para su ejecución.

Esta política presenta los siguientes inconvenientes:

- Si un usuario ejecuta un proceso corto inmediatamente después de que un proceso largo haya comenzado a utilizar el procesador, deberá sufrir una larga espera.
- Es muy costosa de llevar a la práctica, ya que la prioridad debe calcularse para todos los procesos en espera, cada vez que termine el proceso que está haciendo uso del procesador.
- Sobrecarga mucho el sistema debido a los cálculos que debe realizar.

El comportamiento de esta política y sus datos pueden verse en la Tabla 5.8 y la Figura 5.11 para el ejemplo propuesto.

Tabla 5.8

Nombre proceso	Instante llegada	Tiempo ejecución	Instante finalización	*T*	*E*	*I*
A	0	3	3	3	0	1.00
B	1	5	8	7	2	0.71
C	4	2	10	6	4	0.33
D	5	6	16	11	5	0.54
E	8	4	20	12	8	0.33
Media				7.8	3.8	0.58

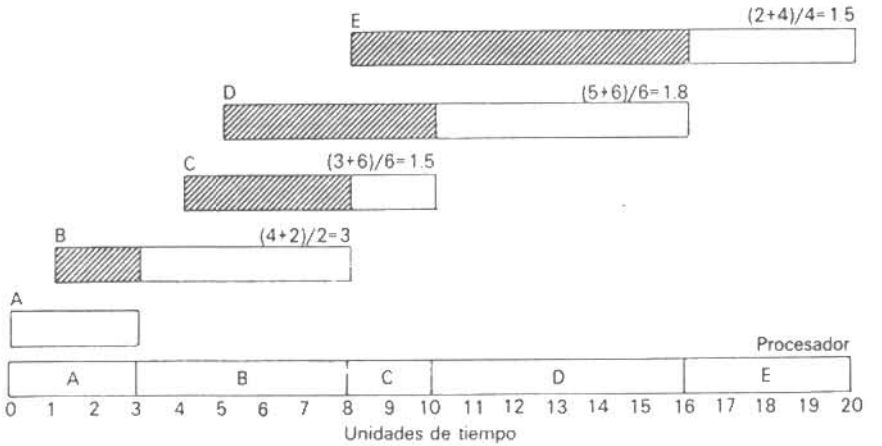


Figura 5.11. Política HRN.

Las características de esta política son:

- No es apropiativa.
- Es justa.
- Es costosa de poner en práctica.
- Produce una gran sobrecarga en el sistema.

5.5.7. Colas múltiples

Cuando los procesos que van a ser ejecutados en una computadora se pueden agrupar en distintos grupos, podemos asignarlos a diferentes colas, cada una con distinta planificación, para darle a cada una de ellas la que realmente necesite.

Esta política divide la cola de procesos preparados en varias colas separadas, de manera que los procesos se asignan a una determinada cola según sus necesidades y tipo.

Para determinar en cada caso qué cola es la que suministrará un proceso para que acceda al procesador cuando éste deje a otro anterior, será controlada por un algoritmo de planificación entre las colas, que normalmente es apropiativo de prioridad fija.

La Figura 5.12 ilustra esta política.

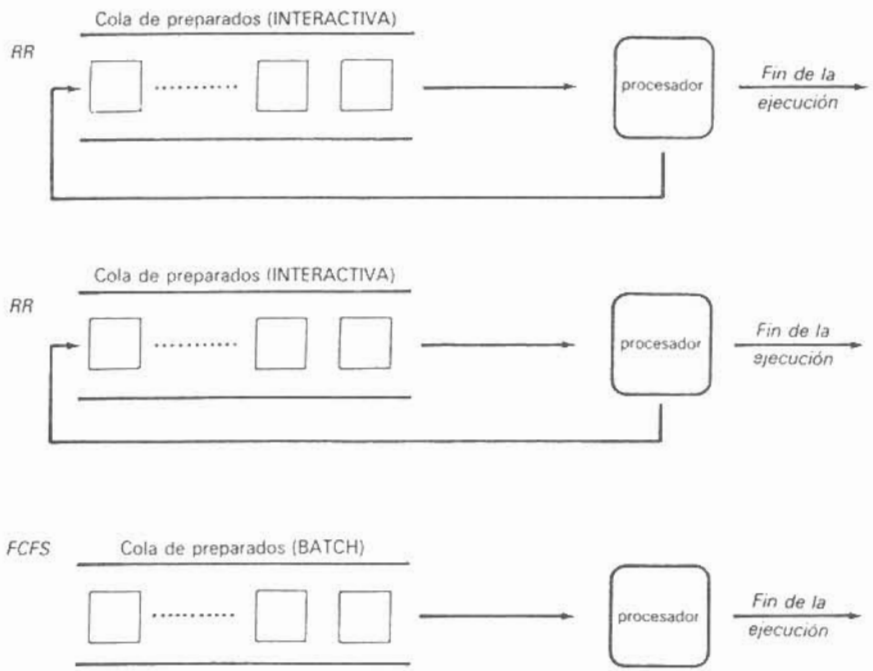


Figura 5.12. Planificación de colas múltiples.

5.5.8. Colas múltiples con realimentación (FB)

Para dar un trato justo a los procesos, es necesario conocer previamente todos sus parámetros característicos: la longitud, si están limitados por entrada/salida o por procesador, la memoria que van a necesitar, etc. Como estos datos no suelen ser conocidos, es difícil determinar el trato que debe recibir cada proceso.

Después de analizar las políticas anteriores, es fácil concluir que se deben adoptar las siguientes cuestiones:

- Favorecer los procesos cortos.
- Favorecer los procesos limitados por entrada/salida.
- Determinar la naturaleza del trabajo a realizar.

El método de colas múltiples con realimentación divide los procesos en varias colas de procesos preparados: cola 0, cola 1, cola 2, y así sucesivamente, de manera que las de numeración más baja tendrán asignadas mayor prioridad.

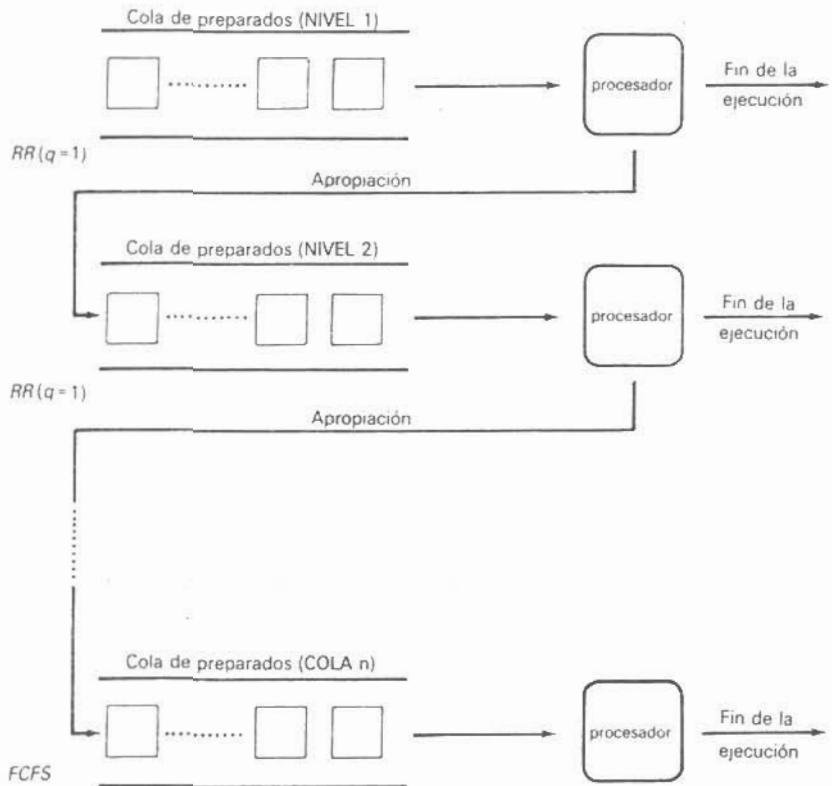


Figura 5.13. Planificación FB.

Cuando el proceso que está haciendo uso del procesador finaliza su quantum, se selecciona un nuevo proceso del principio de la cola del nivel más bajo que tenga algún proceso. Una vez que un proceso haya consumido el quantum de su cola un determinado número de veces, sin haber finalizado su ejecución, será colocado al final de la de nivel inmediatamente superior al anterior.

La política FB *(Feedback Multiple Queues)* intenta dar un trato justo a los procesos por medio de separación de los mismos en categorías, para así darles el servicio que necesitan. Los procesos limitados por procesador irán a las colas de menor prioridad (nivel más alto), mientras los de mayor prioridad serán aquellos procesos muy interactivos (Figura 5.13).

En la Tabla 5.9 y en la Figura 5.14 pueden verse los datos y la gráfica, respectivamente, de esta política, permitiendo un quantum a cada proceso de la cola antes de pasar a la siguiente.

Tabla 5.9

Nombre proceso	Instante llegada	Tiempo ejecución	Instante finalización	*T*	*E*	*I*
A	0	3	11	11	8	0.27
B	1	5	18	17	12	0.29
C	4	2	7	3	1	0.67
D	5	6	20	15	9	0.67
E	8	4	17	9	5	0.80
		Media		11.0	7.0	0.54

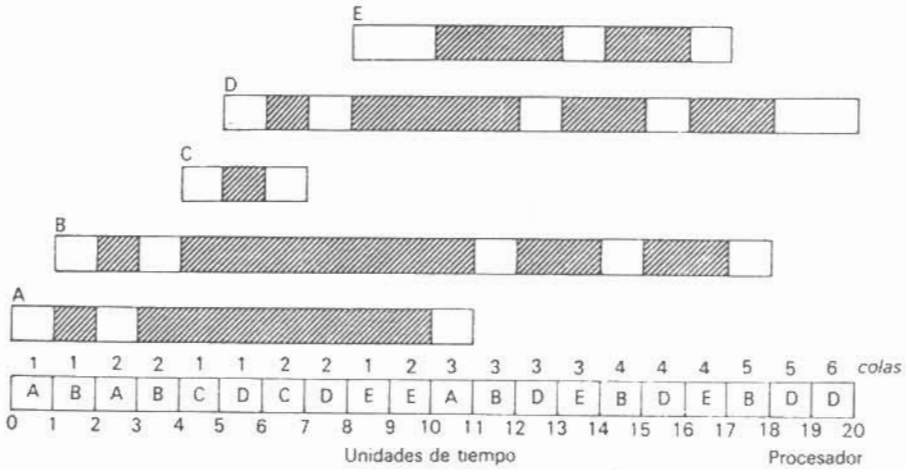


Figura 5.14. Política FB.

Las características de esta política son:

- Soporta bien la sobrecarga.
- Es apropiativa.
- Es muy adaptable a las necesidades del sistema, ya que cada cola puede ser gestionada de forma diferente.

CUESTIONES

1. ¿A qué se conoce con el nombre de planificación del procesador?
2. ¿Qué cometidos tiene la planificación a largo plazo, a medio plazo y a corto plazo?
3. ¿Cuáles son los objetivos que intentan cubrir las políticas de planificación del procesador?
4. ¿Qué criterios se deben tener en cuenta a la hora de elegir o diseñar un algoritmo de planificación del procesador?
5. ¿Cuáles son las medidas que se utilizan para el estudio del comportamiento de las distintas políticas de planificación del procesador?
6. ¿Qué es un algoritmo de planificación del procesador?
7. ¿En qué consiste una política apropiativa?
8. Comentar brevemente la política FCFS.
9. Comentar brevemente la política RR.
10. Comentar brevemente la política SJN.
11. Comentar brevemente la política SRT.
12. Comentar brevemente la política de prioridad.
13. Comentar brevemente la política HRN.
14. Comentar brevemente la política de colas múltiples.
15. Comentar brevemente la política FB.

CAPITULO 6

Proceso paralelo e interbloqueo

6.1. PROCESO PARALELO

En este capítulo vamos a estudiar algunos aspectos sobre el proceso paralelo y la concurrencia. Para comenzar, conviene fijar y definir algunos conceptos que serán utilizados a lo largo del mismo.

- **Actividades**
  - **Proceso.** Es un programa en ejecución.
  - **Tarea.** Son las distintas partes de un proceso que se ejecutan simultáneamente.
- **Sistemas**
  - **Multiprogramación.** Admiten varias actividades que comparten el procesador, pero sólo una puede estar ejecutándose en un momento dado.
  - **Multiproceso.** Las actividades se ejecutan en sus propios procesadores, compartiendo memoria común.
  - **Proceso distribuido.** Las actividades se ejecutan en sus propios procesadores, conectados a través de una red de comunicaciones.
- **Paralelismo**

Es la ejecución de diversas actividades simultáneamente en varios procesadores. Si sólo existe un procesador gestionando multiprogramación, se puede decir que existe **pseudo-paralelismo.** Se trata de un concepto físico producido por la existencia de varios procesadores.

- **Concurrencia**

Es la existencia de varias actividades ejecutándose simultáneamente, y necesitan sincronizarse para actuar conjuntamente. Se trata, en este caso, de un concepto lógico, ya que sólo hace referencia a las actividades, sin importar el número de procesadores presentes.

Es importante resaltar que para que dos actividades, sean concurrentes, es necesario que tengan alguna relación entre si, como puede ser la cooperación en un trabajo determinado o el uso de información compartida.

En un sistema monoprocesador, la existencia de multiprogramación es condición necesaria, pero no suficiente para que exista concurrencia, ya que los procesos pueden ejecutarse de forma totalmente independiente. Por ejemplo, un editor y un compilador pueden estar ejecutándose simultáneamente en una computadora sin que exista concurrencia entre ellos. Por otro lado, si un programa se está ejecutando y se encuentra grabando datos en un archivo, y otro programa también en ejecución está leyendo datos de ese mismo archivo, si existe concurrencia entre ellos, pues el funcionamiento de uno interfiere en el otro.

Si un sistema es multiprocesador, también pueden presentarse situaciones de concurrencia siempre y cuando las actividades necesiten actuar entre sí, bien por utilizar información común, o por cualquier otra causa.

6.1.1. Exclusión mutua

A continuación vamos a estudiar una serie de problemas que pueden surgir debido a la concurrencia de las actividades. Estos problemas no son tema propio de los sistemas operativos, pero es conveniente conocerlos para aquellos casos en que se puedan presentar y así utilizar aquellas herramientas que el sistema operativo ofrece para su solución.

Para abordar el estudio consideraremos los datos compartidos como un recurso que el sistema operativo debe gestionar para asegurar un uso correcto.

El principal problema con que nos encontraremos, que es intrínseco a la concurrencia, es el denominado **Exclusión Mutua.** Para su explicación, nos basaremos en el siguiente ejemplo:

Supongamos que en el sistema existe un archivo formado por registros compuestos por 5 campos (Figura 6.1).

Figura 6.1. Estructura del registro.

El diseño del archivo indica que el contenido de los campos es el siguiente:

- **Campo A.** Número del documento de identidad (DNI).
- **Campo B.** Nombre.
- **Campo C.** Primer apellido.
- **Campo D.** Segundo apellido.
- **Campo E.** Domicilio.

Para que un registro sea válido, debe estar totalmente actualizado, es decir, si se modifica el valor del campo A, el resto de los campos deben ser coherentes con el nuevo valor de dicho campo, ya que de otro modo el registro sería inconsistente.

Si en el momento en que un proceso está escribiendo o modificando un registro existiese otro proceso que realizase la lectura de ese mismo registro, y el primero de ellos sólo hubiese tenido tiempo de escribir o modificar el campo A, la información obtenida por el segundo proceso sería inconsistente. Para evitar esta situación, se deben **sincronizar los procesos** de manera que mientras uno esté escribiendo, ningún otro puede leer. Ahora bien, esto no ocurre en aquellos casos en que dos procesos tratan de leer en el mismo archivo, aún coincidiendo en el mismo registro.

Para generalizar el caso anterior, supongamos los dos procesos que coinciden en el mismo registro, uno escribiendo y el otro leyendo, que representamos en la Figura 6.2, y que los llamaremos ESCRIBIR y LEER, se encuentran en un sistema monoprocesador multiprogramado y son los únicos presentes en ese momento.

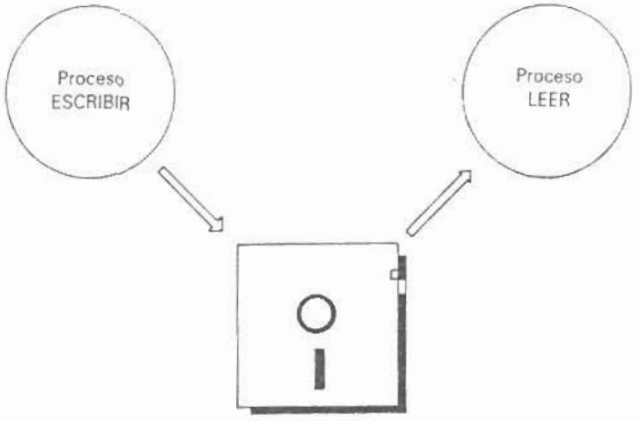


Figura 6.2. Concurrencia.

En el momento de un cambio de proceso del uno al otro se pueden producir las siguientes situaciones:

- **Sin sincronización entre procesos.** Puede darse el caso de que ESCRIBIR esté actualizando un registro y se quede a medias, sorprendiéndole el cambio de proceso, por tanto, terminará de escribirlo cuando vuelva a hacer uso del procesador.
  Con el cambio le tocará el turno al proceso LEER, que accederá a dicho registro pudiendo leerlo completamente. Es evidente que los datos leídos serán inconsistentes.
- **Con sincronización entre procesos.** Supongamos algún mecanismo que prohíba la lectura *(bloqueo de registro)* a cualquier proceso, mientras el proceso ESCRIBIR esté realizando alguna operación. En este caso, LEER, al hacer uso del procesador y solicitar una operación de lectura sobre un registro que se encuentra bloqueado, quedaría en espera de que el registro quede totalmente escrito y se proceda a su desbloqueo. LEER pasaría a estado bloqueado. ESCRIBIR terminaría su trabajo sobre el registro y en el siguiente cambio LEER procedería a hacer el suyo.

Esta sincronización por la cual una actividad impide que otras puedan tener acceso a un dato mientras se encuentra realizando una operación sobre el mismo es lo que se conoce como **Exclusión Mutua.**

La zona de código de un proceso que no puede ser interrumpida por otro, por los motivos anteriormente expuestos, se denomina como **sección crítica.**

6.1.2. Sincronización

Si una actividad desea impedir que otra acceda a ciertos datos compartidos, mientras no se cumpla una determinada condición, debemos sincronizar las actividades con dicha condición. Por tanto, la sincronización es un elemento necesario para asegurar la exclusión mutua.

Los algoritmos que se han diseñado para este fin se clasifican en tres grupos:

▪ Espera activa

Son aquellos algoritmos que basan todo su funcionamiento en establecer la espera de entrada a la sección crítica con un bucle que será roto en el momento en que se cumpla una determinada condición. Se llaman de espera activa porque el proceso no queda bloqueado durante su ejecución, sino que estará compitiendo por el procesador constantemente. Por este motivo, estos algoritmos sobrecargan el sistema innecesariamente. Fueron los primeros en utilizarse y han sido sustituidos por otros más eficientes.

Entre los distintos algoritmos de este tipo existentes podemos citar:

- **Espera con mutex.** Algoritmo que utiliza un switch (MUTEX) a través del cual se produce la sincronización.
- **Alternancia.** Ligeramente mejor que el anterior, utiliza también una variable TURNO para realizar el sincronismo entre los procesos.
- **Algoritmo de DEKKER.** Resuelve el problema mediante la solución propuesta por Dekker, basando su funcionamiento en una tabla unidimensional de dos elementos lógicos (Switches).

▪ Espera no activa

Son los algoritmos que establecen la espera para entrar en la sección critica bloqueando el proceso, haciendo que deje de competir por el procesador hasta que se cumpla la condición de desbloqueo.

Entre los algoritmos existentes, citaremos los siguientes:

- **Semáforos.** Para eliminar los problemas que se producen con los algoritmos de espera activa, fundamentalmente los referidos a la sobrecarga que producen en el sistema, E. W. Dijkstra (1965) diseñó un mecanismo basado en una variable entera utilizada como contador de peticiones de entrada a una sección crítica. Esta variable es compartida por todos los procesos del sistema. Este nuevo tipo de variable se denomina **semáforo** por su capacidad de gestionar el tráfico de procesos que desean acceder a datos compartidos.

  Con este sistema, cuando un proceso intente entrar en una sección crítica mientras otro está accediendo a los datos compartidos, se bloqueará de igual manera que cuando un proceso accede a un recurso que está ocupado.

- **Regiones críticas.** Son sistemas que permiten establecer protecciones contra una mala utilización de los usuarios. Para ello sólo permiten que los datos compartidos se puedan acceder desde determinadas regiones quedando transparentes desde el resto.

  Tiene pequeños inconvenientes relacionados con la sincronización y no permite que varias actividades puedan realizar operaciones de lectura simultánea.

- **Regiones críticas condicionales.** Consiste en una mejora del método anterior tratando de resolver algunos problemas de sincronización que se presentaban.

- **Monitores.** Uno de los problemas existentes en los mecanismos anteriores es que el programador tiene que proporcionar de forma explícita el modo de sincronización. Para evitarlo, B. Hansen y C. A. R. Hoare desarrollaron un nuevo mecanismo denominado **Monitor,** que debe ser soportado por el lenguaje correspondiente.

  Un monitor permite compartir, segura y eficientemente, datos entre varias actividades, garantizando la exclusión mutua, sin necesidad de que el programador tenga que suministrarla explícitamente.

  Se basa en dos premisas: la primera es la abstracción de datos consistente en una técnica capaz de separar las operaciones a ejecutar sobre los datos, de los detalles de diseño propios de los mismos (los lenguajes Modula y Ada soportan este tipo de estructuras). La segunda es que realizan la exclusión mutua de forma implicita.

  La finalidad más útil de los monitores es reunir todas las funciones que operan sobre un conjunto de datos compartidos en un sólo módulo, de manera que todos los accesos a esos datos estarán forzados a utilizar dichas funciones.

- **Contadores de eventos.** Es un mecanismo para sincronizar actividades sin que sea necesario forzar la exclusión mutua, ya sea por que no deseamos limitar la concurrencia de las actividades, o simplemente porque no lo necesitemos. Se basa en una variable entera cuya misión es contar determinadas operaciones.

- **Mensajes.** Es un mecanismo que permite a los procesos intercambiar aquella información que sea necesaria durante el desarrollo normal de su ejecución. Por tanto, es más un mecanismo de cooperación que de sincronización. Esta cooperación se realiza por medio de mensajes que se envian entre si los procesos. Se basa en una zona de memoria compartida que gestiona el sistema operativo directamente y que permanece oculta a los procesos. De esta manera el proceso que envía un mensaje a otro deposita la información en la zona de memoria compartida, y el receptor lo lee de ella posteriormente. En este sistema, las directrices de envio y recepción establecen una sincronización entre los procesos al tener que esperar por dichos mensajes, antes de continuar con su ejecución.

Existen dos tipos de comunicación entre procesos:

- **Directa.** Los procesos envían y reciben los mensajes directamente entre sí, de manera que se asocia un enlace bidireccional único entre cada dos procesos.
- **Indirecta.** Los mensajes son enviados y recibidos a través de **mailboxes** o **buzones de correos.** Con este método cada proceso puede estar relacionado con tantos buzones como desee consiguiendo comunicarse con tantos procesos como sea necesario.

- **Llamadas remotas.** El paso de mensajes de un proceso a otro es un mecanismo muy utilizado para resolver los problemas de concurrencia entre procesos. Es análogo al paso de parámetros en una llamada a una rutina o procedimiento. Si unimos los conceptos de mensajes y paso de parámetros, tendremos lo que se conoce como **rutinas** o **procedimientos remotos,** creandose una copia del mismo cada vez que son ejecutadas *(llamadas remotas),* siendo dicha ejecución concurrente. Este tipo de interacción asegura que hasta que un proceso no termine determinadas operaciones, el siguiente permanecerá en espera.

- **Rendez-vous.** Es la culminación de todos los mecanismos anteriores, tratándose de una ligera modificación del método de las llamadas remotas, donde la llamada, en lugar de ser a todo un procedimiento, lo es solamente a un grupo de sentencias dentro de él. De igual forma, se trata de un procedimiento similar al monitor, pero con más versatilidad y potencia. Este método se ha llevado a la práctica con el lenguaje Ada.

▪ Mecanismos hardware

Son instrucciones hardware que aseguran la exclusión mutua. Entre las más utilizadas citaremos las siguientes:

- **Deshabilitar interrupciones.** Las computadoras actuales permiten que las interrupciones puedan ser deshabilitadas, de esta forma, si un dispositivo generase una interrupción estando deshabilitada, el reconocimiento y tratamiento de la nueva interrupción se retendrá hasta que se habiliten de nuevo las interrupciones. Por este medio, se puede forzar la exclusión mutua deshabilitando las interrupciones mientras haya alguna actividad en la sección crítica. De este modo, dicha actividad no podrá ser interrumpida y, por tanto, no se podrá producir ningún cambio de proceso. Normalmente la deshabilitación y nueva habilitación de las interrupciones suele hacerse con una instrucción máquina, por lo que resulta ser una operación muy rápida.

- **Instrucción TEST_AND_SET.** Muchas de las computadoras actuales suministran una instrucción máquina denominada Test and Set, cuya misión es la de forzar la exclusión mutua. La ventaja de este mecanismo es que no puede ser interrumpida por ser una instrucción hardware y el inconveniente es que no son muchas las computadoras que la poseèn. Esta instrucción, por sí sola, no basta para asegurar la exclusión mutua, por lo que tendremos que basarnos en ella para construir los denominados locks.

- **LOCK.** Se basa en la instrucción anterior y su cometido es permitir el acceso a la sección crítica a un proceso en caso de no existir otra actividad dentro de su sección crítica, no permitiéndolo en caso contrario. En este caso, el número de actividades y de procesadores compartiendo memoria principal puede ser cualquiera, y además, son mecanismos simples y fáciles de probar y depurar.

6.2. INTERBLOQUEO

En la teoría de los sistemas operativos, se puede definir el problema del interbloqueo como la situación de un conjunto de procesos en un estado de espera tal que ninguno de ellos tiene suficientes criterios para continuar su ejecución. Circunstancias parecidas suceden a menudo en la vida real, por ejemplo, en una carretera de dos direcciones, donde en un determinado punto de cruce con la vía férrea, se ha construido un puente que por problemas urbanísticos o de presupuesto sólo deja pasar vehículos en un sentido. Dado este punto crítico en la mencionada carretera, se presentan las siguientes situaciones:

- Un vehículo llega al puente y no se encuentra ningún otro en sentido contrario. En este caso, cruza haciendo uso del puente y no ocurre nada anormal.
- Si el paso por el puente es controlado por un semáforo en cada lado de manera que 100 metros antes de cada semáforo se sitúen sendos detectores de presencia de vehículos cuya finalidad sea poner en rojo el semáforo del sentido contrario ante la presencia de un vehículo, podría suceder que si llegan al mismo tiempo vehículos en los dos sentidos se pongan los dos semáforos en rojo impidiendo el paso de vehículos en ambos sentidos. En este caso, el camino queda bloqueado, lo que representa un silogismo al interbloqueo entre procesos.
- Si no habiendo semáforos, el conductor situado en uno de los extremos es lo suficientemente educado que deja pasar en primer lugar al del otro extremo y antes de terminar de cruzar este último aparece por el mismo extremo otro vehículo, y así sucesivamente. Puede ocurrir que la educación del primer conductor haga que no cruce el puente mientras aparezcan vehículos por el otro extremo. Esta situación podemos emparejarla con la de postergación indefinida que estudiaremos más adelante.

Visto el ejemplo anterior que nos introduce en los conceptos básicos que vamos a tratar a continuación referentes al interbloqueo y a la postergación indefinida, damos paso al estudio de recursos y del modelo de sistema en los que basaremos dichos conceptos.

6.2.1. Recursos

Se entiende como recurso un elemento que un programa o proceso puede utilizar en la computadora donde se está ejecutando. Se engloban bajo el concepto de recurso, tanto los dispositivos hardware (por ejemplo, una impresora), como una cierta cantidad de información (por ejemplo, un registro de un archivo). No obstante, en una computadora pueden existir muchos tipos de recursos, e incluso varios del mismo tipo. Por ello definiremos un recurso como algo que puede ser utilizado por un solo proceso en un instante dado.

Para que un proceso pueda utilizar un recurso, deberá realizar la siguiente secuencia de operaciones:

- **Solicitar el recurso.** Si no estuviera disponible el proceso, quedará bloqueado hasta que le pueda ser asignado.

- **Utilizar el recurso.**
- **Liberar el recurso.**

6.2.2. Modelo

En primer lugar vamos a definir lo que entendemos por interbloqueo, también conocido como **abrazo mortal** *(deadlock)*. Se dice que un conjunto de procesos han alcanzado un estado de interbloqueo si cada uno de ellos espera que ocurra algo que sólo puede ser producido por uno de los procesos del propio conjunto (no necesariamente tiene que ser el mismo suceso).

Como todos los procesos están en espera, ninguno de ellos será el primero en producir el suceso deseado y por tanto permanecerán esperando indefinidamente.

Para formalizar todo lo expuesto hasta el momento, vamos a fijar los principios en que se basa todo sistema informático:

- Posee un número finito de recursos.
- Existe un número finito de procesos que compiten por los recursos.
- Los recursos se pueden dividir en tipos de tal forma que cada uno de ellos esté compuesto por recursos idénticos entre sí.
- Los procesos deben realizar las tres acciones expuestas anteriormente sobre los recursos: solicitar, utilizar y liberar.
- Un proceso puede pedir tantos recursos como necesite para realizar su trabajo, ya sean del mismo tipo o no, siempre que no excedan del total existente en el sistema.
- Las operaciones sobre los recursos se realizarán a través de llamadas al sistema operativo, de manera que si se solicita un recurso que está siendo utilizado, quedará bloqueado en espera de que se libere dicho recurso.

6.2.3. Postergación indefinida

Un problema asociado al interbloqueo es el de la postergación indefinida. En cualquier sistema que existan procesos en espera por operaciones sobre recursos y por las decisiones de planificación del sistema operativo, es posible que un determinado proceso quede esperando indefinidamente por el recurso deseado, mientras otros reciben la atención del sistema. Esta situación se conoce como **postergación indefinida (starvation).**

Este tipo de problemas suele surgir en sistemas gestionados por prioridades ya que un proceso puede quedarse sin el control del procesador debido a que continuamente lleguen nuevos procesos de prioridad más alta. Como ya se vio, una forma de corregirlo es por medio del envejecimiento de las prioridades.

6.2.4. Condiciones de interbloqueo

Podemos asegurar que un conjunto de procesos ha llegado al interbloqueo si se cumplen las siguientes condiciones:

- **Exclusión mutua.** Existencia al menos de un recurso compartido por los procesos, al cual sólo puede acceder uno simultáneamente.
- **Posesión y espera.** Debe existir algún proceso que tenga asignado un recurso y esté esperando a que se le asignen otros que están siendo utilizados por otros procesos.
- **No apropiación.** Los recursos no pueden ser apropiados por los procesos, es decir, los recursos sólo podrán ser liberados voluntariamente por sus propietarios.
- **Espera circular.** Si existen varios procesos P_1, P_2, ..., P_n, donde P_1 estará esperando un recurso que tenga asignado P_2, P_2 estará esperando uno de P_3, y así sucesivamente hasta P_n que estará esperando uno asignado a P_1.

En caso de que alguna de las condiciones anteriores no estuviera presente, podemos asegurar sin ningún género de dudas que no existe interbloqueo.

6.2.5. Tratamiento de interbloqueo

Existen cuatro estrategias para el tratamiento del interbloqueo:

- **Ignorar**

Se trata de no tener en cuenta en el sistema operativo este tipo de situaciones, teniendo en cuenta que su aparición es bastante improbable. La solución en este caso si se presenta es volver a arrancar el sistema.

- **Prevenir**

Consiste en evitar alguna de las cuatro condiciones que deben estar presentes para que pueda aparecer el interbloqueo, con lo cual nunca se podrá presentar tal situación.

- **Evitar**

Se pueden evitar los posibles interbloqueos siempre y cuando tengamos disponible cierta información sobre los recursos que necesita cada proceso por adelantado. Para evitar el interbloqueo se han estudiado diversos métodos, entre los que podemos citar el algoritmo del banquero.

- **Detectar y recuperar**

Consiste en abortar un proceso cuando existen indicios de que se pueda producir interbloqueo. Para abortar un proceso, el sistema operativo se basa en varias características del mismo:

- **Prioridad.** Se elimina el menos prioritario.

- **Tiempo de procesador usado.** Cuanto menos tiempo haya estado el proceso a abortar menos trabajo se pierde y más fácil será recuperarlo.
- **Tipo de recursos utilizados.** Si los recursos son muy críticos y escasos, será preferible liberarlos cuanto antes.
- **Necesidades de recursos.** Es conveniente eliminar aquellos procesos que necesitan un gran número de recursos.
- **Facilidad de suspensión/reanudación.** Se eliminarán antes aquellos procesos cuyo trabajo perdido sea fácil de recuperar.

CUESTIONES

1. ¿Cuáles son las diferencias esenciales entre los sistemas con multiprogramación, multiproceso y proceso distribuido?
2. ¿En qué consiste el paralelismo?
3. ¿Qué es la concurrencia entre procesos?
4. ¿En qué se fundamenta la exclusión mutua?
5. ¿Por qué motivos se hace necesaria la sincronización entre procesos?
6. Enumerar los distintos algoritmos diseñados para asegurar la exclusión mutua
7. ¿En qué consiste el interbloqueo entre un conjunto de procesos?
8. Definir el concepto de recurso.
9. ¿Qué operaciones tiene que hacer un proceso para poder hacer uso de un recurso?
10. ¿En qué consiste la postergación indefinida y qué tipos de procesos son los más expuestos a ella?
11. ¿Cuáles son las condiciones que se tienen que cumplir para que se pueda presentar el interbloqueo entre procesos?
12. ¿Cómo se puede evitar una situación de interbloqueo entre procesos?

CAPITULO 7

Gestión de la memoria principal

7.1. INTRODUCCION

Para que un programa pueda ser ejecutado en una computadora, tanto él como los datos que vaya a manejar deben estar almacenados en la memoria principal o física.

Por otra parte, para mejorar el rendimiento del procesador y su capacidad de proceso, se pueden repartir los servicios del mismo entre varios programas que necesitan estar simultáneamente cargados en la memoria, por tanto se hace necesario **«compartir»** la misma.

En el funcionamiento de una computadora podemos considerar la memoria principal como el recurso central, ya que tanto el procesador como los dispositivos de entrada/salida acceden a ella para leer y/o grabar la información que manejan (Figura 7.1).

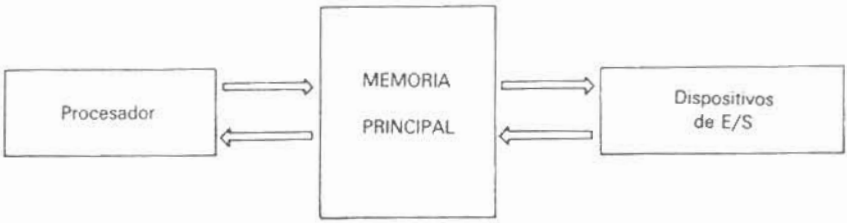


Figura 7.1. La memoria como recurso central.

El procesador leerá de la memoria una instrucción para ejecutarla, y a su vez tomará de la misma los datos que necesite para al final depositar el posible resultado también en la memoria. Esta operación se estará repitiendo constantemente, y por ello la velocidad a la que se realicen estos accesos, tanto de lectura como de escritura, condicionarán la rapidez y eficacia de la computadora.

En general, se consideran importantes dos parámetros relacionados con la velocidad de lectura y escritura de datos en la memoria principal. Se denomina **tiempo de acceso** a la memoria al tiempo que transcurre entre el inicio y el fin de una operación de lectura o escritura sobre la misma. El segundo parámetro característico es el **tiempo de ciclo de memoria** que marca el retraso que impone el hardware entre el fin de una operación y el principio de la siguiente. Ambos factores marcan la velocidad de la memoria principal.

7.2. DIRECCIONAMIENTO

La memoria física se puede ver como una sucesión de bytes o palabras, cada uno con su propia dirección, de tal forma que se puede acceder a ellos de forma directa indicando dicha dirección (Figura 7.2).

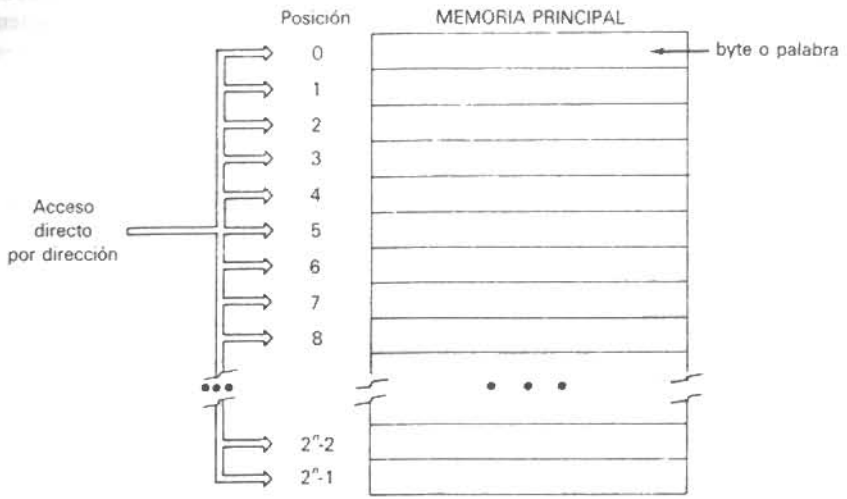


Figura 7.2. La memoria principal como una sucesión de bytes o palabras.

Cuando se desarrolla el hardware de una computadora, se define el esquema de direccionamiento y la configuración de su memoria, de tal forma que quede perfectamente identificado cómo será el funcionamiento y capacidad de todo acceso a la misma. Por ejemplo, si para direccionar un byte de la memoria se utilizan 16 bits, sólo podremos direccionar 2^{16} bytes de la misma, que será el espacio máximo direccionable de la computadora.

El hardware también impone la unidad de información a la que se accede en cada direccionamiento. Si la computadora es direccionable por palabra, serán éstas las que ocuparán las sucesivas posiciones a las que podremos acceder (en general el tamaño de una palabra dependerá de la computadora y estará compuesto por varios bytes).

Según el tamaño de la memoria a la que se accede en cada operación, existen dos tipos de computadoras.

- Direccionables por byte.
- Direccionables por palabra.

7.2.1. Asignación de direcciones

Hasta ahora sabemos que los programas, para ser ejecutados, deben estar y compartir memoria con otros programas, pero, ¿en qué direcciones se deben cargar?

El programador que diseña su programa no puede saber dónde se cargará, simplemente definirá una sentencia de inicio que marcará el comienzo de sus instrucciones, y a partir de ella, describirá el resto de su programa. Posteriormente, el compilador traducirá a lenguaje máquina el programa fuente y les asignará a las diferentes instrucciones y variables un desplazamiento con respecto a la sentencia inicial que se denomina **cero relativo.** A esta operación se le llama **asignación de direcciones relativas** al programa. Una vez enlazado el programa *(link),* el cargador podrá colocarlo en memoria asignándole al cero relativo una dirección real, quedando definidas a partir de ella el resto de direcciones del programa por medio de los mencionados desplazamientos. Se realizará lo que se denomina **transformación de direcciones relativas en absolutas o reales** (Figura 7.3).

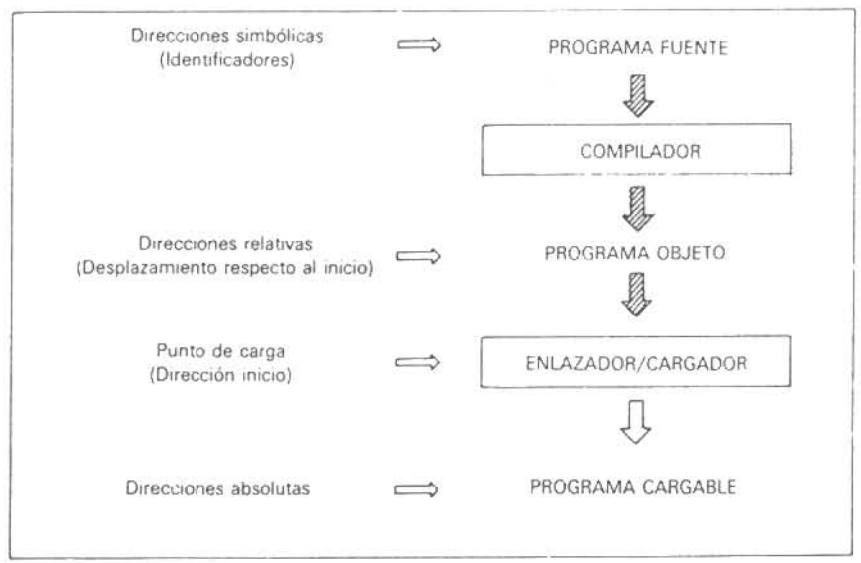


Figura 7.3. Asignación de direcciones en los programas.

Se logra así independizar el desarrollo de un programa de su futura dirección de carga en memoria.

7.3. JERARQUIA DE ALMACENAMIENTO

Las primeras computadoras utilizaban memorias de tipo magnético como eran las de núcleos de ferrita. Dado el nivel tecnológico del momento, se trataba de dispositivos muy grandes y caros que mantenían en todo momento los programas y datos que ejecutaban. Poco tiempo después, parecía ilógico mantener cargados en memoria principal aquellos programas y datos que no fueran a ejecutarse, por otro lado, tampoco sería práctico compilarlos

y enlazarlos cada vez que fuera necesaria su ejecución. Por ello, el desarrollo de dispositivos rápidos de almacenamiento como las cintas y los discos permitió guardar en ellos dichos programas y datos, transfiriéndolos a la memoria principal en el momento en que fueran a ejecutarse, encargándose el sistema operativo de realizar la gestión de este almacenamiento secundario.

En la década de los sesenta, las nuevas tecnologías permitieron el desarrollo de memorias mucho más rápidas que las magnéticas, aunque también más caras. Estas memorias rápidas denominadas **cache** se comenzaron a utilizar para almacenar los programas y datos más utilizados, lográndose una mejora general en el funcionamiento del sistema que las incluía (Figura 7.4).

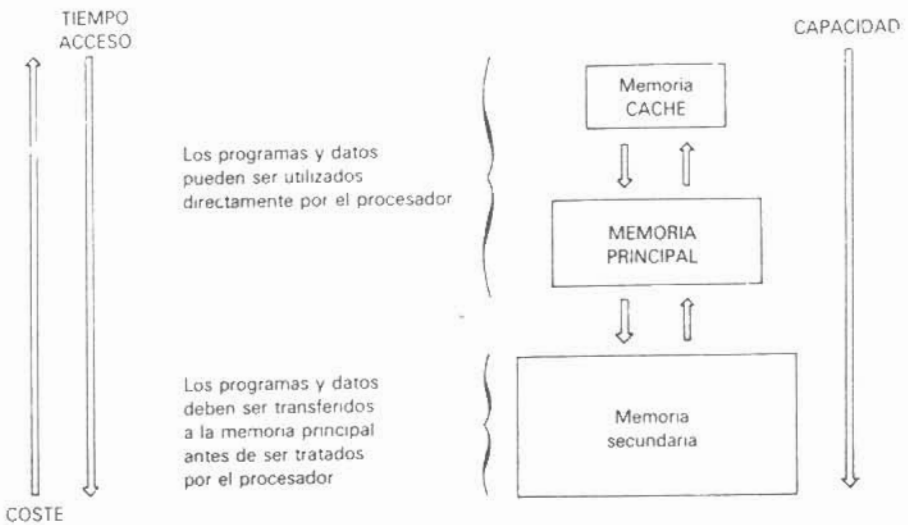


Figura 7.4. Jerarquía de almacenamiento.

7.4. GESTION DE LA MEMORIA

Al ser la memoria un recurso tan caro y tan influyente en el rendimiento general de una computadora, debe ser manejado de la forma más eficiente posible. Por este motivo, el sistema operativo dedica gran parte de su software a su gestión **(gestor de memoria),** buscando la utilización más adecuada al servicio que debe dar.

Históricamente se han conjugado avances tecnológicos con las exigencias del propio sistema que cada vez procesa más trabajo en menos tiempo. Estos factores han permitido el desarrollo de nuevas técnicas de gestión de la memoria.

7.4.1. Monoprogramación

Como ya se vio en el Capítulo 1, la monoprogramación es el modo de trabajo en el que solamente un programa utiliza el procesador, siendo además el único existente en la memoria principal, de tal forma que hasta que éste no termine su trabajo no cederá el control al siguiente programas que será cargado en memoria, sustituyendo al anterior. Veremos a continuación cómo se ha venido gestionando y cómo se gestiona la memoria en aquellos sistemas que utilizan este modo de trabajo.

- **La memoria dedicada**

Las primeras computadoras utilizaban lo que se denomina régimen dedicado, consistente en que el programador accedía directamente al hardware y gestionaba la memoria en sus programas. En estos primeros equipos se programaba en lenguaje máquina, sin existir sistema operativo y consiguientemente tampoco gestor de memoria (Figura 7.5).

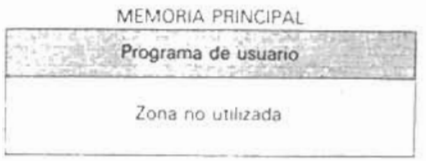


Figura 7.5. Memoria dedicada.

De esta forma, la utilización de la memoria es ineficaz y se obliga a un conocimiento profundo del hardware utilizado. El programador dedicaba gran parte de su tiempo y esfuerzo a gestionar el propio hardware, desviándolo de su principal objetivo que era el proceso de la información.

- **División de la memoria. El monitor residente**

La introducción de los sistemas operativos para utilizar mejor el hardware dividió la memoria en dos zonas, una utilizable por el usuario, y otra reservada para la parte residente del propio sistema operativo denominada comúnmente **monitor.** Este, entre otras funciones, se encarga de gestionar la memoria (Figura 7.6).

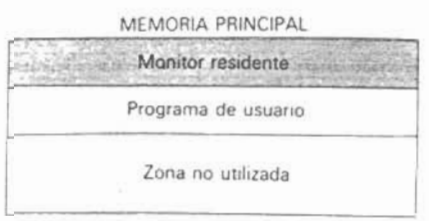


Figura 7.6. Monitor residente.

Este reparto de la memoria introduce nuevos problemas que es necesario resolver, estos problemas son:

- Cómo asegurar la independencia de ambas zonas (Protección).
- Cómo asegurar que los programas de usuario no se vean afectados por esta división de la memoria.

■ Protección de la memoria

Debido a que el programa monitor y el programa de usuario van a compartir la memoria, se hace necesario proteger la zona del sistema operativo contra cualquier intento de acceso a dicha zona por parte del programa.

Se establece, por tanto, una **dirección frontera** que limita la zona del sistema operativo. Cualquier dirección de memoria que pida o solicite el programa de usuario se compara con dicha dirección frontera, permitiendo el acceso o no, según corresponda. Este control se realiza directamente por el hardware (Figura 7.7), a través de uno de sus registros.

Evidentemente, este control provoca un cierto aumento de tiempo en el acceso a la memoria, pero éste queda compensado con la mejora en rendimiento que permite el sistema operativo.

La dirección frontera, en general, suele ser variable para permitir que un sistema pueda ir evolucionando en prestaciones y versiones de su propio sistema operativo, en definitiva que pueda variar su tamaño, obligando por tanto a variar la dirección frontera.

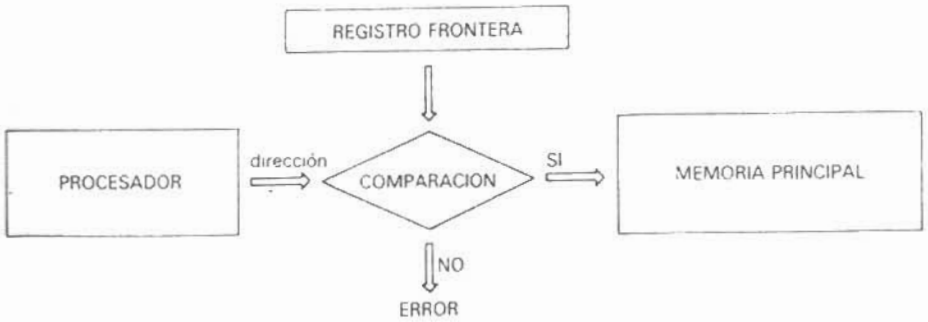


Figura 7.7. Protección de memoria.

■ Reasignación de direcciones

Una vez inicializado el sistema operativo, el contenido del registro frontera indicará el punto a partir del cual puede cargarse el programa de usuario. Para ello, es necesario reasignar las direcciones del programa en función de la frontera. Existen dos formas de realizar la reasignación, una **estática** y otra **dinámica.**

La **reasignación estática** se realiza durante la compilación, o bien durante la carga del programa en memoria. De esta manera, cualquier variación de tamaño en el sistema operativo exige una nueva compilación o carga del programa. Es una técnica fácil de realizar, pero demasiado rígida.

La **reasignación dinámica** se realiza durante la ejecución del programa. Un dispositivo especial del hardware interceptará cada dirección lógica generada por el programa y le sumará el contenido del registro frontera para obtener la dirección real correspondiente (Figura 7.8).

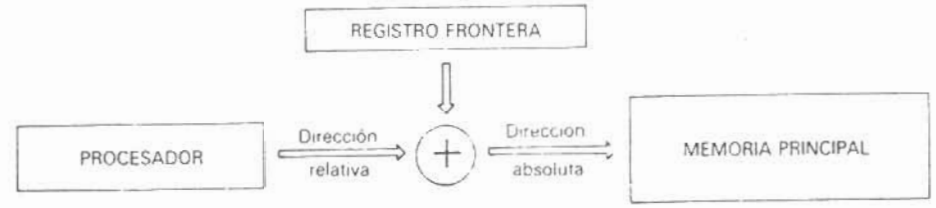


Figura 7.8. Reasignación dinámica.

Cualquiera que sea la técnica utilizada, el usuario no manejará direcciones reales en su programa. Utilizará direcciones relativas que podrán variar entre 0 y el máximo permitido por el sistema operativo. Este es el denominado **espacio lógico de direcciones.**

Posteriormente, el sistema operativo, con la ayuda del hardware, establecerá la correspondencia adecuada entre las direcciones relativas y las direcciones reales que configuran lo que se denomina **espacio físico de direcciones** *(memoria real).*

Esta separación entre la visión del usuario *(espacio lógico)* y la memoria real *(espacio físico)* permite gestionar ésta con mayor eficacia.

■ Intercambio de almacenamiento

El desarrollo de dispositivos rápidos de almacenamiento secundario *(discos)* hizo posible mejorar la utilización de la memoria.

La necesidad de atender a varios usuarios en los sistemas de tiempo compartido, impulsó el desarrollo de técnicas de **intercambio de almacenamiento** o **swapping.**

Cuando un programa cargado en memoria quede a la espera de una nueva orden, el sistema operativo lo descargará *(swap-out)* en dispositivos rápidos de almacenamiento secundario. En su lugar cargará el de otro usuario *(swap-in),* traído de dichos dispositivos (Figura 7.9).

Evidentemente, la eficacia de esta técnica dependerá, principalmente, de la velocidad con que se realicen la carga y descarga de los programas y de la velocidad de acceso a los dispositivos de almacenamiento secundario.

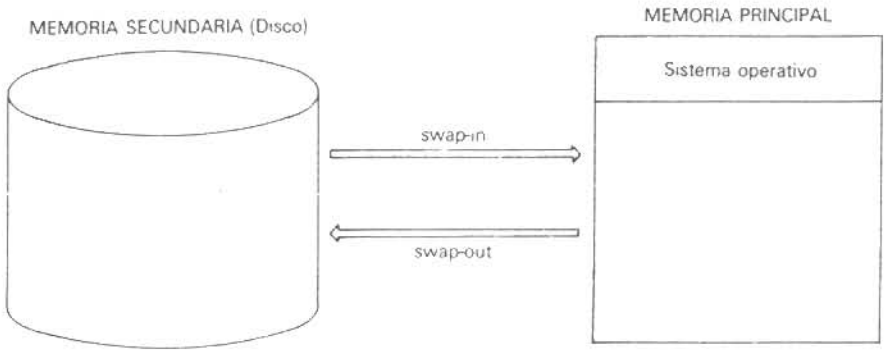


Figura 7.9. Intercambio de almacenamiento.

Se puede reducir la influencia del cambio de programa si se solapa con la ejecución. Para ello, el sistema divide la zona de usuario en dos partes. Mientras en una está procesándose un programa, en la otra se está cargando otro (Figura 7.10).

Figura 7.10. Intercambio solapado.

En este caso, el sistema debe encargarse de las interrupciones y situaciones de los procesos extraídos de la memoria *(operaciones de entrada/salida, etc.)*. Por ejemplo, si un programa se queda a medias en una operación de entrada de datos y el sistema lo extrae de la memoria, se encargará de almacenar dichos datos de entrada en lo que se denomina **buffer** hasta que el programa vuelva de nuevo a la memoria y empiece a ejecutarse otra vez.

Hasta aquí hemos considerado sistemas en los que, en un momento dado, sólo existe un programa de usuario en la memoria principal, es decir, sistemas que trabajan en monoprogramación.

Con el solapamiento en el intercambio se introduce un nuevo modo de trabajo en el que existen en un determinado momento varios programas de usuario en la memoria, que pasamos a estudiar a continuación.

7.4.2. Multiprogramación

Como acabamos de ver, la necesidad de utilizar más eficazmente las computadoras y su memoria nos introduce en la técnica de la multiprogramación que, como vimos en el Capítulo 1, permite el acceso al procesador de varios procesos simultáneamente, repartiendo el tiempo entre todos ellos, según diversas técnicas. Para poder repartir el procesador entre varios procesos, necesitamos tenerlos en memoria principal, por ello se divide ésta en trozos denominados **particiones** o **regiones,** donde se cargarán los diferentes procesos. El número de particiones indicará el grado de multiprogramación del sistema.

▪ Protección de la memoria

Si se encuentran simultáneamente varios procesos en memoria, debemos proteger sus respectivas particiones contra accesos no deseados.

El mecanismo de protección deberá, por tanto, actuar sobre cada referencia de memoria generada, siendo rápido en su gestión, pues de lo contrario, producirá una pérdida de tiempo importante. La rapidez justifica que se utilicen, para esta función, determinados registros hardware. Para cada partición se utilizan dos registros límite cuyos contenidos apuntarán a su parte superior e inferior, de tal manera que, cada dirección generada en el proceso que se trate deberá estar entre dichos límites (Figura 7.11).

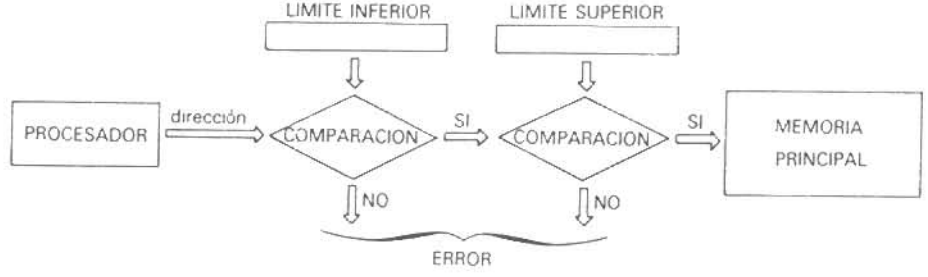


Figura 7.11. Protección de memoria por registros límite.

Esta técnica exige que las direcciones generadas por los procesos sean absolutas, asignadas bien durante la compilación o durante la carga del programa. En ambos casos la asignación será estática.

Otra solución más flexible, carga un registro con la dirección de comienzo de la partición y el otro con el tamaño de la misma, denominándose **registro base** y **registro límite.** En este caso es posible la asignación dinámica de direcciones ya que bastará actualizar el contenido del registro base para apuntar a otra zona de memoria. Cada dirección generada por el proceso deberá ser menor que el registro límite y se sumará el contenido del registro base para obtener así la dirección absoluta correspondiente (Figura 7.12).

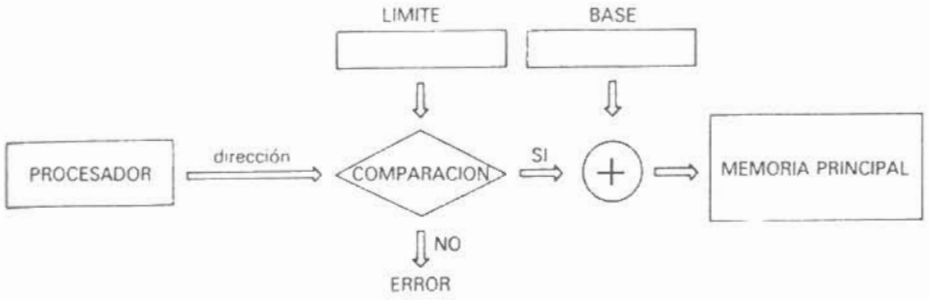


Figura 7.12. Protección de memorias por registros base y límite.

▪ Particiones contiguas de tamaño fijo

Se puede gestionar la memoria con particiones continuas de tamaño fijo, de tal forma que el número de ellas y sus tamaños se definirán al inicializar el sistema, quedando inamovibles para toda la sesión hasta que se apague el equipo. Por ejemplo, si tenemos una computadora de 640K de memoria principal, de las que 128K son utilizadas por el sistema operativo, quedarán 512K para programas de usuario. Se puede inicializar el sistema con tres particiones de 128K, 128K y 256K, quedando la memoria distribuida como muestra la Figura 7.13.

Figura 7.13. Particiones contiguas de tamaño fijo.

Cuando un programa tenga que cargarse para su ejecución, el sistema operativo le asignará una partición que pueda contenerlo, para ello es necesario que los programas declaren sus necesidades de memoria al sistema.

Las primeras técnicas que utilizaron este tipo de partición de la memoria realizaban la gestión por medio de colas de espera, de tal forma que al entrar un programa en ejecución, el sistema operativo le asignaba a una cola de acuerdo con la memoria que solicita (Figura 7.14).

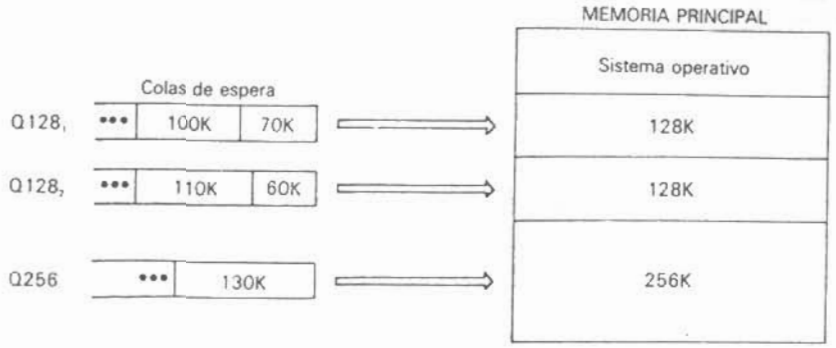


Figura 7.14. Particiones con colas de espera.

Esta técnica es simple y fácil de desarrollar, pero su eficacia está muy condicionada por las exigencias de memoria de los programas y su orden de llegada.

Para solucionar problemas que pueden aparecer en el método anterior, se puede utilizar otra técnica consistente en una sola cola de espera, y será el sistema por medio del **Planificador de Trabajos** el encargado de seleccionar qué programa cargar y en qué partición.

En este caso, el planificador de trabajos puede realizar la selección de varias formas:

- **Respetando el orden de llegada (FIGO-*First In, First Out*).** Método fácil de programar, pero provoca en ocasiones infrautilización de la memoria.
- **Seleccionar el trabajo de tamaño más adecuado a la partición libre** *(el más adecuado o Best Fit).* Se realiza analizando la cola de espera y eligiendo el de tamaño más adecuado a la partición libre.
- **Seleccionar el primer trabajo que queda en la partición disponible** *(el primero disponible o First Fit).* De características similares al anterior.

La Figura 7.15 muestra esta técnica de gestión.

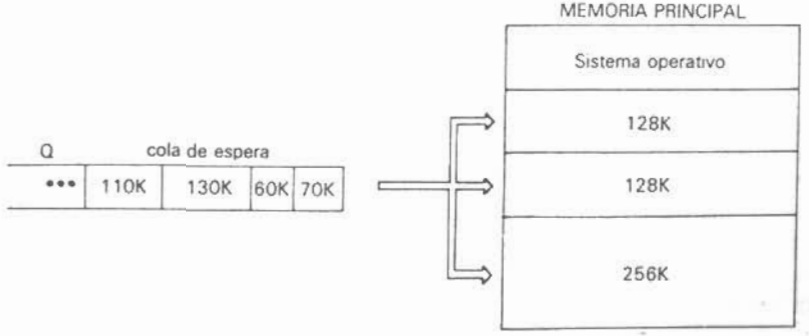


Figura 7.15. Particiones con cola de espera única.

Se mejora notablemente el rendimiento si se añade a la gestión de las particiones técnicas de intercambio, de tal forma que un programa pendiente de un suceso o interrupción puede ser sustituido por otro.

El desarrollo de esta técnica de gestión representó un notable avance respecto a esquemas anteriores. Sin embargo, su eficacia queda condicionada por la elección del tamaño y número de las particiones en relación con las características de los programas a tratar.

Si se analiza la utilización de las particiones, aparecen dos problemas originados por la propia estructura de las mismas que disminuyen la efectividad de la gestión, éstos son los siguientes:

- Si un programa necesita 100K de memoria y se encuentra en una partición de 128K, se desaprovechan 28K denominándose el suceso **fragmentación interna** (Figura 7.16).
- Por otra parte podemos tener un programa de 130K en la partición de 256K y otro de 100K en una de 128K, presentándose una petición por parte de un programa de 140K que no podrá ser cargado en la partición libre, mientras por otra parte existe, en teoría, memoria total suficiente. A este suceso se le denomina **fragmentación externa** (Figura 7.16).

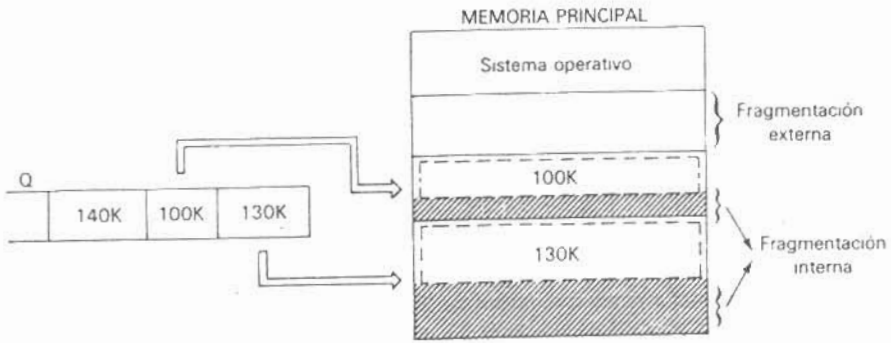


Figura 7.16. Fragmentación de la memoria.

▪ Particiones contiguas de tamaño variable

Los inconvenientes de la técnica anterior nacen de la dificultad de definir unas particiones de tamaño adecuado para todos los trabajos que se deben tratar. Este problema se afrontó desarrollando una nueva técnica de gestión que asigna dinámicamente la memoria a los trabajos de acuerdo con su tamaño.

En este caso, el sistema operativo mantendrá una tabla interna donde registrará las zonas de memoria disponible o huecos, asignando a cada trabajo una partición del tamaño solicitado. Esta técnica es la denominada de **particiones contiguas de tamaño variable** (Figura 7.17).

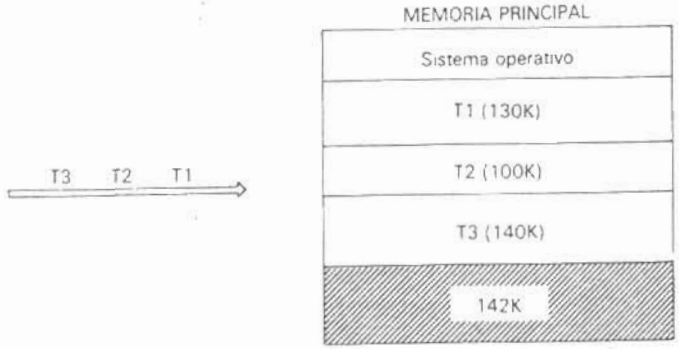


Figura 7.17. Carga de trabajos en particiones variables.

Cuando se arranca el sistema, a medida que van apareciendo trabajos en la cola, se van asignando las sucesivas particiones hasta que la memoria se completa, aunque siempre aparece una pequeña fragmentación externa *(parte final de la memoria)*. A medida que van terminando los distintos trabajos, el planificador va sustituyendo un programa por otro, de tal forma que van apareciendo huecos libres distribuidos entre las zonas ocupadas. El sistema asignará estos huecos mientras existan trabajos en espera que quepan en ellos. Lo hará buscando el primer trabajo que quepa en el hueco a asignar *(First Fit)* o aquel que se acople mejor *(Best Fit)*.

Cuando aparecen huecos contiguos, el gestor de la memoria tiende a unificarlos, por lo que en ningún momento existirán dos particiones contiguas vacías.

Con esta técnica, la memoria sufre un constante proceso de fragmentación. Para evitarlo, surge la idea de recolocar las zonas de memoria utilizadas para lograr cada cierto tiempo un único hueco de memoria disponible. Esta operación se denomina **compactación** y no es sencillo de realizar, pues existen diversas formas de recolocar varios programas en la memoria y con diferente coste. Además, es necesario que los programas sean reubicables para poder cambiar de zona de memoria principal (Figura 7.18).

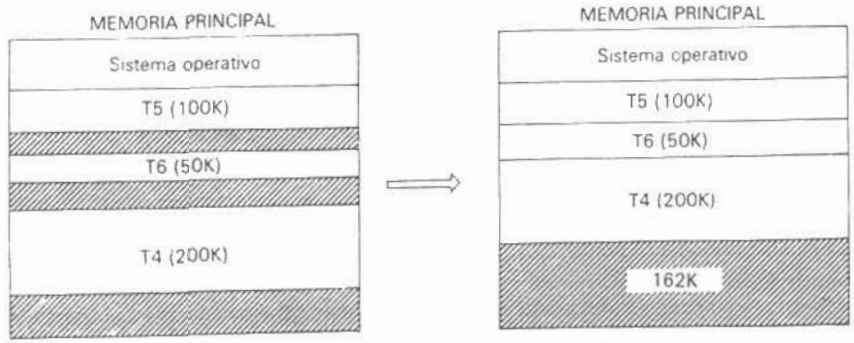


Figura 7.18. Compactación de trabajos.

Con las particiones variables se logra un mejor aprovechamiento de memoria que con las particiones fijas. Ahora bien, la fragmentación que sufre la memoria es su mayor inconveniente debido a la sobrecarga que se origina en el procesador para realizar la compactación con la consiguiente pérdida de tiempo.

7.4.3. Paginación

La necesidad de asignar memoria en cantidades contiguas para cada programa es el mayor inconveniente para un buen aprovechamiento de la misma. La paginación es una técnica de gestión que permite asignar la memoria de forma discontinua. Con este fin, se divide la memoria en trozos de tamaño fijo llamados **armazones** o **frames** y la lógica en bloques de igual tamaño denominados **páginas.** El sistema operativo mantiene internamente una tabla de páginas donde relaciona cada página cargada en memoria con el frame que la contenga, es decir, su dirección inicial en memoria real (Figura 7.19).

Figura 7.19. Paginación.

Cada dirección que genere el procesador será interceptada y dividida en dos componentes, un **número de página (p)** y un **desplazamiento en la página (d).** Utilizando **p** como índice, el gestor de memoria del sistema recorrerá la tabla de páginas hasta localizar dicha página, y a continuación sumará **d** a la dirección de carga correspondiente, obteniendo así la dirección real adecuada.

El tratamiento anterior exige que se realice por hardware, pues de no ser así, se introduciría un retraso excesivo (Figura 7.20).

Cada programa se divide en páginas, y éstas se cargan en frames libres que no tienen por qué ser contiguos. En general, un frame suele tener un tamaño potencia de 2 como es habitual, y éste se establece en el diseño de la arquitectura de la computadora (por ejemplo, la computadora 370 de IBM dependiendo del sistema operativo que utilice, tiene páginas de 2.048 o 4.096 bytes).

▪ Gestión de la memoria

La paginación es una forma de reasignar direcciones dinámicamente. El sistema analizará cada nuevo trabajo que se disponga a entrar para conocer el número de páginas que ocupa

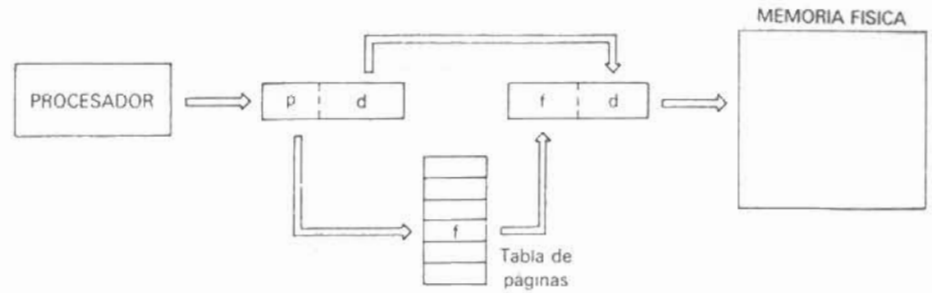


Figura 7.20. Hardware de paginación.

y buscará en su lista de frames libres un número igual de ellos. Si éstos existen, cargará en ellos las páginas del programa y construirá la correspondiente tabla de páginas, actualizando la lista de frames libres. Cada trabajo en memoria tendrá su propia tabla de páginas apuntada por el bloque de control del proceso.

De esta manera, se logra evitar la fragmentación externa ya que cualquier frame libre es asignable a un trabajo que lo necesite.

Por otro lado, seguirá existiendo fragmentación interna puesto que, en general, los trabajos no ocuparán un tamaño múltiplo del tamaño de la página. Esta fragmentación se reduce si la página tiene un tamaño pequeño, pero se necesitaría una tabla de páginas mayor.

▪ Rendimiento. Memoria cache. Registros asociativos

Para agilizar la conversión de direcciones, el sistema mantiene en memoria la tabla de páginas de cada uno de los trabajos activos, y utiliza un registro especial, llamado **registro base,** para indicar la dirección de la tabla de páginas del trabajo en ejecución en ese momento. De esta forma cuando se conmute de un trabajo a otro, se restaura la dirección de la tabla de páginas correspondiente.

Para transformar cada dirección lógica *(p,d)* generada por el procesador en su dirección real, el sistema accede en primer lugar a la tabla de páginas correspondiente *(Registro base + p)* y posteriormente a la dirección real *(frame + d).* De esta manera, cada dirección provoca dos accesos a memoria y se duplican los tiempos de ejecución. Con ello, hemos logrado mejorar la utilización de la memoria, sin embargo, la duración en los procesos es mayor.

Para solventar este problema, se recurre a memorias pequeñas de muy alta velocidad (también muy caras) donde se pueden mantener las entradas a las tablas de páginas más utilizadas. Esta memoria, conocida como memoria **cache,** es similar a la memoria principal pero con menor tiempo de acceso y con ello se logra reducir notablemente el retraso producido por la paginación.

En el caso de sistemas pequeños, con tablas de páginas de pocas entradas, se puede utilizar un conjunto de registros hardware para contener dicha tabla. Estos registros son los llamados **registros asociativos** (Figura 7.21).

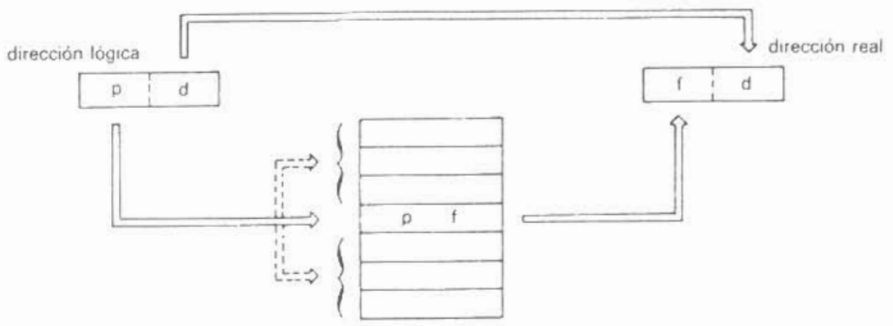


Figura 7.21. Registros asociativos.

Se define como **tiempo efectivo de acceso a memoria** al tiempo medio de acceso de todas las direcciones esté o no su entrada de página en la memoria cache.

▪ Páginas compartidas

La técnica de la paginación permite que varios procesos o usuarios compartan páginas cargadas en memoria. Esta posibilidad es especialmente positiva para los sistemas de tiempo compartido. Imaginemos un sistema con 20 usuarios que utilizan el programa editor de textos, en él se cargará una sola vez el programa correspondiente, y éste será utilizado por todos ellos aunque los datos sean propios de cada uno. Para ello, cada usuario contendrá en su tabla de páginas las entradas correspondientes al editor, apuntando todas ellas a los mismos frames y a distintos para los datos.

Puesto que es muy común utilizar páginas compartidas, el contenido de éstas debe permanecer inalterado. Las páginas que se deseen compartir deben tener código reentrante, es decir, que durante su ejecución no se altera ningún valor interno.

Si el contenido de las páginas compartidas no debe modificarse, debemos protegerlas contra todo intento de escritura. Necesitamos un sistema de protección de las páginas en memoria que no sólo controle las direcciones a las que se puede acceder, sino también el modo en que se realiza dicho acceso. Para ello, se añaden a las entradas de la tabla de páginas una serie de **bits de protección.**

El esquema de protección se complementa con otro bit por entrada de la tabla de páginas que se denomina **bit de validez.**

Con la paginación se produce una quiebra importante entre la visión que tiene el usuario de la memoria y la forma en que la utiliza. El usuario desarrolla su programa considerando la memoria como algo contiguo, pero éste se carga troceado o paginado en zonas discontiguas de la memoria real. El encargado de hacer corresponder el espacio lógico de direcciones *(continuo)* y el espacio físico de direcciones *(discontinuo)* es el hardware de traslación dinámica de direcciones.

7.4.4. Segmentación

La segmentación es una técnica distinta de gestión de memoria que pretende acercarse más al punto de vista del usuario.

Los programas se desarrollan, generalmente, en torno a un núcleo central *(principal)* desde el que se bifurca a otras partes *(rutinas)* o se accede a zonas de datos *(tablas, pilas, etc.).* Desde este punto de vista, un programa es un conjunto de componentes lógicos de tamaño variable o un conjunto de segmentos, es decir, el espacio lógico de direcciones se considera como un **conjunto de segmentos,** cada uno definido por su tamaño y un número (Figura 7.22).

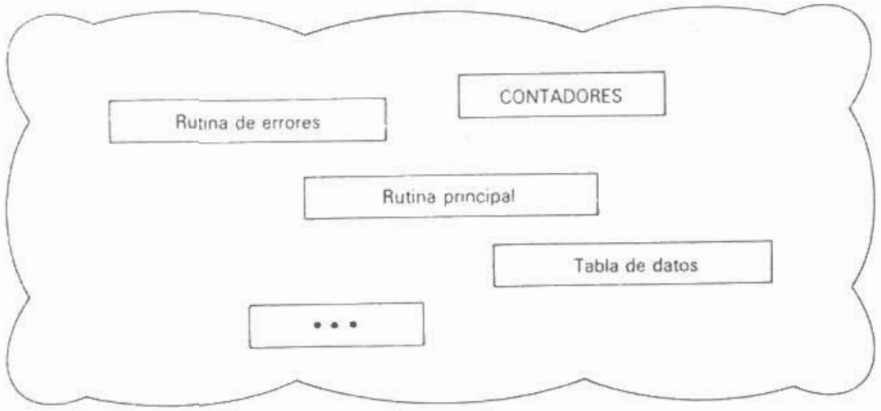


Figura 7.22. Espacio lógico de direcciones.

La segmentación de un programa la realiza el compilador y en ella cada dirección lógica se expresará mediante dos valores: **número de segmento (s)** y **desplazamiento dentro del segmento (d).**

▪ Hardware de segmentación

Puesto que la memoria física se direcciona linealmente, será necesario transformar cada dirección lógica *(s,d)* en una dirección real *(r).* Esta conversión la realiza un dispositivo especial de hardware consultando la **tabla de segmentos** correspondiente (Figura 7.23).

▪ Rendimiento

Esta técnica permite reducir la fragmentación interna de la memoria provocada por la paginación, ya que asigna a cada programa la cantidad de memoria que requiere.

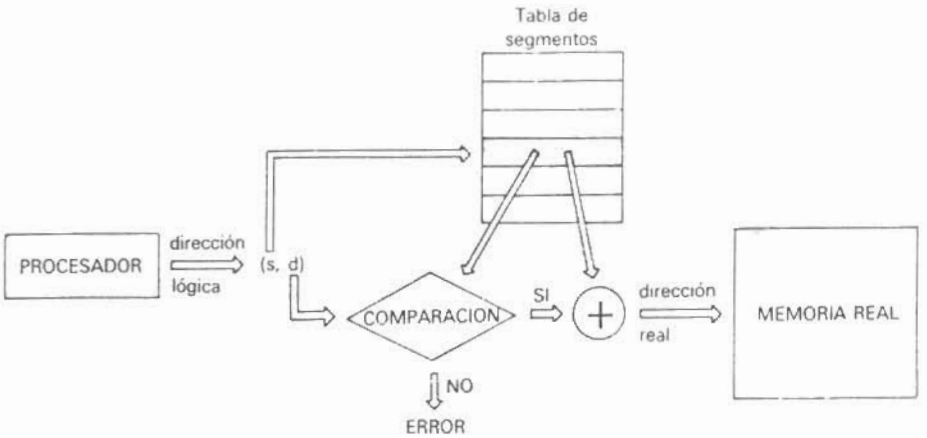


Figura 7.23. Hardware de segmentación.

La carga de un programa en memoria exige la búsqueda de los huecos adecuados a sus segmentos, y puesto que éstos son de tamaño variable, se ajustarán lo más posible a las necesidades, produciéndose huecos pequeños. En este caso se produce fragmentación externa.

La eficacia de la segmentación requiere, de igual forma que la paginación, el uso de memorias cache para lograr unos tiempos de acceso adecuados.

De igual forma que en la paginación, se pueden compartir segmentos entre varios procesos.

7.4.5. Sistemas combinados

Hemos visto que, tanto la paginación como la segmentación, tienen ventajas e inconvenientes, y parece lógico intentar combinar ambas técnicas para aprovechar sus caracteristicas positivas.

En todo sistema se define, por su arquitectura, un espacio máximo de direcciones lógicas, tamaño máximo de cualquier programa que quiera ejecutarse, e implica un número de páginas. Por esto, algunos sistemas *(Serie 370 de IBM)* utilizan una técnica de **paginación segmentada** consistente en segmentar la tabla de páginas adecuándola al tamaño del programa. Para ello, mantiene una tabla de segmentos cuyas entradas indican la dirección de inicio de cada tabla de páginas y su tamaño. Se utiliza un hardware especial (Figura 7.24).

Otros sistemas *(MULTICS, GE 645)* optan por paginar los segmentos, es decir, utilizan segmentos cuyo tamaño siempre es un número entero de páginas. Esta técnica se denomina **segmentación paginada.**

La tabla de segmentos de la segmentación pura cambia de contenido y sus entradas ya no apuntan al inicio del segmento correspondiente indicando su tamaño, sino que apuntan a una tabla de páginas del segmento indicando su longitud (Figura 7.25).

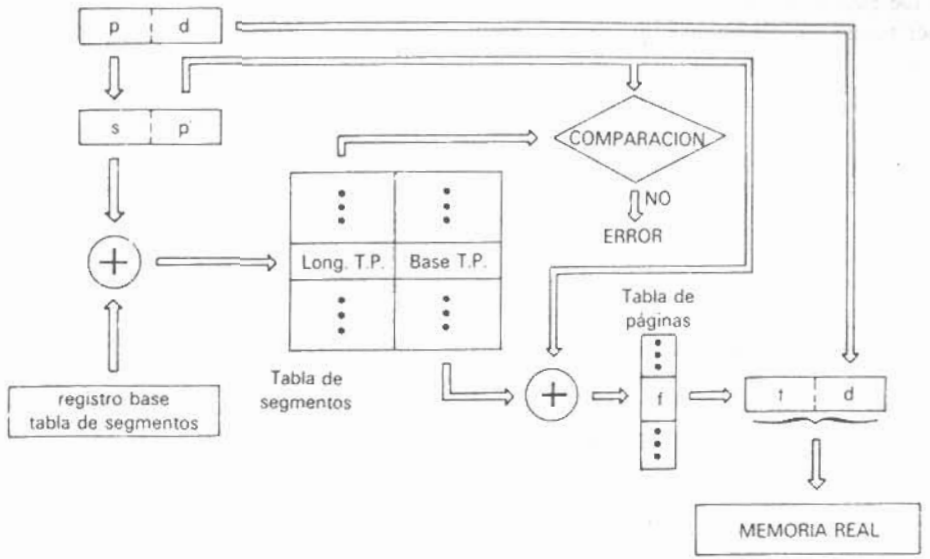


Figura 7.24. Hardware de paginación segmentada.

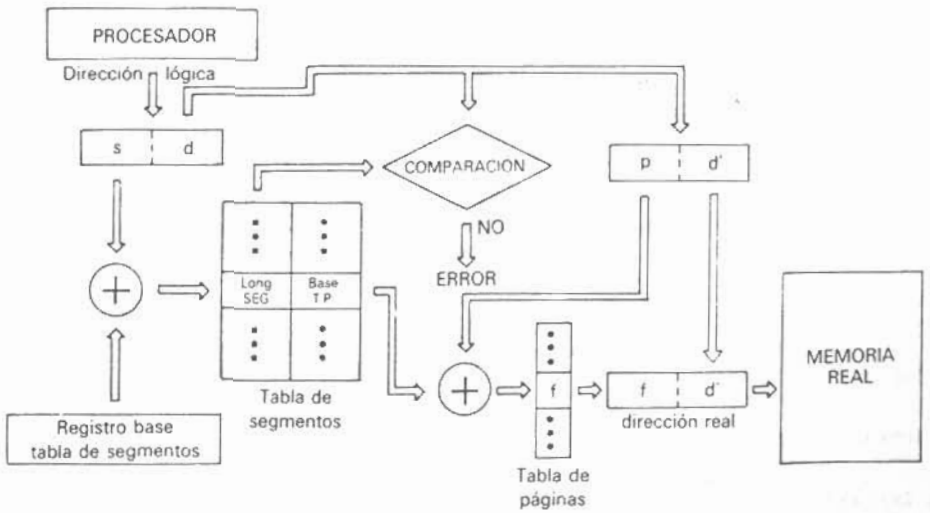


Figura 7.25. Hardware de segmentación paginada.

De esta forma se evita la fragmentación externa propia de la segmentación, pues cualquier hueco será de una página como mínimo, y por tanto utilizable para cualquier segmento que lo necesite.

7.4.6. Memoria virtual

Los sistemas de gestión de la memoria vistos hasta ahora están sujetos a la exigencia de que un programa, para ejecutarse, debe estar cargado en memoria principal en su totalidad. Sin embargo, no todas las partes del programa se ejecutan normalmente.

El programador diseña rutinas que sólo se ejecutan en determinadas circunstancias. Por ejemplo, aquellas que tratan situaciones de error. Con este enfoque no parece necesario que todo el programa esté cargado en memoria para poder ser procesado.

La **memoria virtual** es una técnica de gestión que, combinando hardware y software, permite la ejecución de programas parcialmente cargados en memoria real.

Esta forma de trabajar aporta ventajas importantes:

- Si los programas se pueden ejecutar por partes, la memoria lógica puede ser mayor que la real disponible.
- Puesto que cada programa ocupa menos memoria real, se puede elevar el índice de multiprogramación, y por tanto, la eficiencia del sistema.
- Al cargar menos cantidad de cada programa, se necesitan menos operaciones de entrada/salida para las operaciones de carga e intercambio de los mismos.

Las diferentes partes de un programa se van cargando en memoria a medida que se necesitan, y por ello, esta técnica debe considerar tres aspectos importantes:

- **Carga.** Las porciones del programa se cargan cuando se necesiten **(petición de página)** o bien se pueden cargar por adelantado **(anticipación** o **prepaginación).**
- **Colocación.** Los sistemas de memoria virtual que utilicen segmentación deben decidir, al cargar un nuevo segmento, si lo hacen en el hueco más adecuado o bien en el primero posible.
- **Sustitución.** Lo normal será que toda la memoria real esté ocupada, y cuando se necesite cargar una nueva parte de un programa habrá que reemplazar alguna de las existentes. Es importante definir la selección de la parte a reemplazar.

Esta técnica es difícil de llevar a la práctica. Los sistemas que la utilizan son complejos, pero a la vez, deben ser muy eficientes. De lo contrario, la ejecución de los programas puede empeorar notablemente.

▪ Carga por petición de páginas

Este es el esquema de carga más común cuando se utiliza la técnica de memoria virtual. Su funcionamiento es similar al de un sistema de paginación con intercambio (Figura 7.26).

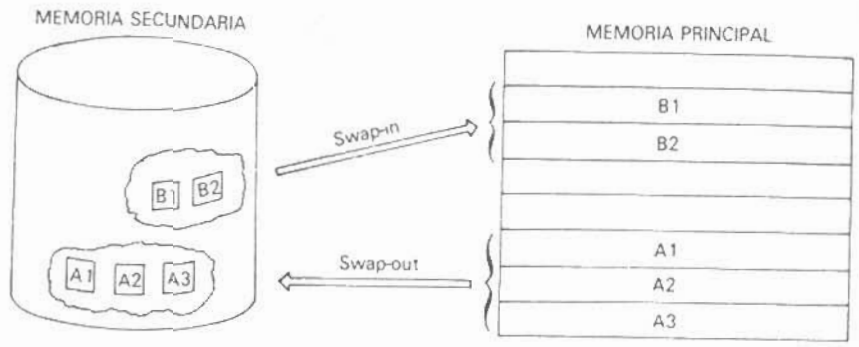


Figura 7.26. Páginas e intercambio.

Ahora el algoritmo de intercambio no llevará a memoria todo el programa, sino que sólo cargará aquellas páginas que se le pidan.

Cuando el procesador genere una dirección que pertenezca a una página que no se encuentre en memoria **«falta de páginas»,** la buscará, y a continuación, la traerá a memoria desde el correspondiente dispositivo de memoria secundaria.

La técnica de la memoria virtual se puede llevar a la práctica con el mismo hardware que la paginación de memoria real y un software más complejo.

El tratamiento de las faltas de página introduce un retraso adicional en la ejecución de los programas y afecta por tanto al rendimiento general del sistema. El diseño del software necesario debe ser especialmente cuidadoso en este aspecto.

▪ Reemplazamiento de páginas

La técnica de petición de páginas permite reducir, como ya se ha dicho, la memoria ocupada por cada programa durante su proceso.

Todos los sistemas que utilizan la técnica de memoria virtual sobreutilizan en mayor o menor medida la memoria física para obtener un mejor rendimiento. Pero para ello, deben hacerse algo más complejos, veamos por qué.

Supongamos que uno de esos programas intenta acceder a una dirección cuya página no está en memoria. El hardware la intercepta y cede el control al sistema operativo que, después de comprobar que es una dirección legal, la trata como una falta de página. Buscará la página correspondiente en el almacenamiento secundario y, una vez localizada, le buscará un hueco en memoria. En este punto al analizar la lista de frames libres comprueba que no hay hueco disponible y, como responsable de la gestión de la memoria, debe resolver la situación sin afectar al programa.

Una posible solución es sacar de memoria alguno de los programas cargados y utilizar los frames que deja libres, pero esto lleva consigo una sobrecarga importante en la operación de intercambio *(swap).*

La solución adoptada, generalmente, es sustituir alguna de las páginas cargadas *(reemplazar páginas)*. Para ello, la rutina del sistema que gestiona la interrupción de falta de página deberá trabajar de la siguiente manera:

- Encontrar la página solicitada en el almacenamiento secundario.
- Encontrar un frame libre.
- Si existe, utilizarlo.
- Si no existe, utilizar un algoritmo de reemplazamiento para seleccionar la página a reemplazar.
- Salvar la página reemplazada en el almacenamiento secundario, actualizando las tablas afectadas.
- Llevar la página solicitada al frame libre y actualizar las tablas correspondientes.

■ Algoritmos de reemplazamiento

El algoritmo óptimo será aquel que seleccione para su sustitución la página que vaya a tardar más en ser utilizada. Cuando se produce una falta de página, la memoria está ocupada por un grupo de páginas y una de ellas, la que provocó la falta, va a ser usada inmediatamente y, por tanto, no debe ser sustituida. De las otras, algunas se necesitarán al cabo de unas pocas instrucciones, otras al cabo de algunas más. Es evidente que la mejor solución es reemplazar la que tarde más instrucciones en ser necesitada, ya que su ausencia tardará más en hacerse sentir y durante ese intervalo pueden acabar otros procesos con la consiguiente liberación de frames.

La dificultad de esta solución radica en la imposibilidad de prever el comportamiento futuro de los procesos, pero como en otras muchas situaciones de la vida cotidiana, se puede prever, en cierta medida dicho comportamiento, partiendo de la experiencia del pasado, de las referencias a memoria que hasta el momento han realizado los procesos.

Los algoritmos que se han ideado para controlar el reemplazamiento parten de la utilización pasada de las páginas para aproximarse a la solución óptima. Su idoneidad quedará definida por dos factores: número de faltas de página que provoca y el coste de su utilización (la sobrecarga que produce en el sistema).

Un factor que condiciona el número de faltas de página es el número de frames disponibles en el hardware. En general, a más frames menos faltas (Figura 7.27).

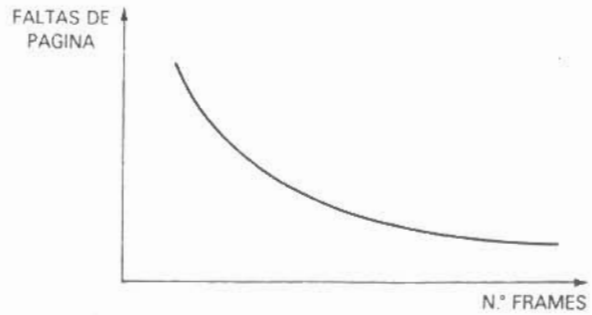


Figura 7.27. Número de frames y faltas de página.

■ Algoritmo de reemplazamiento FIFO

Este es el más sencillo. Cuando se necesite sustituir una página, se elegirá aquella que lleve más tiempo en memoria (primera en llegar, primera en salir-*First In, First Out*).

Para controlar el tiempo de permanencia en memoria de las páginas, este algoritmo utiliza una cola de llegada a memoria de las mismas. La primera de la cola será aquella que lleve más tiempo cargada y, según el criterio FIFO, será la primera en ser sustituida.

Un criterio de selección tan simple es fácil de programar y a la vez producirá poca sobrecarga en el sistema, pero su eficacia es relativa.

No es improbable que la razón por la que una página lleve más tiempo en memoria sea su mayor utilización, y en ese caso, este algoritmo sustituirá la página menos adecuada. Pensemos en un sistema de tiempo compartido con varios usuarios utilizando un editor de textos, compartiendo páginas del mismo. Normalmente serán estas las más antiguas en memoria y, aunque su utilización es muy alta, el criterio FIFO las reemplazará provocando inmediatamente una sucesión de falta de páginas.

Este algoritmo es inadecuado para sistemas de tiempo compartido, pero puede ser útil en entornos batch.

■ Algoritmo LRU

Este algoritmo es una buena aproximación a la solución óptima, considerando que aquellas páginas muy usadas en el pasado reciente lo serán también en el futuro. En el mismo sentido, las páginas poco utilizadas en el pasado seguirán siéndolo en el futuro, y se deberá sustituir aquella que haya sido menos usada recientemente *(Least Recently Used)*.

En este caso, la utilización del tiempo de uso es un criterio que refleja mucho mejor el comportamiento de los procesos en su utilización de la memoria.

Ponerlo en práctica es relativamente difícil, ya que el sistema debe contabilizar, de alguna forma, el tiempo en el que se produce cada referencia a memoria para poder clasificar las páginas según su uso. Este control se puede realizar de varias maneras:

- **Contadores de hardware.** La solución más simple es incorporar un contador de referencias a memoria y añadir un campo a las entradas de la tabla de páginas que pueda almacenar el valor del contador. Para servir una falta de página, el sistema busca el valor del contador registrado que sea menor.

- **Matrices de hardware.** En este caso, si la computadora dispone de N frames, el hardware debe mantener una matriz de N × N bits puestos inicialmente a cero. Al accederse a una página K, el hardware pone a 1 todos los bits de la fila K y a 0 todos los bits de la columna K. En cualquier momento, la página menos usada recientemente es aquella cuya fila tenga el menor valor binario.

- **Pilas.** Otra alternativa es mantener una pila de los números de las páginas utilizadas. Cada vez que se utiliza una página, su número se coloca al principio de la pila. En un momento determinado la página menos recientemente utilizada será la del fondo de la pila.

En todas estas opciones del algoritmo LRU se exige disponer de un hardware especial, y por ello, su aplicación se limita a aquellas computadoras que dispongan de los elementos necesarios.

▪ Otros algoritmos

Dadas las dificultades del algoritmo LRU, se ha tratado de encontrar algoritmos más sencillos y menos costosos de realizar.

Una práctica bastante extendida es la utilización de un **bit de referencia** asociado a cada entrada de la tabla de páginas. Cada vez que se utiliza una página, el hardware activa el bit correspondiente. Conforme se van utilizando las distintas páginas cargadas en memoria se activan sus bits de referencia y, en un instante dado, aquellas cuyos bits estén a cero, serán candidatas a ser reemplazadas. En general, los bits de referencia se restauran cada cierto tiempo *(puesta a cero)*.

Otro algoritmo es el denominado **menos frecuentemente usado** *(LFU-Least Frequently Used)*. Consiste en asociar un contador a cada página, de tal forma que, a intervalos regulares, se produce una interrupción que pasa control al sistema y éste añade al contador de cada página el valor del bit de referencia, poniendo después a 0 todos los mencionados bits de referencia. Cuando se produce una falta de página se sustituirá aquella cuyo valor del contador sea menor.

En ocasiones, algunos algoritmos se complementan utilizando un **bit de modificación** por cada página, indicando si el contenido de la misma ha variado o no desde que se cargó en memoria. Se deben sustituir preferentemente aquellas páginas que no se hayan modificado, pues así se evita tener que salvar su contenido en el almacenamiento secundario, con el consiguiente ahorro en operaciones de entrada/salida.

7.4.7. Criterios de reemplazamiento de páginas

Si se produce una falta de página en nuestro sistema, cualquiera de los algoritmos analizados nos puede ayudar, con mayor o menor efectividad, a seleccionar la página a sustituir. Pero, ¿entre qué páginas elegimos?, ¿las del propio proceso o entre todas las existentes? En el primer caso hacemos una **selección local** o **por proceso,** y en el segundo, lo haremos **global** o **general.**

En la Figura 7.28 tenemos tres procesos: A, B y C, ocupando la memoria con una serie de páginas. Al lado de cada una de ellas se especifica la edad o tiempo que cada una lleva en memoria. Supongamos que el proceso A necesita traer otra de sus páginas a memoria produciendo la correspondiente falta de página. Si aplicamos el criterio local de selección, la página sustituida será la tercera del proceso A(A3), y si se aplica el criterio global se sustituirá la tercera de B(B3).

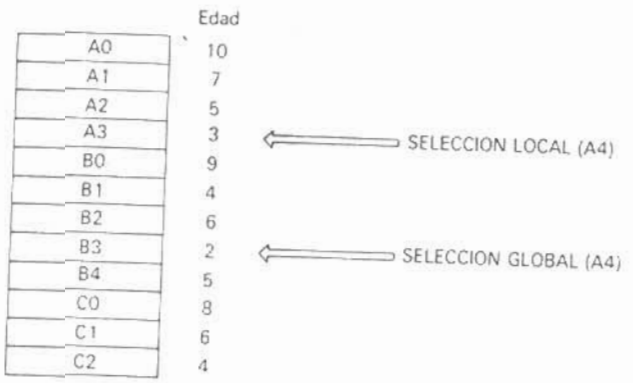


Figura 7.28. Reemplazamiento local y general.

7.4.8. Asignación de memoria

En el diseño de un sistema operativo se debe afrontar otro aspecto importante de la gestión de la memoria como es determinar cuántos frames asignar a cada proceso activo. La propia arquitectura de una computadora impone unos mínimos de asignación por proceso, por tanto, el propio sistema debe asegurar a todo proceso un número mínimo de páginas en memoria *(mínimo estructural)*.

Si por cualquier razón un proceso se queda con menos frames de los indicados, el sistema deberá suspenderlo y sacarlo *(swap-out)*.

Aunque en principio el número de frames asignados a un proceso podrá reducirse hasta ese mínimo, de hecho, cualquier proceso en un momento dado está utilizando un número mayor. Suponiendo que por su prioridad u otra razón, un proceso no pueda robarle frames a otros procesos, deberá sustituir una de sus páginas activas provocando, casi inmediatamente, otra falta de página, y así sucesivamente. A esta situación se le denomina **hiperpaginación** o **trashing** en la que se consume más tiempo paginando que procesando.

▪ Localidad de los procesos

Para evitar situaciones como las descritas, es necesario asignar a cada proceso el número de frames que necesite en cada momento de su ejecución. El problema radica en cómo conocer ese número.

Afortunadamente, el comportamiento de los procesos permite afrontar el problema, ya que todo programa está compuesto por varias partes o subrutinas de forma que su ejecución se realiza por fases, una detrás de otra. En cualquier momento el proceso direcciona sólo una pequeña parte de sus páginas. Este comportamiento se conoce con el nombre de **localidad de referencias de los procesos,** y con su ayuda, podemos conocer el número mínimo de frames que requiere un proceso para evitar una situación de hiperpaginación.

■ **Frecuencia de falta de página**

Una forma directa de controlar la asignación de memoria y evitar situaciones no deseadas se basa en el control de la frecuencia con que un proceso produce faltas de página. Puesto que la hiperpaginación se caracteriza por un gran número de dichas faltas, el objetivo es mantener la frecuencia de las mismas entre unos límites preestablecidos (Figura 7.29).

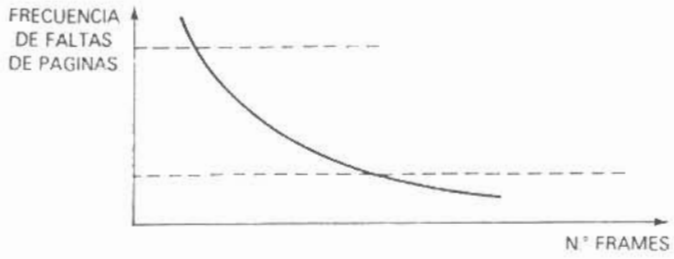


Figura 7.29. Frecuencia de faltas de página.

Si la frecuencia de faltas de página de un proceso sobrepasa el límite superior, el sistema le asignará más frames, y si cae por debajo del límite inferior, indica que al proceso se le pueden reducir el número de frames asignados.

7.5. CONSIDERACIONES DE DISEÑO

Diseñar el componente de gestión de la memoria de un sistema operativo no queda limitado a elegir los algoritmos más adecuados para la asignación de frames y reemplazamiento de páginas, hay otros factores que deben tenerse en cuenta para lograr un funcionamiento efectivo.

Uno de esos factores es el **tamaño de las páginas,** que afecta notablemente al rendimiento del sistema. Este factor está limitado por las características constructivas de la propia computadora y decidir su tamaño adecuado no es sencillo puesto que hay que equilibrar diversos factores:

- **Tabla de páginas.** Estas se encuentran en memoria principal y es necesario reducir al máximo su tamaño, lo que nos condiciona a diseñar páginas grandes (tablas con pocas filas).
- **Fragmentación.** El tamaño de los programas no suele ser un múltiplo exacto de páginas, lo que nos produce la ya mencionada fragmentación de la memoria. Ello nos recomienda páginas pequeñas.
- **Transferencia.** Llevar una página del almacenamiento secundario a memoria y viceversa es una operación larga, y el factor más determinante es el tiempo de latencia y búsqueda del dispositivo. Desde este punto de vista, es bueno que las páginas sean grandes para transferir más bytes en cada operación de entrada/salida.

- **Faltas de página.** A mayor tamaño de página se producirán menos faltas de página, reduciéndose la sobrecarga que conllevan.

El tamaño de las páginas es siempre una potencia de 2, variando entre 512 y 4.096 bytes. Cada fabricante optará por el más adecuado a los objetivos que se pretenden.

La gestión de la memoria debe posibilitar también compartir páginas entre procesos y, para ello, dichas páginas deberán llevar un indicativo que les asegure un tratamiento especial.

7.6. TENDENCIAS ACTUALES

El desarrollo tecnológico actual ha permitido reducir considerablemente el coste de las unidades de memoria, y gracias a ello es posible disponer hoy de memorias de altas prestaciones a costes aceptables. Consecuentemente, las exigencias de diseño impuestas por el tamaño de la memoria en el pasado han desaparecido.

Estas nuevas técnicas permiten no sólo tener memorias más grandes y de costos más bajos, sino también más rápidas, permitiendo cambios notables en el diseño de los nuevos sistemas operativos (mayores capacidades de direccionamiento, uso elevado de memorias cache, etc.).

Las computadoras pequeñas y medias actuales ya tienen memoria de varios megabytes, por lo que permiten sistemas operativos mayores como son, por ejemplo, UNIX y OS/2. En los de mayor tamaño, la tendencia está marcada por la introducción de niveles de memoria ultrarrápida para almacenar el código más utilizado.

La utilización de estas nuevas facilidades del hardware han obligado a desarrollar nuevas técnicas de gestión de la memoria principal y, en definitiva, nuevos sistemas operativos.

CUESTIONES

1. ¿Por qué se hace necesario compartir la memoria entre varios programas?
2. ¿Cuáles son y qué representan los dos parámetros relacionados con la lectura y escritura de datos en la memoria principal?
3. ¿Qué tipos de computadoras existen según el tamaño de la memoria a la que se accede en cada operación de lectura o escritura?
4. En un sistema informático, ¿qué tipos de memoria suelen existir y cuál es su jerarquía?
5. ¿Por qué motivo el sistema operativo dedica gran parte de su software a la gestión de la memoria principal?
6. Realizar un esquema de los métodos de gestión de memoria relacionados con los sistemas monoprogramados.
7. Realizar un esquema de los métodos de gestión de memoria relacionados con los sistemas multiprogramados.
8. ¿En qué consiste y cuál es el fin de la paginación de la memoria?
9. ¿Qué es una memoria cache?
10. ¿En qué consiste y cuál es el fin de la segmentación de la memoria?
11. ¿Para qué se utiliza la técnica de gestión de la memoria denominada de memoria virtual?
12. En la gestión de memoria virtual, ¿qué es y cuándo se produce lo que se denomina petición de página?
13. Comentar algunos criterios de reemplazamiento de páginas utilizadas en algoritmos actuales.
14. ¿Qué consideraciones son necesarias para el diseño de la gestión de memoria en un sistema operativo?
15. ¿Cuáles son las tendencias actuales relacionadas con la memoria principal y su gestión?

CAPITULO 8

Gestión de entrada/salida

8.1. INTRODUCCION

El control de las operaciones de entrada/salida es otra de las misiones que debe realizar un sistema operativo para facilitar el uso de los distintos dispositivos que forman parte de un sistema informático.

Los dispositivos hardware de la computadora cuya misión es la de intercambiar datos con el procesador y la memoria principal en un sentido, en otro o en ambos, comúnmente denominados **periféricos,** no son fáciles ni cómodos de utilizar directamente por los procesos. Por otro lado, los procesos no necesitan conocer las peculiaridades ni características de dichos dispositivos, sino únicamente intercambiar datos con ellos. Por tanto, estos detalles deben ser ocultados para que de este modo las operaciones de entrada/salida sean independientes del tipo y modelo del dispositivo (Figura 8.1).

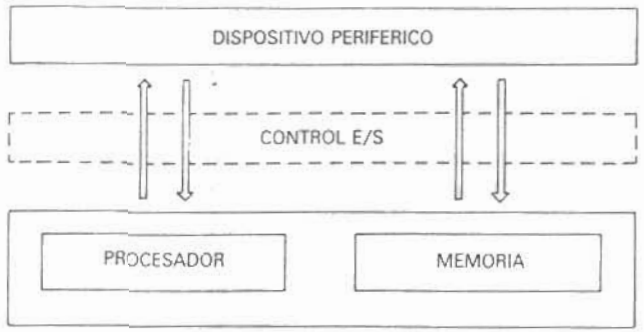


Figura 8.1. Gestión de operaciones de entrada/salida.

En general, el software de gestión de las operaciones de entrada/salida que posee un sistema operativo representa, aproximadamente, el 50 por ciento del total, por lo que esta parte es de suma importancia, más para el diseñador y programador de sistemas que para los programadores de aplicaciones y usuarios, los cuales necesitan, al menos, un conocimiento global de dicha gestión.

Este capítulo comienza con una breve descripción de los dispositivos hardware comúnmente utilizados, para pasar a continuación a la descripción del interfaz que permite la conexión entre dispositivos y la computadora. Seguidamente se describe la gestión de estos dispositivos por el sistema operativo a través del software de control, comúnmente denominado **driver** del dispositivo. Finalmente, se aborda el interfaz que el sistema operativo ofrece a los procesos de usuario para la ejecución de las operaciones de entrada/salida.

8.2. DISPOSITIVOS HARDWARE

A lo largo de la historia de las computadoras se han desarrollado muchos tipos de dispositivos que podemos reunir en tres grandes grupos:

- **Dispositivos de almacenamiento.**
- **Terminales.**
- **Comunicaciones.**

A su vez, los dispositivos se pueden clasificar, atendiendo al tipo de información que manejan y cómo lo hacen, en los siguientes grupos:

- **Dirigidos a bloques.** Tratan la información en bloques de tamaño fijo (256 a 1.024 bytes). Su característica principal es que se puede leer cada bloque como una unidad independiente de las demás (por ejemplo los discos).
- **Dirigidos al carácter.** Entregan o aceptan cadenas de caracteres sin tener en cuenta ninguna estructura prefijada. No son direccionables ni pueden realizar operaciones de búsqueda (por ejemplo un terminal).

8.2.1. Dispositivos de almacenamiento

▪ Discos

Son los dispositivos para almacenamiento secundario más comunes. Aceptan y recuperan datos a alta velocidad (superior a 2 Mbits/seg.). Los datos son transferidos entre el disco y la memoria en bloques.

Los discos pueden ser fijos y removibles. Los primeros no se pueden cambiar, mientras que los segundos pueden ser intercambiados, con lo que se consigue un mayor volumen de almacenamiento.

Pueden reunirse varios discos en un paquete **(disk pack)** para conseguir con ello un mayor almacenamiento en un reducido espacio. Todos los discos de un disk pack giran a la misma velocidad (aproximadamente 60 vueltas/seg.). Cada disco se divide en pistas concéntricas de tal forma que todas las homólogas de los distintos discos forman lo que se denomina **cilindro.** Cada pista, a su vez, se divide en **sectores,** y cada sector contiene un **bloque de información.**

Los discos flexibles **(floppy disk)** son similares, pero sólo constan de un disco y son siempre removibles. Estos discos giran a menos velocidad, y sólo lo hacen cuando necesitan realizar una transferencia de información.

Cada sistema operativo necesitará conocer cómo es el formato que tiene el disco, es decir, cuántos sectores tiene cada pista y cuántos bytes pueden ser almacenados en un sector. Si un sector es detectado como incorrecto, se marca como tal, y el sistema operativo no vuelve a acceder al mismo.

La lectura y escritura en un disco se realiza gracias a una cabeza magnética que se mueve a lo largo de un radio del disco, hacia el centro o hacia la periferia, en un movimiento conocido como **búsqueda** *(seeking).* Las órdenes de magnitud del tiempo invertido en esta operación oscilan entre 20 y 80 mseg. En el caso de los disk pack existe una cabeza por cara de cada disco, de manera que el movimiento de las mismas es simultáneo, recorriendo la misma pista de cada disco en el mismo instante (Figura 8.2).

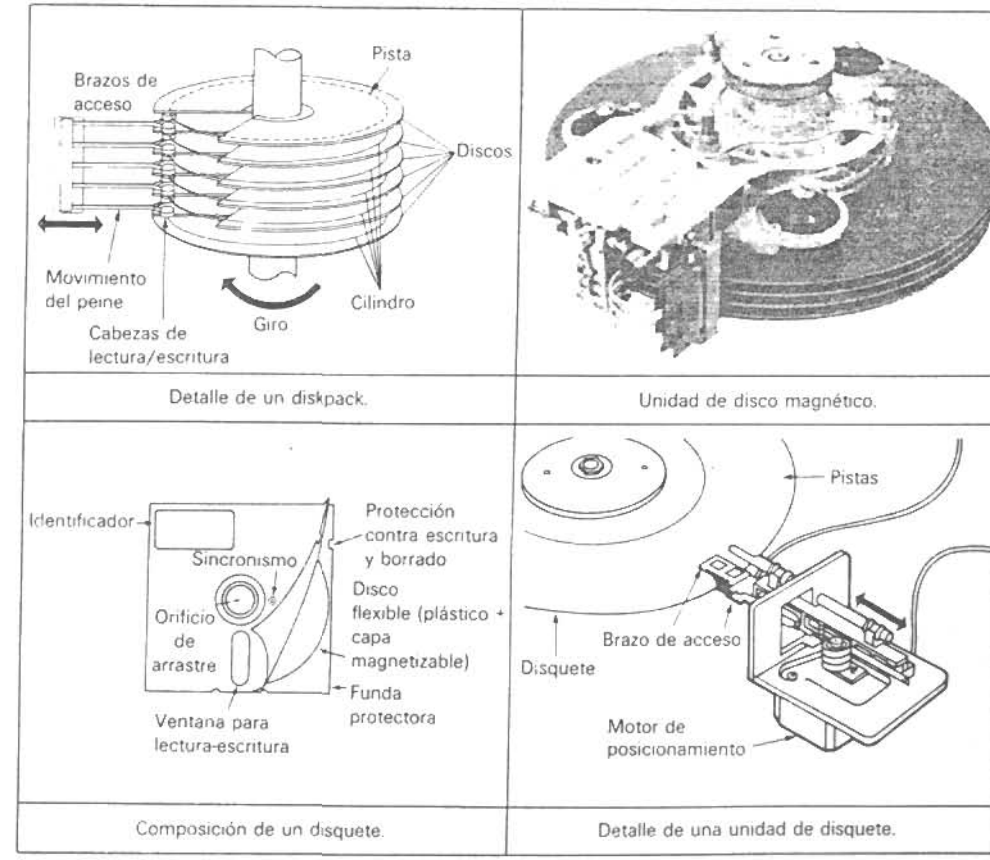


Figura 8.2. Discos.

▪ Cinta magnética

Se destinan principalmente al almacenamiento de grandes archivos de datos y a copias de seguridad periódicas.

Al igual que los discos, se trata de un medio magnético de almacenamiento consistente en una cinta plástica recubierta por una cara de una fina capa de óxido magnetizable. Generalmente, se denomina **trama** *(frame)* a un conjunto de nueve bits de los cuales 8 se destinan para byte de datos y el noveno se conoce como bit de paridad, para el control de errores.

El empaquetamiento de los datos se realiza a densidades que oscilan entre 800 y 6.250 tramas por pulgada de cinta. La densidad de una cinta es constante en toda ella.

Las tramas se agrupan en registros separados por **huecos** *(GAPS)* y, a su vez, los registros se agrupan en **archivos** *(files)* separados por huecos mayores que los anteriores.

La transferencia de datos a/o desde cinta debe ser realizada a través de **buffers** de tamaño adecuado.

Muchos sistemas operativos graban un registro inicial en la cinta incluyendo el número de serie, propietario y otras informaciones necesarias para el sistema operativo. Este registro se conoce como **etiqueta** *(label)*.

La lectura de las cintas se puede hacer en los dos sentidos: avance y retroceso (Figura 8.3).

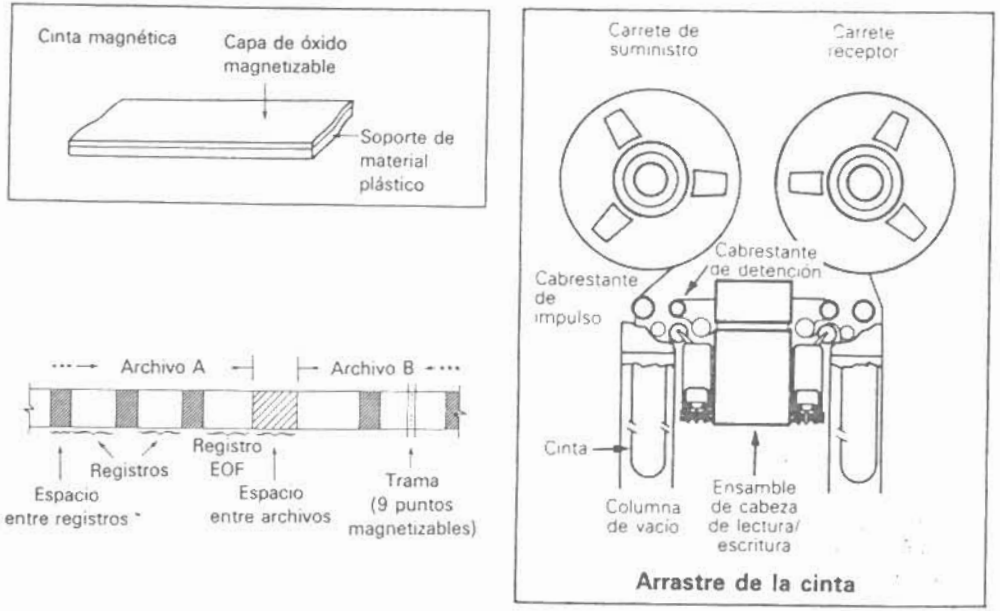


Figura 8.3. Cinta magnética.

▪ Tambores magnéticos

Los tambores se suelen emplear para el intercambio o swapping. Tienen forma cilíndrica y están divididos en pistas circulares, cada una de las cuales tiene su propia cabeza de lectura/escritura.

El tambor está constantemente girando a una velocidad aproximada de 3.000 revoluciones/minuto.

Actualmente, y debido a su tamaño y precio, han caído en desuso, siendo su principal sustituto el disco magnético (Figura 8.4).

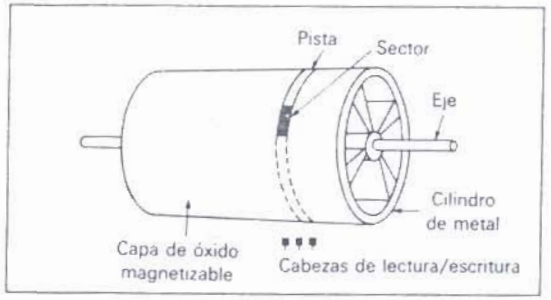


Figura 8.4. Tambor magnético.

8.2.2. Terminales

En general, se denomina **terminal** al conjunto formado por un teclado y una pantalla conectados a la computadora para introducir datos a través del primero y recibirlos a través de la segunda (Figura 8.5).

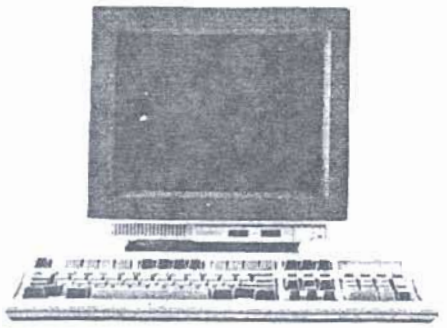


Figura 8.5. Terminal teclado-pantalla (Cortesía IBM).

Los terminales pueden dividirse en dos categorías: los que se conectan a través del estándar RS-232, y los mapeados en memoria.

■ **Terminales RS-232**

Constan de teclado y pantalla que transmiten bit a bit en serie. La velocidad de transmisión de estos bits viene dada en **baudios** (bits/seg.), siendo valores utilizados: 300, 1.200, 2.400, 4.800 y 9.600.

Estos terminales se conectan a la computadora a través de un cable físico. Este cable en su entrada a la computadora termina en una tarjeta hardware o interfaz que a su vez se conecta al bus de dicha computadora.

Se conocen, generalmente, con el nombre genérico **TTY.**

■ **Terminales mapeados en memoria**

No necesitan línea de conexión a la computadora ya que están directamente conectados al bus del mismo. En estos terminales, el teclado se conecta directamente al bus y es independiente de la pantalla.

8.2.3. Líneas de comunicaciones

Son dispositivos cuya misión es la de conectar entre sí computadoras y éstas con sus terminales cuando la distancia es grande. Suelen llevar consigo elementos físicos, como puede ser la propia línea, y los adaptadores a la línea denominados **modems,** y lógicos, como pueden ser el **protocolo de comunicación** y el **método de control y detección de errores** que se utilice (Figura 8.6).

Las líneas de comunicaciones pueden ser de varios tipos:

- **Síncronas o asíncromas.** Según se transmita la información entre dispositivos de forma sincronizada *(reloj)* o no.

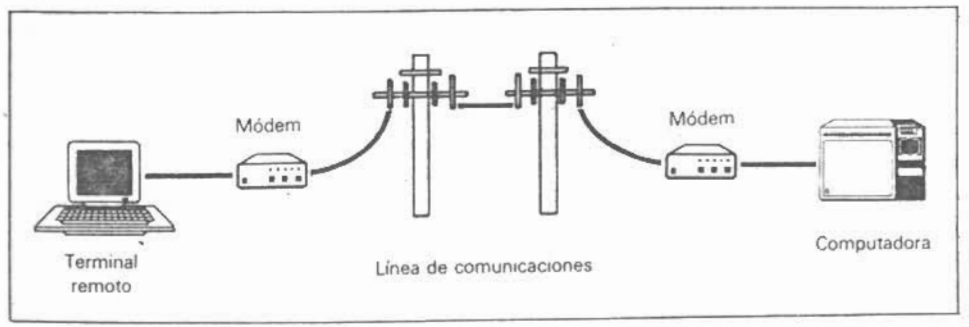


Figura 8.6. Línea de comunicaciones.

- **Según el sentido de la transmisión.**
  - **Simplex** cuando los datos se transmiten en una sola dirección.
  - **Half-duplex** si los datos pueden transmitirse en ambas direcciones pero no simultáneamente.
  - **Full-duplex** cuando los datos pueden transmitirse en ambas direcciones simultáneamente.

8.3. INTERFAZ PROCESADOR-PERIFERICO

La velocidad y complejidad de los periféricos determinan cómo deben ser conectados al procesador. Estudiaremos tres tipos de conexión:

■ **Registros**

Los dispositivos se pueden conectar al procesador por los registros de los dispositivos que pueden ser accedidos directamente en una zona determinada de la memoria o, indirectamente, por medio de instrucciones hardware que devuelven el estado del mismo.

Estos registros tienen cuatro misiones:

- Transferir el estado del dispositivo *(status).*
- Transferir instrucciones al dispositivo.
- Transferir datos desde el dispositivo.
- Transferir datos al dispositivo.

Ahora bien, el procesador sólo puede dar comienzo a las operaciones de entrada/salida sin poder controlar su terminación, pero para ver cuándo se ha completado una operación se pueden emplear dos métodos:

- **Polling.** Consiste en leer constantemente el registro de status del dispositivo. Tiene el inconveniente de ocupar el procesador un tiempo no deseado.
- **Interrupciones.** El procesador continúa con otros trabajos y sólo cuando el dispositivo concluye la operación llama la atención del procesador, interrumpiéndole para que trate dicha situación y realice las acciones que considere necesarias, es decir, sirva la interrupción.

■ **Controladores**

Los dispositivos complejos (discos...) no se conectan directamente al procesador, sino que lo hacen a través de un controlador que contiene el estado del dispositivo *(status),* controla el mismo y chequea los datos transferidos.

El controlador es el que acepta las órdenes del procesador y se comunica con él a través de registros como si se tratara de un dispositivo.

El controlador, también llamado unidad de control, puede manejar varios dispositivos del mismo tipo (Figura 8.7).

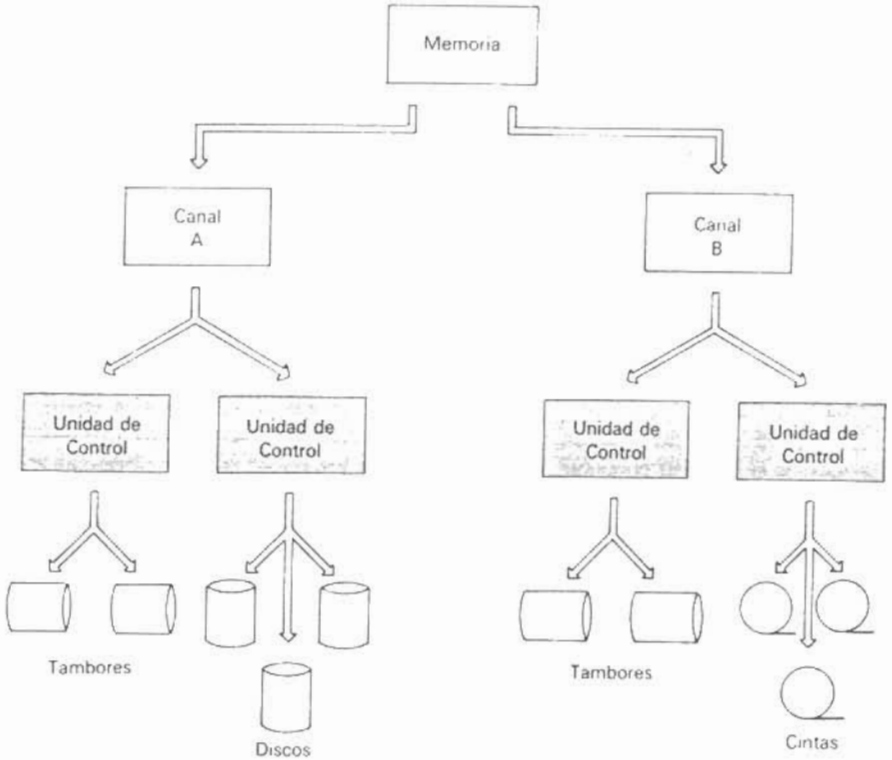


Figura 8.7. Controladores de dispositivos.

■ **Canales**

Normalmente los controladores se conectan al procesador a través de un **canal** o **procesador de entrada/salida** *(IOP-Input Output Processor)*. El propósito de un canal es conseguir que los dispositivos sean tratados como virtuales, abstractos o transparentes.

Los canales son manejados por comandos y cuando terminan la operación devuelven el status correspondiente e interrumpen al procesador.

Los canales pueden ser de varios tipos:

- **Selectores.** Pueden manejar varios dispositivos, pero sólo pueden transferir datos de uno en uno.
- **Multiplexores.** Manejan varios dispositivos y pueden transferir datos simultáneamente (Figura 8.8).

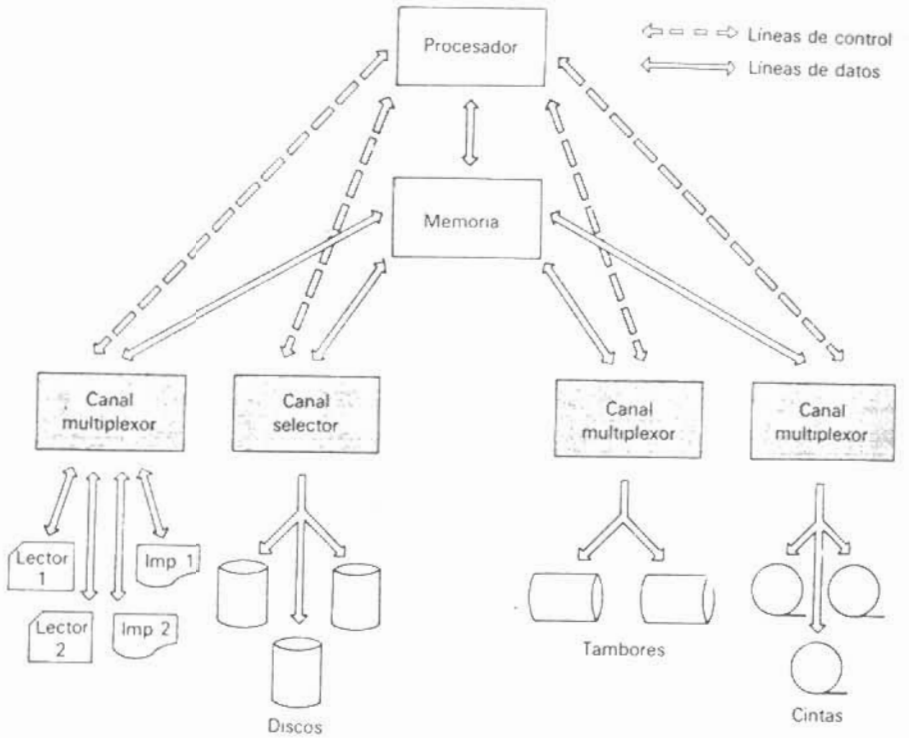


Figura 8.8. Canales en una computadora.

8.4. SOFTWARE DE CONTROL DE ENTRADA/SALIDA (DRIVER)

Se define driver como «el software formado por un conjunto de rutinas y tablas que, formando parte del núcleo del sistema operativo, ejecutan y controlan todas las operaciones de entrada y salida sobre cualquier periférico conectado a la computadora, siendo particulares para cada dispositivo».

Un driver no es un proceso o tarea independiente gestionado por el sistema operativo, sino un conjunto de tablas en las que se aloja la información que caracteriza a cada periférico conectado a la computadora, y una serie de rutinas que controlan toda la gestión de los mismos y las informaciones que fluyen en un sentido o en otro. Se encuentran permanentemente alojados en memoria principal y requieren una elevada rapidez de ejecución sin formar parte del proceso de usuario que los utilice (Figura 8.9).

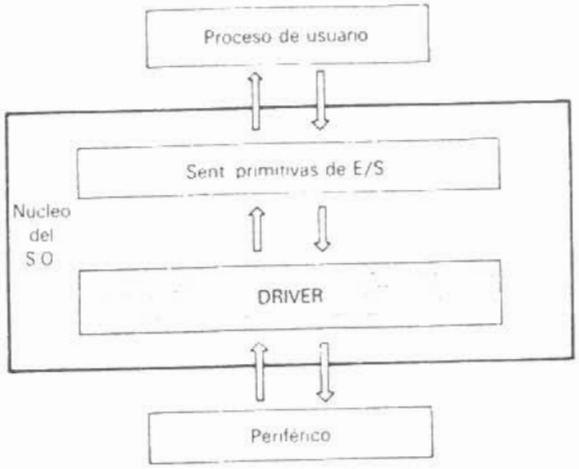


Figura 8.9. Driver de entrada/salida.

El tratamiento por el núcleo de un sistema operativo de toda la información de entrada/salida desde, o a un periférico, se puede dividir en dos niveles para su estudio:

- **Tratamiento independiente del periférico**

Está formado por el conjunto de rutinas que procesan información sin atender a las características propias del periférico.

- **Tratamiento dependiente del periférico**

Es el conjunto de rutinas que el núcleo del sistema operativo ofrece para controlar el propio dispositivo periférico.

El proceso de gestión de operaciones de entrada/salida se encuentra representado en la Figura 8.10.

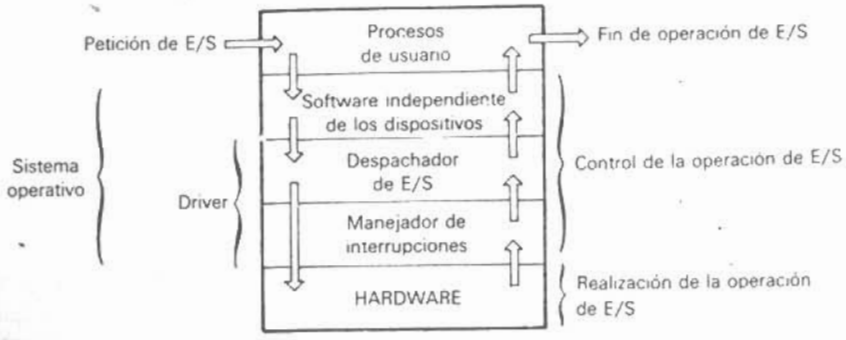


Figura 8.10. Gestión de operaciones de entrada/salida.

Sin entrar en detalles, el proceso de entrada/salida queda representado esquemáticamente en la Figura 8.11.

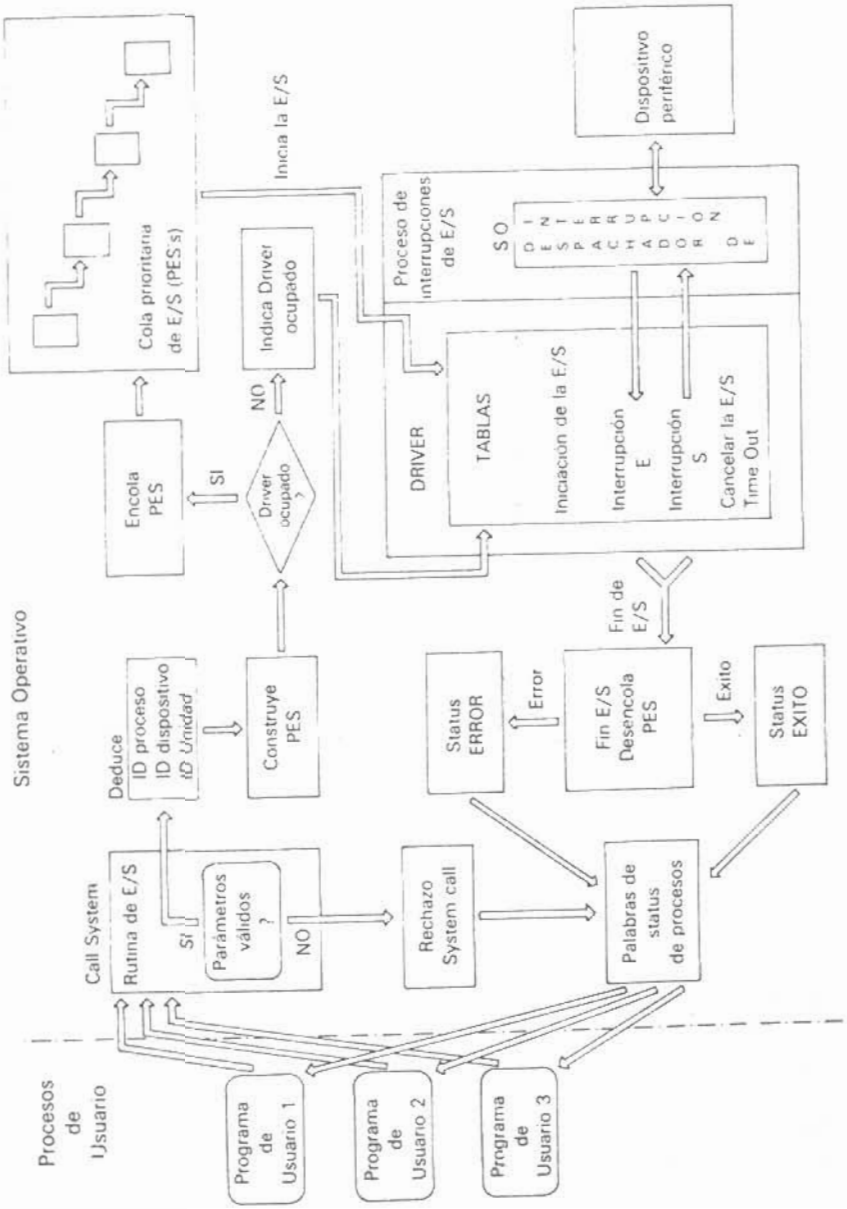


Figura 8.11. Proceso de entrada/salida.

8.4.1. Funciones de un driver

Entre las funciones que realiza un driver podemos citar las siguientes:

- **Definir las características del periférico** al resto del sistema operativo.
- **Inicializar los registros** asociados al periférico en el momento del arranque *(bootstrap)* del sistema operativo.
- **Habilitar y deshabilitar el dispositivo** para un proceso.
- **Procesar** todas **las operaciones** de entrada/salida solicitadas por un proceso.
- **Cancelar** toda operación de entrada/salida en el momento que sea necesario por cualquier motivo.
- **Procesar** todas las **interrupciones** hardware generadas por el propio periférico.
- **Tratar** los **errores** y estado del dispositivo haciendo la correspondiente comunicación al usuario.

8.4.2. Rutinas de un driver

Son los puntos de entrada al driver y pueden ser llamadas directamente por el núcleo del sistema operativo o por una interrupción hardware del dispositivo periférico. En general, en un driver podemos encontrar las siguientes rutinas (Figura 8.12):

- **Inicialización.** Es llamada por el núcleo del sistema operativo en la inicialización del sistema. La rutina se encarga de inicializar el dispositivo incluyendo la información correspondiente en los registros de estado y operación del mismo.
- **Atención de peticiones de entrada/salida.** Esta rutina atiende todas las peticiones de los procesos de usuario para realizar operaciones de entrada/salida.
- **Gestión de interrupciones.** Es la rutina que maneja todas las interrupciones *(Interrupt handler)* del dispositivo. Toma el control cuando el dispositivo periférico origina una interrupción en el procesador.
- **Cancelación de operaciones de entrada/salida.** Es una rutina que da por finalizadas las operaciones de entrada/salida sobre el dispositivo cuando se produzca alguna circunstancia que le obligue a ello.
- **Otras.** Existen otras rutinas menos importantes, como pueden ser: el **time-out** que controla el tiempo de proceso de la operación y el **Power-fail** que actúa en el arranque y al reanudarse el proceso después de un corte de la alimentación del sistema.

8.4.3. Estructuras de datos de un driver

Las rutinas de un driver para dar un correcto servicio a las peticiones de entrada/salida necesitan para cada dispositivo una serie de datos que se encuentran en estructuras de datos

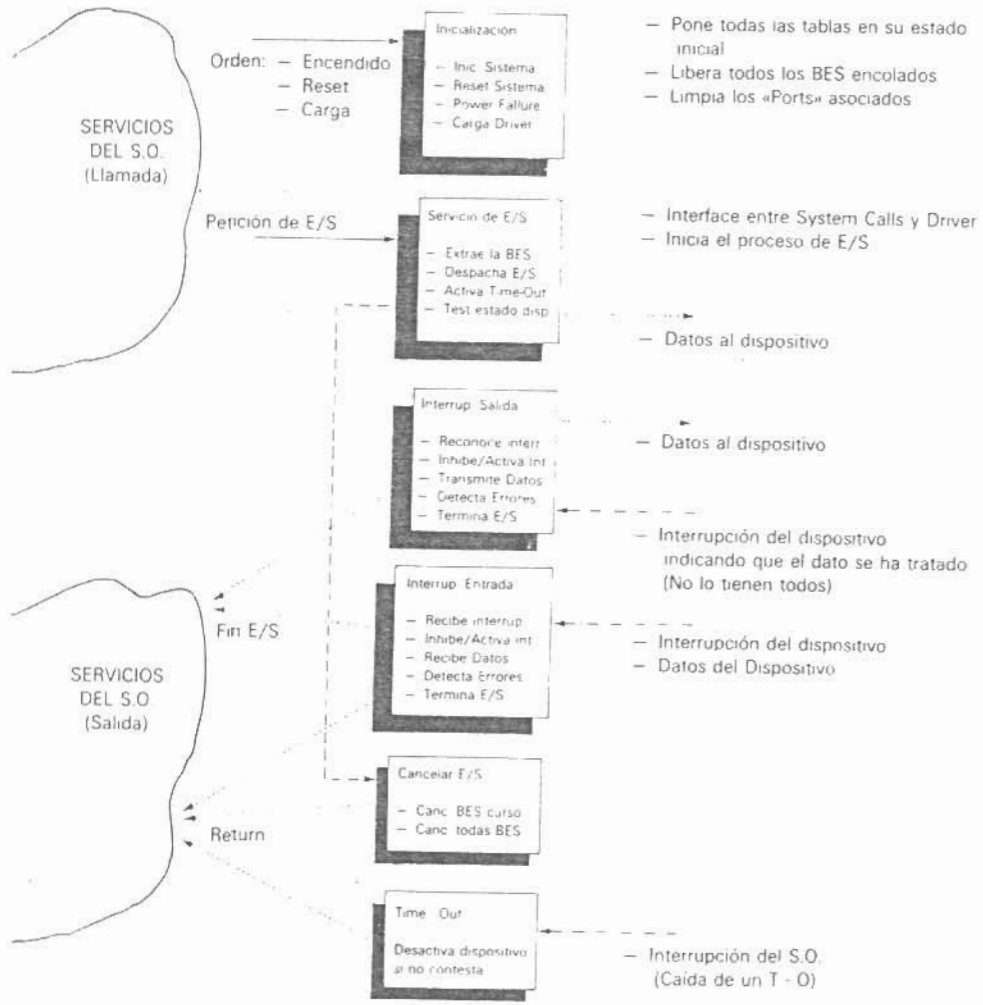


Figura 8.12. Funciones y rutinas de un driver.

en forma de tabla, de manera que su composición depende del sistema operativo, aunque tienen forma y nombres similares a los siguientes:

- **Bloque de control del driver (BCD)**

Es la representación del driver desde el punto de vista del sistema operativo. Contiene aquellos parámetros que son susceptibles de ser variados dinámicamente y aquellos que definen el tipo de dispositivo que puede ser atendido por el driver. Los datos que suele contener son:

- Dirección del siguiente BCD.
- Nombre del driver.
- Dirección de comienzo de los bloques de control de unidades (BCU) que controle el driver.
- Número de unidades a servir.
- Dirección de comienzo de la rutina de inicialización del driver.
- Estado del driver (On/off line...).
- Dirección de comienzo de la cola de bloques de entrada/salida (BES) en modo serie.
- Dirección del BES que está siendo servido.
- Variables particulares del driver.

▪ Bloque de control de la unidad (BCU)

Cada dispositivo físico se relaciona desde el punto de vista del sistema operativo como una unidad dentro del tipo al que le corresponda y es definido e identificado por el sistema operativo por medio de su BCU. En general, contiene los siguientes datos:

- Dirección del siguiente BCU del driver.
- Número de unidad.
- Estado de la unidad.
- Número del vector de interrupción asociado.
- Dirección de la rutina de gestión de la interrupción.
- Dirección del puerto (port) de entrada/salida.
- Dirección del BCD al que pertenece.
- Dirección del PCB del proceso que tiene reservada esta unidad.
- Dirección del comienzo de la cola de bloques de entrada/salida (BES) en modo paralelo.
- Dirección del BES que está siendo servido.
- Características de la unidad.
- Variables particulares del driver.

▪ Paquete de petición de entrada/salida (PES)

Cuando un proceso de usuario intenta hacer una operación de entrada/salida, el sistema operativo crea un paquete asociado a dicho proceso y a dicha petición para ser tratado por el driver. Este paquete se coloca en una cola prioritaria para ser atendido por el driver al que va dirigido. Los datos que normalmente contiene son:

- Dirección del siguiente PES en la cola.
- Prioridad de la petición de entrada/salida.
- Proceso que ha lanzado la petición.
- Dirección donde devolver el resultado de la petición.
- Función a realizar (entrada o salida).
- Identificador del dispositivo.
- Identificador de la unidad.
- Dirección de la lista de parámetros de entrada de la llamada al sistema operativo.

La Figura 8.13 muestra estas estructuras de datos.

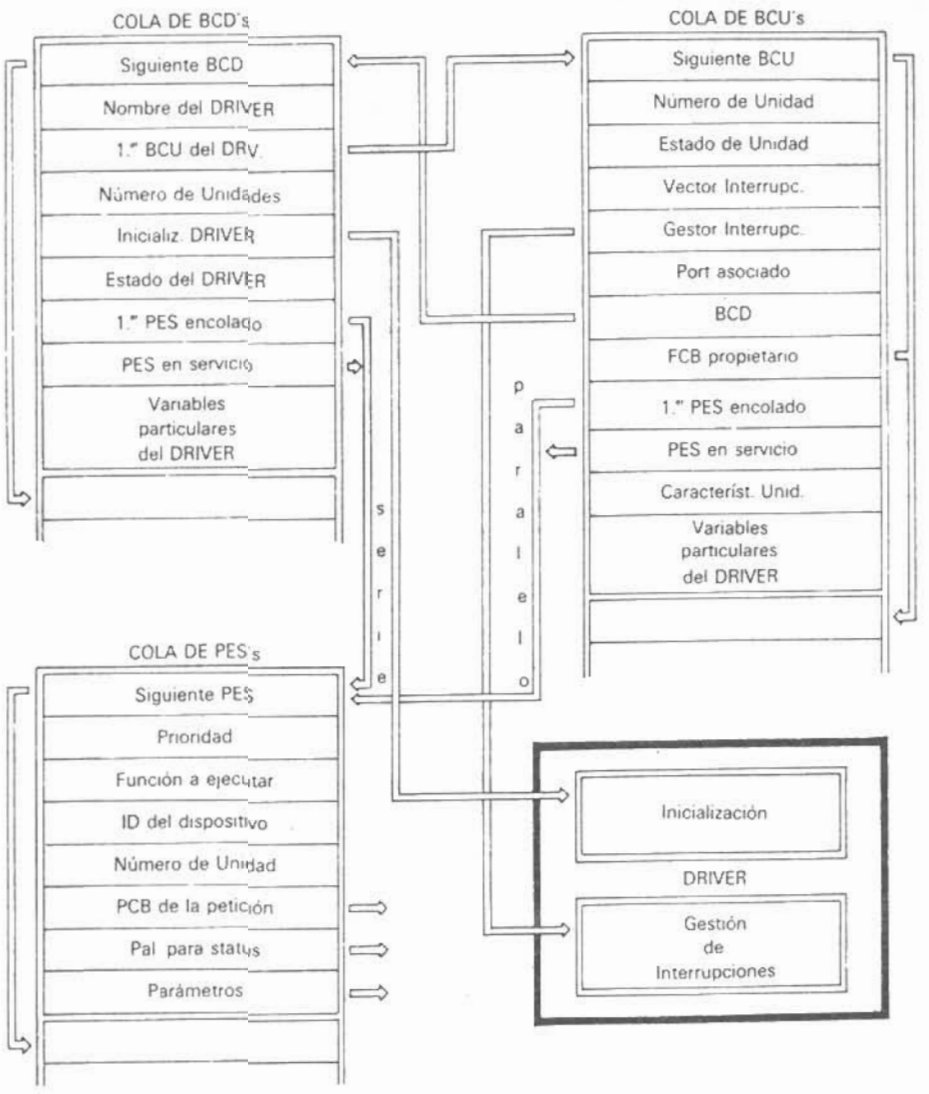


Figura 8.13. Tablas del driver.

8.5. INTERRUPCIONES VECTORIZADAS

Los sistemas operativos realizan diversidad de operaciones y están preparados para aceptar interrupciones que provienen de los dispositivos periféricos. Para poder reconocer qué dispositivo ha sido el causante de una interrupción y poder darle el tratamiento adecuado, el

sistema operativo destina parte de su memoria (la más baja) para almacenar las direcciones de los ya mencionados manejadores de interrupciones asociados a cada dispositivo. Cada palabra almacenada que contiene la dirección de un manejador de interrupción se conoce con el nombre de **vector de interrupción.**

Por tanto, el vector de interrupción es un número que nos indica la palabra que contiene la dirección de una rutina que debe tratar una interrupción.

Un sistema operativo admite un máximo de vectores de interrupciones que es fijado en el momento de la generación del mismo.

8.6. DIRECCIONES DE ENTRADA/SALIDA DEL DISPOSITIVO

Los dispositivos periféricos o sus controladores están provistos de unos registros hardware que pueden ser leídos o escritos por el sistema operativo para facilitar las operaciones que se efectúan sobre ellos. Estos registros suelen ser (Figura 8.14):

- **Estado.** Este registro indica el estado en que se encuentra el dispositivo (Interrupciones habilitadas o no, error de operación, configuración, etc.). En este caso el sistema operativo sólo puede leer de él.
- **Operación.** Es el registro donde el sistema operativo, a través del driver, puede escribir indicando al dispositivo qué operación es la que se solicita (entrada, salida, informar del estado, etc.).
- **Datos.** Es el registro donde se depositan los datos relativos a la operación.

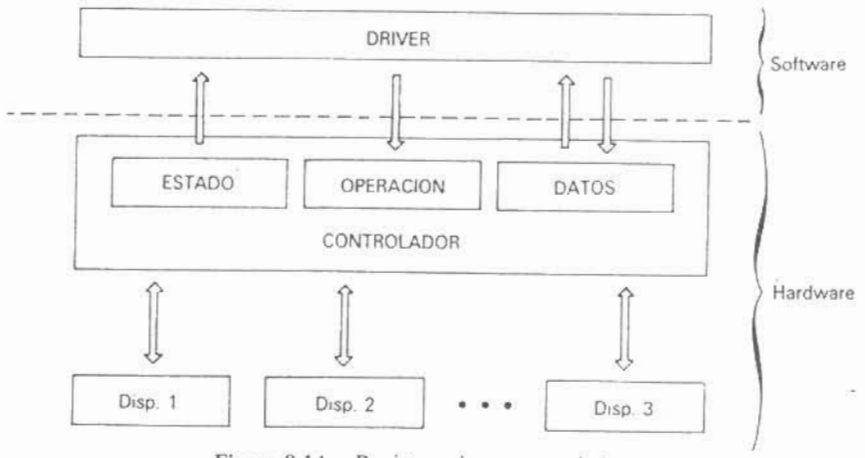


Figura 8.14. Registros de un controlador.

La dirección de la zona de memoria que contiene los valores de estos registros se conoce con el nombre de **dirección de entrada/salida** *(I/O address)* del dispositivo. El sistema operativo asigna una zona de memoria que contiene los valores de los registros de todos los dispositivos que se conoce con el nombre de **página de entrada/salida** *(I/O page).* En general, esta zona suele ser la más alta de la memoria (Figura 8.14).

CUESTIONES

1. Explicar por qué es necesario incluir rutinas de control y adaptación para la realización de operaciones de entrada/salida.
2. ¿En qué tres grandes grupos se pueden reunir los dispositivos hardware que se han desarrollado a lo largo de la historia?
3. Comentar brevemente características, tipos y forma de actuar de los dispositivos de almacenamiento en disco.
4. Comentar brevemente características y aplicaciones de las cintas magnéticas.
5. ¿Qué es un terminal y en qué dos categorías se pueden dividir?
6. ¿Para qué se utiliza una línea de comunicación y de qué tipos pueden ser?
7. ¿De qué tres formas pueden ser conectados los dispositivos periféricos al procesador?
8. ¿Qué es un controlador?
9. ¿Cuál es el propósito general de un canal?
10. Definir el concepto de software de control de las operaciones de entrada/salida o driver.
11. Comentar brevemente el proceso de gestión de una operación de entrada/salida y realizar un esquema del mismo
12. ¿Cuáles son las funciones de un driver?
13. Realizar un esquema simplificado de las estructuras de datos de un driver.
14. ¿En qué consiste un vector de interrupción?
15. Comentar el significado que tienen los nombres «direcciones de entrada/salida» y «página de entrada/salida».

CAPITULO 9

Gestión del almacenamiento secundario

9.1. INTRODUCCION

Como ya hemos mencionado, el desarrollo de los sistemas operativos impone la existencia de capacidad para almacenar información, código y datos fuera de la memoria principal (recuérdese, por ejemplo, la técnica de intercambio de almacenamiento swapping). También los usuarios necesitan disponer de un medio de almacenamiento donde poder mantener sus programas de uso más o menos frecuente en formato ejecutable, programas fuente en fase de desarrollo, textos editados, archivos de datos, etc.

El hardware suministra esta capacidad de almacenamiento secundario mediante distintos tipos de soportes de la información (discos y cintas magnéticas), así como sus correspondientes unidades o periféricos (unidad de disco y unidad de cinta). Dispositivos de características físicas y funcionales diferentes pero que permiten todos ellos guardar un gran volumen de información.

En este capítulo analizaremos la utilización de estos dispositivos. Entre ellos los discos que actualmente son los más usados y veremos las diferentes maneras de gestionar sus accesos y las operaciones que sobre ellos se deban hacer. A continuación veremos las distintas alternativas para gestionar el espacio existente en el disco.

Como el sistema operativo es el encargado de manejar el soporte físico a través de su propia unidad, estudiaremos también los diferentes modos de acceso que pueden ser utilizados por el usuario para realizar operaciones con la información almacenada.

El sistema operativo tiene la misión de hacer ver al usuario el tratamiento de la información almacenada en soportes externos desde un punto de vista lógico, independizándolo de la realidad física. De esta manera, los cambios tecnológicos introducidos en el soporte físico y en su unidad, no forzarán a cambiar la estructura lógica de los archivos y sus modos de acceso y operación.

Desde otro punto de vista, un sistema informático puede admitir a un gran número de usuarios y consiguientemente necesitará mantener en memoria externa un gran número de programas y archivos. El sistema operativo debe manejar estos archivos asegurando la independencia y el acceso por aquellos usuarios que lo tengan permitido, y esto lo podrá hacer organizándolos según diferentes criterios. La parte que se encarga de esta función es la denominada **subsistema de archivos.**

Por último se analizará el problema de la protección de los archivos ante intentos, voluntarios o no, de acceder a su información sin derecho a ello.

Empezaremos recordando aspectos relativos a la información.

9.2. ESTRUCTURA DE LA INFORMACION

La información que maneja una computadora, como ya es sabido, se compone de combinaciones de ceros y unos que suelen denominarse **dígitos binarios** o **bits** que se corresponden físicamente con presencia o ausencia de pequeñas señales eléctricas sobre los diferentes elementos electrónicos que configuran el hardware de la computadora.

En el almacenamiento de informaciones se utiliza un **conjunto de caracteres** (letras, cifras, operadores y caracteres especiales) que forman cada unidad de información a través de lo que se denomina **cadena de caracteres,** y cada uno de ellos (carácter) se configura por medio de un conjunto de bits, generalmente 8 que permiten juegos de caracteres de 256 elementos (2^8). Los códigos de 8 bits más utilizados en la actualidad son el **ASCII** *(American Standard Code for Information Interchange)* y el **EBCDIC** *(Extended Binary Coded Decimal Interchange Code).*

Un conjunto de caracteres relacionados entre sí se denomina **campo,** y podrá ser numérico, alfabético o alfanumérico. Distintos campos reunidos y referidos a una misma entidad configuran un **registro.** Un conjunto de registros relacionados entre sí configuran lo que se denomina **archivo.**

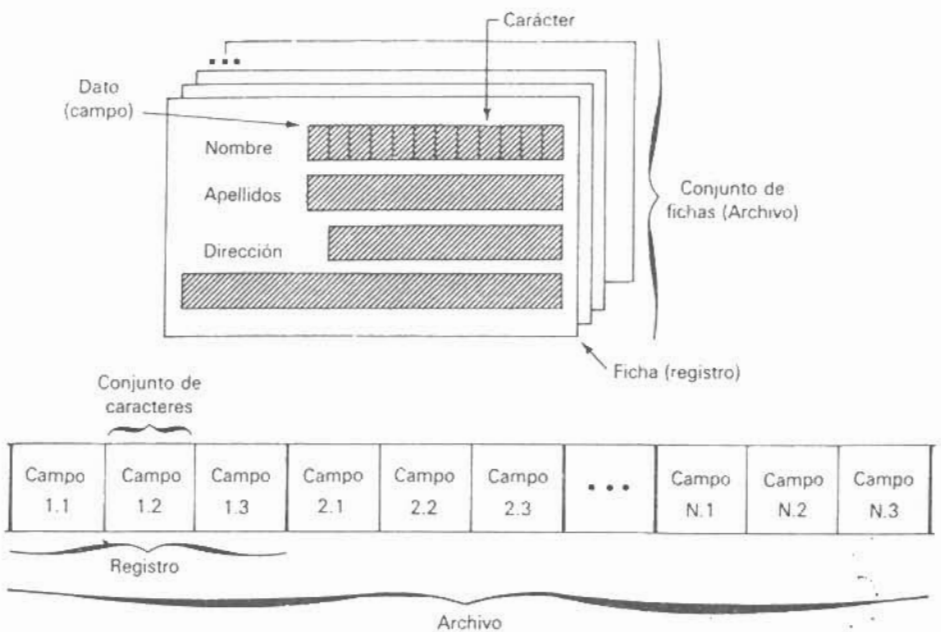


Figura 9.1. Estructura lógica de un archivo.

En general, todo registro tiene uno o más de sus campos que los distinguen y diferencian de los demás registros. Este o estos campos se denominan **clave** o **claves** (primaria, secundaria, etc.). En la Figura 9.1 puede verse la estructura lógica de un archivo.

El último nivel en la jerarquía lo constituyen las **bases de datos,** que son grupos de archivos relacionados entre sí y que se gestionan conjuntamente (Figura 9.2).

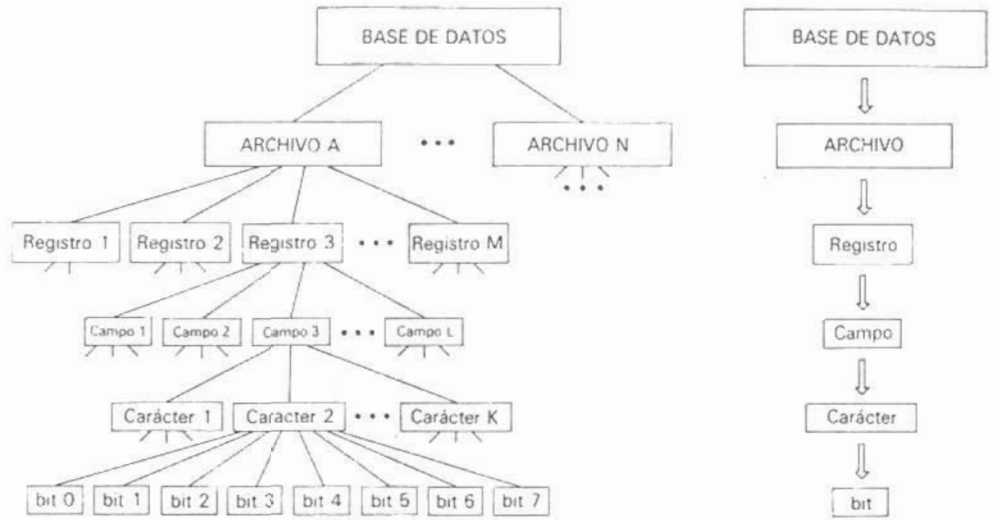


Figura 9.2. Jerarquía de la información.

9.3. SOPORTE FISICO DE LA INFORMACION

Los soportes físicos utilizados para almacenar información que se utilizan en la actualidad son las cintas magnéticas y los discos (magnéticos y ópticos). De todos ellos, los más utilizados son los discos magnéticos y en ellos centraremos gran parte de nuestro estudio.

Los primeros sistemas operativos desarrollaron su gestión de archivos utilizando cinta magnética, que en su día fue un avance notable frente a los archivos en fichas y cinta perforada. No obstante, la cinta magnética, aunque presenta ventajas en cuanto a coste y capacidad de almacenamiento, tiene serios inconvenientes ya que sólo permiten la utilización de archivos secuenciales con el problema de tiempo de acceso que ello representa.

Con la aparición de los discos magnéticos, la cinta pasó a un segundo plano debido a las buenas características de los discos fundamentalmente en cuanto al tiempo de acceso íntimamente ligado a la gestión de archivos de información con acceso rápido. En la actualidad las cintas magnéticas se utilizan para la realización de copias de seguridad y como respaldo de los discos en determinadas operaciones.

Los sistemas operativos actuales basan su capacidad de almacenamiento secundario en los discos magnéticos.

9.3.1. Registros físicos y lógicos. Bloqueo de registros

El hardware de los dispositivos de almacenamiento determina el tamaño de la unidad básica de información que se transfiere en cada operación de lectura o escritura. Esta cantidad de información recibe el nombre de **registro físico** o **bloque** y en los discos se corresponde con lo que se denomina **sector.**

El usuario, por otro lado, maneja la información en unidades lógicas de tamaño variable según sea la aplicación y el archivo de que se trate. Estas unidades reciben el nombre de **registros lógicos** o simplemente **registros.**

Para mejorar la velocidad de acceso a la información y optimizar el uso del dispositivo, el sistema puede bloquear los registros **(registros bloqueados)** agrupando varios en cada bloque físico. Así se consigue un mejor aprovechamiento de la capacidad del bloque (menor fragmentación interna) y se reduce el tiempo de acceso al registro lógico al disponer de varios de ellos en cada operación física (Figura 9.3).

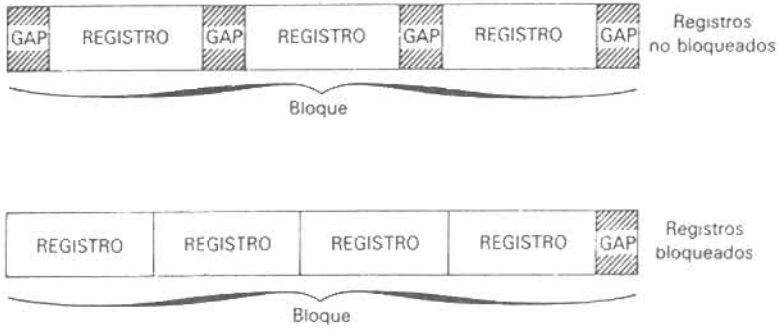


Figura 9.3. Registros bloqueados y no bloqueados.

9.4. PLANIFICACION DE LOS ACCESOS A DISCO

En los sistemas informáticos actuales, como ya hemos dicho, es importante que la gestión de la información sobre disco sea eficiente puesto que representa un altísimo porcentaje de operaciones de almacenamiento y recuperación de datos. Por ello, es necesario agilizar al máximo los accesos a la información contenida en los mismos.

La velocidad a la que se realicen los accesos a disco condicionará aspectos importantes del rendimiento general del sistema (paginación, swapping, carga de programas, etc.) y del rendimiento propio de las aplicaciones de usuario (acceso a los datos en archivos).

Recordando la estructura lógica de un disco (o paquete de discos), la información se graba en pistas concéntricas o si hay más de un disco en cilindros cuyas pistas a su vez se dividen en sectores que en este caso coinciden con la denominación que se ha dado de bloque.

Ante una petición de acceso a un bloque determinado, el hardware del disco debe realizar, fundamentalmente, tres operaciones (Figura 9.4):

- Mover el brazo buscando la pista o cilindro correspondiente **(búsqueda).**
- Esperar a que el bloque se sitúe frente a la cabeza de lectura/escritura **(latencia).**
- Transmitir el bloque deseado **(transmisión).**

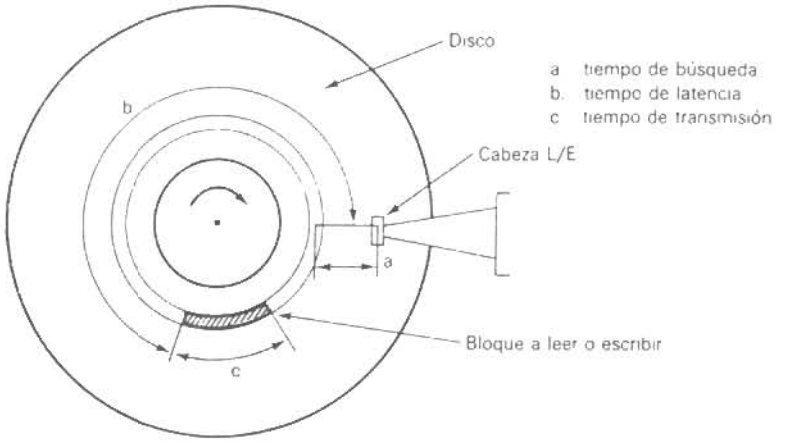


Figura 9.4. Tiempos de acceso a disco.

El tiempo de búsqueda y latencia dependen de la posición relativa del bloque y de las cabezas de lectura/escritura en el momento de la petición mientras que el tiempo de transmisión será igual en todos los accesos puesto que todos los bloques son del mismo tamaño.

9.4.1. Algoritmos de planificación

En los sistemas multiprogramados son varios los procesos en ejecución en un momento dado, y pueden producirse peticiones simultáneas de acceso a un mismo disco. Si mientras se realiza un acceso llegan más peticiones, el sistema deberá mantenerlas en una lista de espera. Finalizado el acceso que se estaba atendiendo, el disco quedará libre y el sistema podrá decidir el orden en el que atenderá las peticiones pendientes, buscando minimizar el desplazamiento del brazo del disco y por tanto el tiempo de servicio de la petición.

Vamos a analizar algunos de los distintos algoritmos que pueden utilizarse para planificar dichas peticiones.

- **Primero en llegar, primero en acceder (FCFS)**

El criterio más simple es aquel que sirve las peticiones según su orden de llegada. Su programación es sencilla y no produce una sobrecarga significativa, pero su eficacia es relativa.

Consideremos un disco en el que coinciden en un período pequeño de tiempo las siguientes peticiones de acceso relacionadas por el número de pista donde se encuentra el bloque a leer o escribir:

98 183 37 122 14 124 65

Suponiendo que el brazo del disco parte del cilindro 53 cuando comienza a atender esas peticiones, el gráfico de la Figura 9.5 muestra los desplazamientos que realiza dicho brazo cuando se utilice el algoritmo FCFS *(First Come, First Served)*.

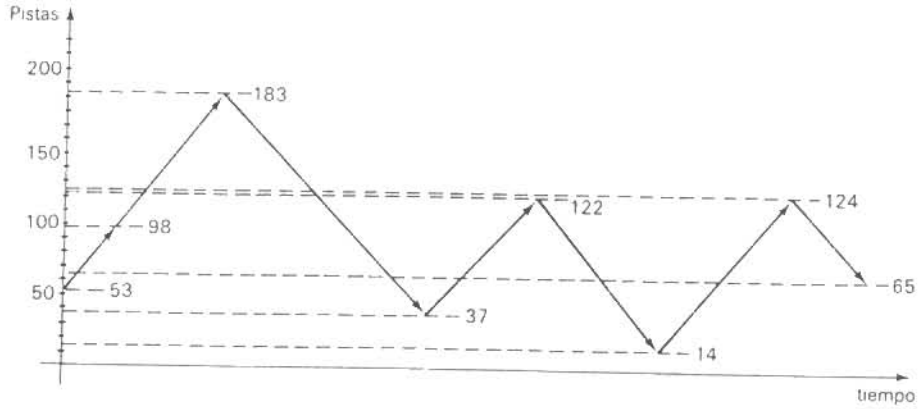


Figura 9.5. Planificación del disco FCFS.

Para atender todas las peticiones, el brazo se ha desplazado un total de 638 cilindros. Sin embargo, se podría reducir notablemente ese recorrido, si las peticiones 37 y 14 se atendieran juntas, antes o después de servir las peticiones 122 y 124.

Este algoritmo puede ser útil para planificar discos con poca carga (pocos accesos) y así lo utilizan algunos sistemas operativos.

■ Primero el de menor tiempo de búsqueda (SSTF)

Este algoritmo atenderá primero la petición más cercana a la última servida, o lo que es lo mismo, aquella que requiera un desplazamiento menor del brazo.

Si aplicamos este criterio a la lista de peticiones anterior, obtendremos el gráfico de la Figura 9.6.

El desplazamiento total es en este caso de 232 cilindros, notablemente menor que el obtenido con el criterio anterior.

Evidentemente este es un algoritmo más eficaz, pero su forma de atender las peticiones puede postergar indefinidamente algunas de ellas. Supongamos que existen dos peticiones pendientes, cilindros 25 y 110. Si mientras se atiende la 25 llegan más peticiones cercanas a ella (o menos alejadas que la 110), este algoritmo atenderá las nuevas aplazando el servicio de la 110.

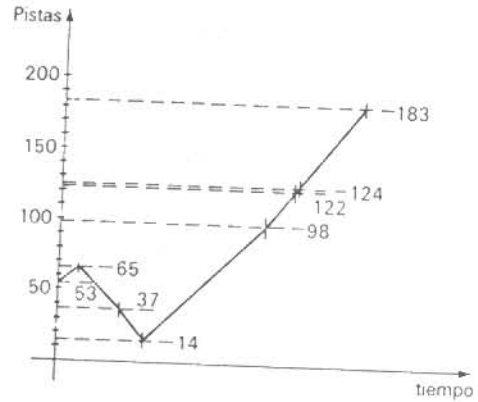


Figura 9.6. Planificación del disco SSTF.

Por la misma razón, la velocidad de servicio de una petición variará según que las peticiones posteriores provoquen su postergación o no. Esta falta de predicción de los tiempos de acceso al disco, hace que el algoritmo SSTF *(Shortest Seek Time First)* sea poco adecuada para los sistemas interactivos.

Estas variaciones tienen menor importancia cuando se trata de procesar trabajos por lotes (batch) y, por ello, este algoritmo es más útil para sistemas batch.

■ Exploración (Scan)

Ahora se establece un recorrido predeterminado para el brazo del disco que va del primer al último cilindro y vuelta. En ese recorrido irá atendiendo las peticiones más cercanas según el sentido del desplazamiento.

Como en los casos anteriores, la Figura 9.7 muestra la aplicación de este algoritmo a la lista de peticiones del ejemplo anterior.

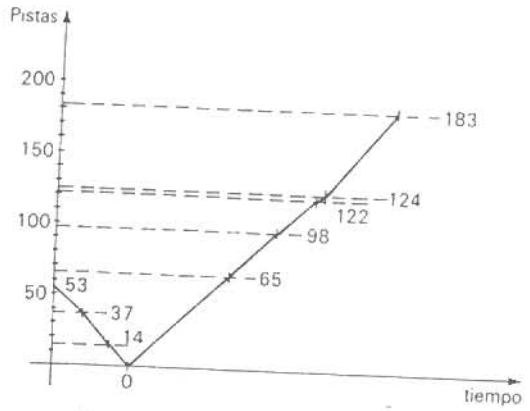


Figura 9.7. Algoritmo de exploración.

El desplazamiento total es de 238 cilindros, similar al logrado con el SSTF pero sin sus inconvenientes.

Este ha sido el algoritmo base a partir del cual se han desarrollado los algoritmos de planificación de discos más utilizados en los sistemas actuales.

- **Exploración circular** (C-Scan)

La exploración circular es una variación del algoritmo anterior, que pretende conseguir unos tiempos de espera más equilibrados, con independencia de que las peticiones correspondan a cilindros de los extremos o del interior del disco.

Con ese fin, el movimiento del brazo sigue siendo el mismo que en el caso anterior, pero sólo atiende peticiones en uno de los sentidos. Al llegar al último cilindro regresará al principio rápidamente sin atender petición alguna.

La Figura 9.8 muestra la aplicación de este algoritmo al caso de los ejemplos anteriores.

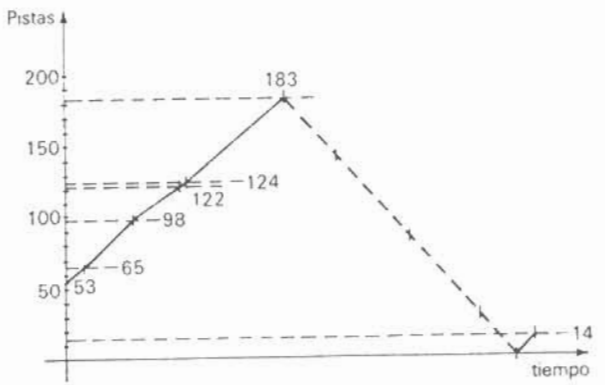


Figura 9.8. Algoritmo de exploración circular.

En la práctica, tanto el algoritmo anterior como éste, el brazo del disco no llegará a los cilindros extremos del disco, sino que cambiará de sentido al servir la última petición.

Estos dos últimos son los algoritmos más adecuados para planificar discos que deban soportar mucha carga.

Como siempre, la decisión de qué algoritmo utilizar en un sistema determinado, dependerá de los objetivos a cubrir por dicho sistema y la carga en disco que deba manejar.

9.5. SOPORTE LOGICO. SUBSISTEMA DE ARCHIVOS

Desde el punto de vista de los usuarios, los archivos son grupos de informaciones relacionadas entre sí sobre los que puede realizar diversas operaciones (lectura, escritura, actualización, borrado, etc.). El sistema será el responsable de realizar dichas operaciones, adecuándolas a las características de los dispositivos físicos utilizados.

La parte del sistema operativo encargada de dichas funciones suele recibir el nombre de **subsistema de archivos,** y su misión es realizar las siguientes tareas:

- **Gestión del almacenamiento.** Debe decidir cómo asignar el espacio de almacenamiento disponible.
- **Métodos de acceso.** Definen cómo puede acceder el usuario a la información almacenada.
- **Gestión de archivos.** Debe controlar los archivos existentes, sus relaciones, como compartirlos, como crearlos, etc.
- **Protección e integridad de los archivos.** Deberá garantizar la información contenida, su integridad y privacidad.

Como cabía esperar, es posible utilizar diferentes criterios y técnicas para realizar estas tareas, y cada sistema será diseñado según aquellos que respondan mejor a sus necesidades y objetivos.

9.6. GESTION DEL ALMACENAMIENTO. ASIGNACION DE ESPACIO

Cuando un usuario desea crear un archivo, el subsistema debe asignarle el espacio necesario para que pueda almacenar su información y ese espacio lo obtendrá del total disponible en ese momento.

Por otra parte, llegará un momento en el que el usuario ya no necesite su archivo y lo borrará. El subsistema añadirá el espacio liberado al total disponible para poder utilizarlo en posteriores peticiones. Esta gestión del espacio de almacenamiento la realiza el subsistema de archivos persiguiendo dos objetivos:

- Utilizar eficazmente el espacio de almacenamiento.
- Posibilitar un acceso rápido a la información almacenada.

Con las cintas magnéticas el problema es menor ya que sus características físicas determinan la forma de asignar el espacio (un registro tras otro y un archivo tras otro).

9.6.1. Control del espacio disponible

Para poder gestionar el espacio de almacenamiento, el subsistema necesita controlar en todo momento el espacio disponible (continuamente se están creando y borrando archivos). Normalmente realizará este control para cada dispositivo por separado, aunque en sistemas pequeños se puede pensar en un control del espacio total.

Centrándonos en la situación más común, el subsistema mantendrá en cada dispositivo una lista de espacio libre a la que irá añadiendo el espacio liberado por los archivos que se borran, y de la que restará el asignado a nuevas peticiones.

Una primera forma de llevar a la práctica dicho control consiste en mantener un **mapa de bits** por dispositivo, en el que cada uno de los bloques del disco está representado por un bit cuyo valor indica su estado (0 libre, 1 asignado). La Figura 9.9 muestra un ejemplo.

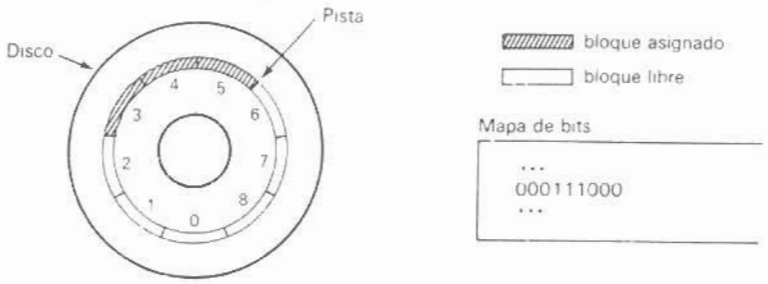


Figura 9.9. Mapa de bits.

Otra solución es mantener en el subsistema un apuntador al primer bloque libre del disco, éste apuntará al siguiente y así sucesivamente. En la Figura 9.10 puede verse un ejemplo de encadenamiento de bloques libres.

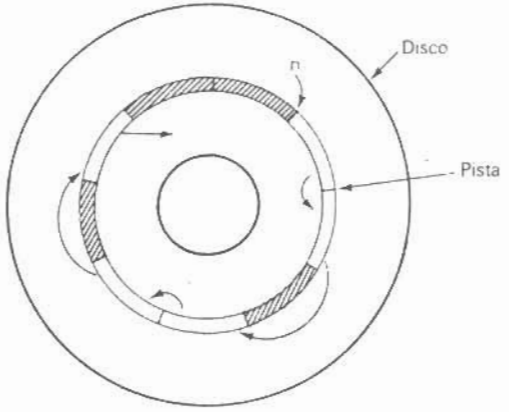
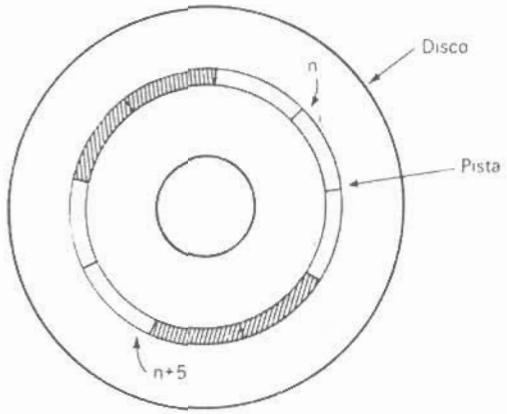


Figura 9.10. Encadenamiento de bloques libres.

Esta solución, sin embargo, no es eficiente ya que el mantenimiento de los apuntadores consume mucho espacio y provoca un gran número de operaciones de entrada/salida.

La solución más extendida parte de la consideración de que el espacio se asigna y se libera, generalmente, en grupos de bloques contiguos y, por ello, se mantiene en cada disco una **tabla de grupos de bloques libres.** Dicha tabla contiene la dirección del primer bloque de cada grupo y el número de ellos que lo forman (Figura 9.11).



Tabla

Inicio	N.° de bloques
• • •	• • •
n	3
n+5	2
• • •	• • •

Figura 9.11. Agrupamiento de bloques libres.

En este caso, ante una petición de espacio, el subsistema buscará en dicha tabla un grupo contiguo de bloques suficiente, lo asignará y actualizará la tabla de control. Si el espacio no se asigna muy fragmentadamente, esta alternativa es eficaz pues permite conocer el espacio disponible con un solo acceso a disco (frente a los continuos acceso requeridos por el método encadenado) y la tabla ocupa poco espacio.

Como siempre, el diseñador decidirá una u otra alternativa según las características del sistema y de los dispositivos. Si por ejemplo éstos son de mucha capacidad, el mapa de bits adquiere tamaños inmanejables y se deberá recurrir a la tabla de grupos.

9.6.2. Directorio de dispositivo

El sistema mantiene en cada disco un espacio reservado donde guarda información relativa a los archivos existentes en el mismo y al espacio libre, este espacio recibe el nombre de **directorio** que creará el sistema en cada disco al formatearlo o inicializarlo (proceso de preparación del disco para poder ser utilizado por el sistema).

La estructura de este directorio la define cada sistema operativo y, por ello, el transporte de dispositivos de almacenamiento entre computadoras queda condicionado por la compatibilidad de los sistemas utilizados.

Por lo general, un directorio o tabla de contenidos consta de una entrada por cada archivo existente en el disco y otra para la información sobre el espacio disponible.

En las entradas de archivo el subsistema registra la información necesaria para su gestión:

- Nombre de archivo.
- Tipo de archivo.
- Localización en el disco.
- Tamaño.
- Protección. Lista de accesos permitidos.
- Contabilidad. Fecha de creación, propietario, etc.
- Contadores (de uso, de posición, etc.).

La forma en que estén organizadas estas entradas en el directorio condicionará la velocidad de acceso a la información. El acceso a un archivo requiere como primer paso su localización y ésta será más rápida si las entradas del directorio están ordenadas alfabéticamente.

Para agilizar la búsqueda del archivo, algunos sistemas exigen que el directorio esté siempre en la misma dirección del disco. Otros permiten definir su situación mediante programas de utilidad especiales, quedando su dirección reflejada en un bloque prefijado. En ambos casos, el subsistema podrá encontrar rápidamente el directorio del disco.

Por otra parte, para evitar los continuos accesos al directorio del disco como paso previo a cualquier operación, algunos sistemas mantienen en memoria una **lista de archivos abiertos** donde quedan reflejados los archivos que se están utilizando y su dirección en el disco que los contiene. Cuando un usuario vaya a trabajar con un archivo, deberá abrirlo previamente y en esta operación el sistema añadirá su nombre y dirección a la citada lista. A partir de ese momento, cada operación sobre dicho archivo irá directamente al mismo sin necesidad de pasar antes por el directorio. Se sustituyen los accesos al directorio por accesos a memoria mucho más rápidos.

Al acabar de utilizar un archivo, el usuario lo cerrará y el subsistema lo borrará de la lista.

9.6.3. Asignación del espacio de almacenamiento

El sistema operativo es el responsable de plasmar en disco los archivos que crean los usuarios, suministrándoles el espacio necesario.

Ante una petición para crear un archivo de N bits, el subsistema deberá comprobar si existe suficiente espacio disponible y, a continuación podrá optar por dos estrategias:

- Asignar N bits contiguos de espacio en disco.
- Almacenar el archivo en trozos no contiguos.

Veamos las características de cada una de estas opciones.

■ Asignación contigua

Según este criterio, se coloca cada archivo en un grupo de bloques contiguos del disco. Cada entrada del directorio del disco contendrá, además del nombre del archivo y otros datos, la dirección del bloque inicial del archivo y el número de bloques que ocupa. Por ejemplo, un archivo que ocupa 5 bloques empezando en el número 3, se encontrará situado en los bloques 3, 4, 5, 6 y 7 (Figura 9.12).

De esta forma, el acceso a un bloque n+1 partiendo del bloque n no exigirá, en la mayoría de los casos, movimiento del brazo del disco. Si el acceso a un archivo se hace de forma secuencial, el subsistema de archivos recordará en cada momento cuál será el siguiente bloque a leer o escribir y el tiempo de búsqueda será prácticamente nulo. La colocación contigua también permite el acceso directo de forma eficaz ya que cualquier bloque puede ser accedido rápidamente partiendo de la dirección inicial del archivo.

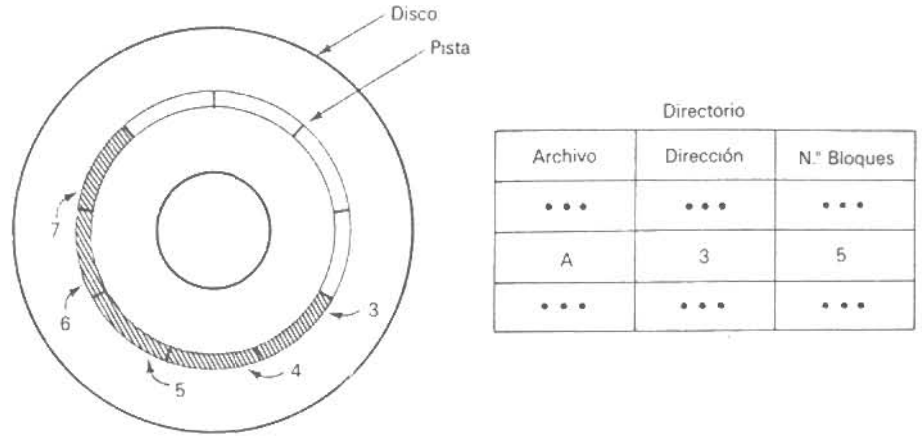


Figura 9.12. Asignación contigua.

El inconveniente mayor en este caso es el de la fragmentación externa que se produce, ya que en un momento dado puede producir un número de huecos grande con suficiente espacio total en disco para un nuevo archivo, pero sin que exista ninguno capaz de contenerlo.

Otro inconveniente de esta técnica es el de tener que conocer por anticipado el tamaño del archivo, lo que suele llevar a los usuarios al normal sobredimensionado de los mismos con la consiguiente pérdida de espacio en disco. Si a pesar de ello el archivo se quedase pequeño, sería necesario definir uno mayor y trasvasar la información con la sobrecarga que esto lleva consigo.

Algunos sistemas pequeños que utilizan discos flexibles los compactan periódicamente con el fin de agrupar los huecos libres en uno de mayor tamaño. Esta solución no es válida para aquellos sistemas que utilicen discos de gran capacidad pues agrupar sus archivos exige mover grandes cantidades de información y esto puede afectar al proceso de otros trabajos del sistema.

Otros sistemas de computadoras de gran tamaño utilizan una técnica de asignación contigua que permite ampliar los archivos después de su creación. Al definir un archivo se deben definir dos valores, el tamaño del mismo y la cantidad de espacio a añadirle en cada crecimiento (llamada **extensión**). Cuando el archivo necesite ser ampliado, el subsistema le irá asignando nuevas extensiones que no tienen por qué ser contiguas a aquél (Figura 9.13).

El último bloque del espacio inicial apuntará a la primera extensión y el último de ésta apuntará a la siguiente. El subsistema limitará el número máximo de extensiones utilizables.

■ Asignación enlazada

Este método es de asignación no contigua, en el que cada archivo es una lista enlazada de bloques del disco que pueden estar en cualquier dirección del mismo.

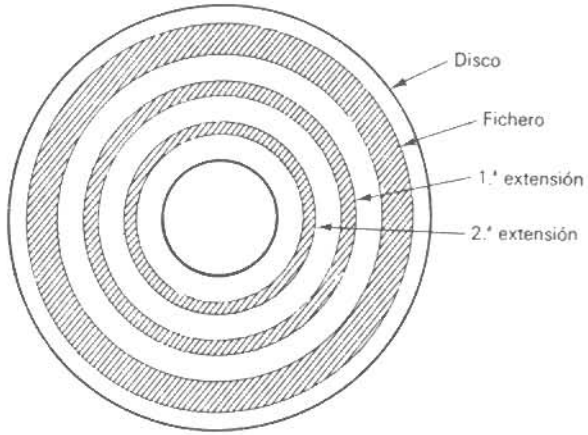
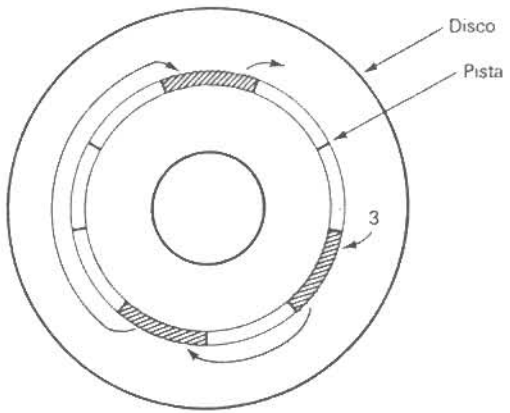


Figura 9.13. Asignación contigua con extensiones.

En el directorio del disco cada entrada de archivo contendrá, además del nombre del mismo y otros datos, un apuntador al primer bloque de la cadena (en algunos sistema también se apunta al último bloque). A partir de este primer bloque, cada uno de los siguientes contiene un apuntador al que le sigue (Figura 9.14).



Directorio

ARCHIVO	Bloque inicio
• • •	• • •
A	3
• • •	• • •

Figura 9.14. Asignación enlazada.

Ahora, para crear un archivo, el subsistema le asignará un bloque físico libre y añadirá una entrada en el directorio. A medida que van aumentando las necesidades de espacio del archivo, el subsistema le irá enlazando bloques libres.

Esta técnica de asignación no provoca fragmentación externa, ya que todos los bloques son del mismo tamaño y por tanto utilizables para las nuevas necesidades de cualquier archivo. Estos podrán crecer mientras existan bloques libres.

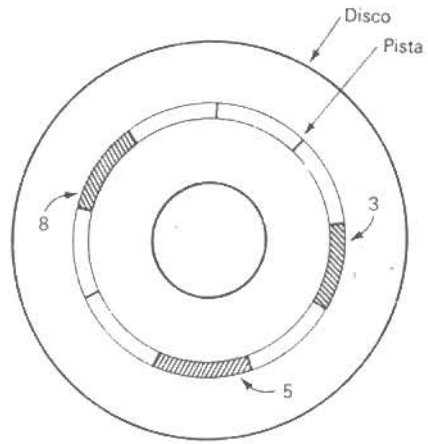
El usuario no necesita declarar el tamaño del archivo y no puede por ello acaparar espacio innecesario en el disco.

El encadenamiento de bloques facilita el tratamiento de archivos secuenciales, pero en el caso de los accesos directos la solución no puede ser más desfavorable, ya que siempre hay que seguir la cadena de apuntadores.

La asignación enlazada es también bastante vulnerable, ya que la pérdida, por cualquier razón (error de hardware o software) de un apuntador de bloque, significa la pérdida del archivo. Con el fin de paliar este peligro se utiliza a veces el doble encadenamiento, es decir, cada bloque apunta al que le sigue y al que le precede. En este caso, aumenta la sobrecarga del método y la ocupación de espacio ocasionado por los apuntadores empieza a ser considerable.

■ Asignación indexada

El origen de los principales inconvenientes de la asignación enlazada está en la distribución desordenada de los apuntadores de los bloques de un archivo. Por ello, se hace inviable el acceso directo. La asignación indexada trata de resolver este problema agrupando todos los apuntadores en un bloque de índices. Cada archivo tendrá su propio bloque de índices y su dirección quedará reflejada en el directorio del disco al crear el archivo (Figura 9.15).



Directorio

ARCHIVO	Bloque Indices
• • •	• • •
A	14
• • •	• • •

Bloque 14

3
5
8

Figura 9.15. Asignación indexada.

No se produce fragmentación externa y se agiliza notablemente el acceso directo a los datos. El acceso a cualquier bloque de un archivo sólo requiere dos acceso a disco, uno al bloque de índices y otro al bloque deseado.

Este método no es apropiado para archivos de pequeño tamaño. Para archivos grandes los apuntadores pueden llenar el bloque de índices, en cuyo caso, el subsistema le encadenará otro bloque de índices o bien podrá recurrir a definir un bloque de índices a bloques de índices.

9.6.4. Rendimiento

Podemos esquematizar las características de los métodos de asignación anteriores, en los siguientes puntos:

- La asignación contigua responde bien a las búsquedas en secuencia y directas. Produce fragmentación externa.
- La asignación enlazada evita la fragmentación externa y logra buenos resultados en los accesos secuenciales. No es válido para accesos directos. Consume espacio para los apuntadores.
- La asignación indexada mejora a la anterior permitiendo el acceso directo. Consume espacio para los bloques de índices.

Algunos sistemas operativos utilizan un solo método de asignación, mientras otros utilizan varios según las características de cada archivo.

9.7. METODOS DE ACCESO

Una vez creado un archivo estando su información almacenada en el soporte correspondiente, supongamos que queremos utilizar su información. ¿Cómo se puede acceder desde el punto de vista lógico a la información de uno de sus bloques?

Independientemente, hasta cierto punto, de la forma en que se haya almacenado la información en el soporte físico, se podrá acceder a ella según un esquema lógico secuencial, directo o bien casi directo por medio de parte de la información contenida en el archivo. El subsistema de archivos del sistema operativo define qué formas de acceso lógico permite y qué métodos de acceso soporta.

Un método de acceso es un conjunto de rutinas y tablas que posibilitan acceder a la información de los archivos según un esquema lógico determinado.

Algunos sistemas ofrecen un solo método de acceso a sus usuarios. Otros, por el contrario, permiten utilizar varios y será el usuario el que decida cuál usar.

Veamos algunos de ellos.

■ Acceso secuencial

Este método permite el acceso a los registros de un archivo en un orden preestablecido, del primero al último y de uno en uno. El efecto es como si el archivo, para el usuario, estuviera organizado por registros consecutivos que sólo permiten llegar a uno cualquiera de ellos pasando previamente por los anteriores (Figura 9.16).

Las rutinas del método de acceso secuencial mantendrán un apuntador al siguiente registro lógico a acceder. Una operación de lectura o escritura, lee o escribe el registro y avanza el apuntador al siguiente.

Este método requiere que los registros lógicos se almacenen siguiendo el orden en el que serán accedidos para su tratamiento, es decir, debe coincidir el orden lógico de los registros y su orden físico.

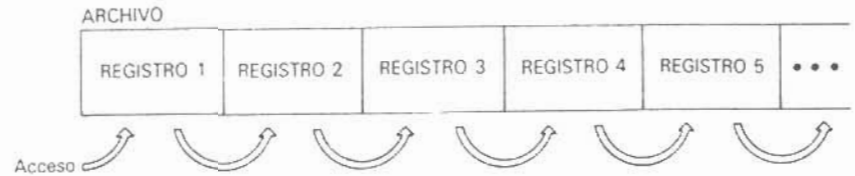


Figura 9.16. Organización lógica secuencial.

Los registros pueden estar ordenados según un campo a través del cual serán accedidos siendo este campo el denominado comúnmente como **campo clave.**

El método de acceso secuencial es sencillo de programar y permite accesos rápidos a la información siempre que se realicen según el orden preestablecido. Además es el método que aprovecha mejor el soporte de información.

La actualización de la información contenida en los registros es complicada, pues es necesario recolocar el contenido total del archivo una vez añadidos los cambios ocurridos. Por ello las operaciones de actualización se realizan de forma masiva (gran número de registros a la vez), copiando el archivo en otro.

Si se quiere acceder a los registros lógicos del archivo según el contenido de otro campo distinto del campo clave, este método obliga a crear un nuevo archivo cuya secuencia guarde el nuevo orden. Ambos archivos tendrán consiguientemente la misma información ordenada de distinta forma. En estos casos, manejar distintas claves provoca el almacenamiento de información redundante.

Es un método muy utilizado por su simplicidad, especialmente en archivos con poca información.

■ Acceso directo

Este método permite el acceso directo a cualquier parte del archivo, es decir, no es necesario pasar por la información anterior para acceder a un determinado registro.

Sólo puede existir este tipo de acceso en aquellos soportes que por su naturaleza lo permitan. Es el caso de los discos, mientras las cintas no pueden tener este tipo de acceso.

El usuario ve el archivo como un conjunto de registros individualizados (numerados con respecto al inicio) a los que puede acceder en cualquier orden. Para ello, ante la petición de un registro determinado el software del método de acceso calcula la dirección del bloque físico que lo contiene y accede a él directamente.

Este cálculo es necesario puesto que el usuario utiliza direcciones relativas de registro (respecto al inicio del archivo), desconociendo el posicionamiento de los registros en el disco. Por ello, si el usuario pide un registro contenido en el bloque 4 de un archivo que comienza en el bloque 75, el subsistema accederá al 78.

Esta sería la forma más sencilla de trabajar por parte del sistema, pero existen otras formas de hacerlo ya que en ocasiones se desconoce el número relativo correspondiente a cada registro y por tanto será necesario relacionar el contenido del mismo con la posición relativa que ocupará dentro del archivo. Una forma de hacerlo es mantener una tabla que contenga

las claves y sus respectivos bloques. Para atender una petición, el método de acceso buscará en dicha tabla la clave solicitada y obtendrá la dirección del bloque correspondiente. Puede también obtenerse un método de transformación de clase que convierta el contenido de uno o varios campos de cada registro en la posición correspondiente. Este último suele producir sinónimos que son registros que a través del método originan la misma posición, con lo que obligan a la existencia de una zona de excedentes que será siempre de acceso secuencial puesto que los registros llegarán en cualquier orden (Figura 9.17).

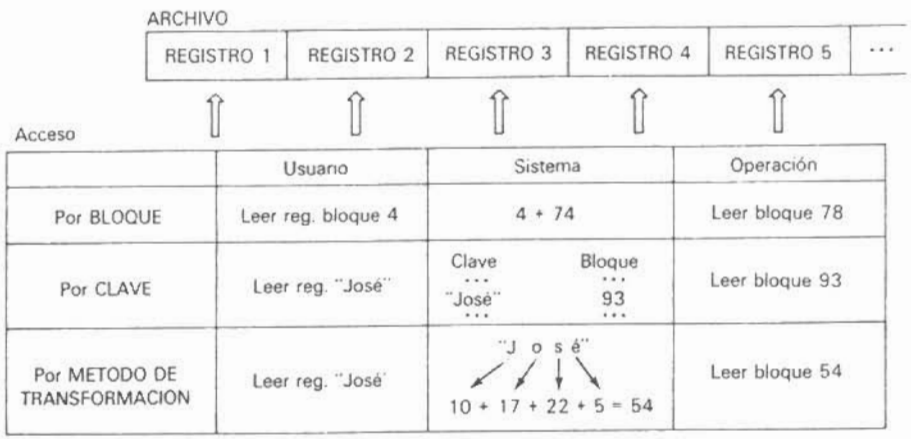


Figura 9.17. Formas del método de acceso directo.

El algoritmo utilizado en la figura sustituye cada letra de la clave por su número de orden en el abecedario y suma sus valores para calcular el bloque asociado.

Este método de acceso es, por sus características, el más adecuado para acceder con rapidez a grandes cantidades de información.

- **Acceso directo indexado**

En este caso se construye un índice o tabla de las relaciones entre las claves y sus bloques físicos para cada archivo.

La localización de un registro se realizará accediendo primero a ese índice y con la dirección del bloque correspondiente a la clave solicitada, se alcanzará el bloque adecuado.

Para archivos grandes se puede utilizar un índice maestro o índice de índices. El maestro apunta al índice secundario que contiene la clave y éste directamente al bloque físico.

La Figura 9.18 muestra un caso concreto en el que las entradas de los índices están organizadas en secuencia de clave (ISAM-Método de acceso con índices secuenciales). Cada bloque contiene en este ejemplo 4 registros lógicos.

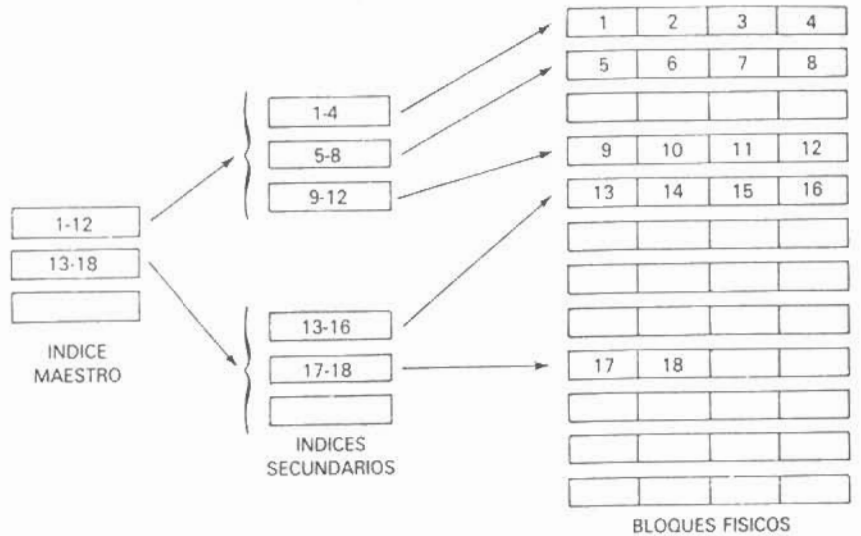


Figura 9.18. Acceso directo indexado.

Para agilizar los accesos, normalmente se mantiene el índice de mayor nivel en memoria principal. Estos índices se crearán durante la carga del archivo.

9.8. DIRECTORIOS DE ARCHIVOS

Hasta ahora hemos analizado las diferentes técnicas que permiten la creación de un archivo en disco y su utilización posterior según unos métodos de acceso definidos.

Ahora bien, en los sistemas existirán diferentes archivos (del propio sistema y de los usuarios) que debe controlar el subsistema de archivos para poder atender las peticiones que sobre ellos se realicen (creación, lectura, ejecución, etc.). Para ello se utilizan directorios que son a su vez archivos de estructura especial, cuyos registros contienen la descripción de los archivos existentes en el sistema. Mediante estos directorios, el subsistema podrá localizar rápidamente un archivo solicitado (en qué volumen está) y conocer sus características (organización, edad, tamaño, protección, etc.).

Desde el punto de vista lógico, un directorio es una tabla de símbolos que refleja los archivos existentes. La estructura de dicha tabla podrá ser más o menos compleja dependiendo de los distintos sistemas operativos, pero en cualquier caso deberán permitir la realización de las siguientes operaciones básicas:

- **Búsqueda.** Esta operación debe localizar, si existe, un archivo determinado. En algunos casos se permite localizar grupos de archivos con una característica común (fecha de creación, primer carácter del nombre, etc.).
- **Creación.** La creación de un nuevo archivo exige añadir una nueva entrada en el directorio adecuado.

- **Borrado.** Una vez desechado un archivo, se debe borrar para poder reutilizar el espacio que ocupa en memoria secundaria. Habrá que anular también su entrada en el directorio.
- **Listado.** Permitirá listar los archivos de un directorio así como el contenido de la entrada de un archivo en dicho directorio.

Pasemos a analizar algunas estructuras de directorios.

▪ Directorios de un nivel

En sistemas pequeños es suficiente disponer de un solo directorio que contendrá la descripción de todos los archivos disponibles.

Un ejemplo de este tipo es el directorio de dispositivo o tabla de contenidos del volumen (VTOC) donde, como ya sabemos, se guarda la descripción de todos los archivos existentes en ese dispositivo y del espacio disponible (Figura 9.19).

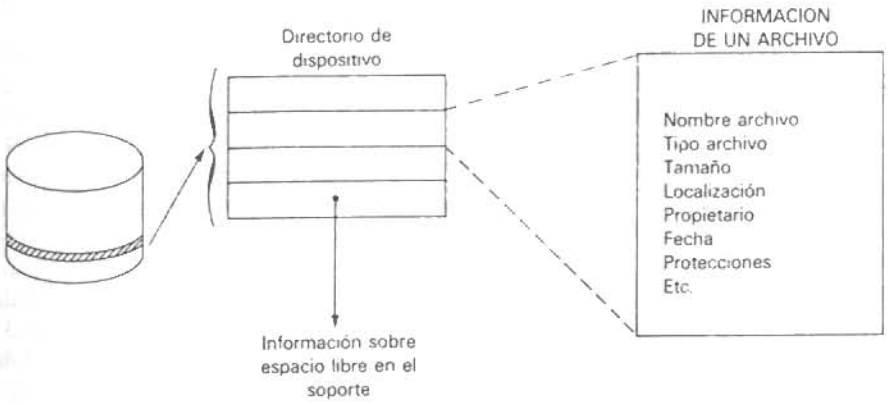


Figura 9.19. Directorio de dispositivo.

El directorio de un dispositivo o bien está en una dirección del disco predeterminada (por hardware o por el sistema) o estará apuntado indirectamente por un bloque físico específico, de forma que su localización sea inmediata.

Este esquema de directorio de un nivel es sencillo de llevar a la práctica pero sus posibilidades son bastante limitadas. Si pensamos en un sistema con dicho esquema y varios discos, y deseamos acceder a un archivo, el subsistema deberá recorrer todos los directorios de los diferentes dispositivos hasta localizar el buscado. Conforme aumenta el número de discos, la búsqueda será más costosa y su duración muy dependiente del lugar que ocupe el dispositivo que contenga el archivo en el orden de búsqueda.

Por otro lado, si pensamos en sistemas de directorio único (Figura 9.20), nos vemos imposibilitados a definir dos o más archivos con el mismo nombre. Esto puede ser aceptable en sistemas monousuarios, pero cuando son varios los usuarios que crean archivos, la limitación es inaceptable.

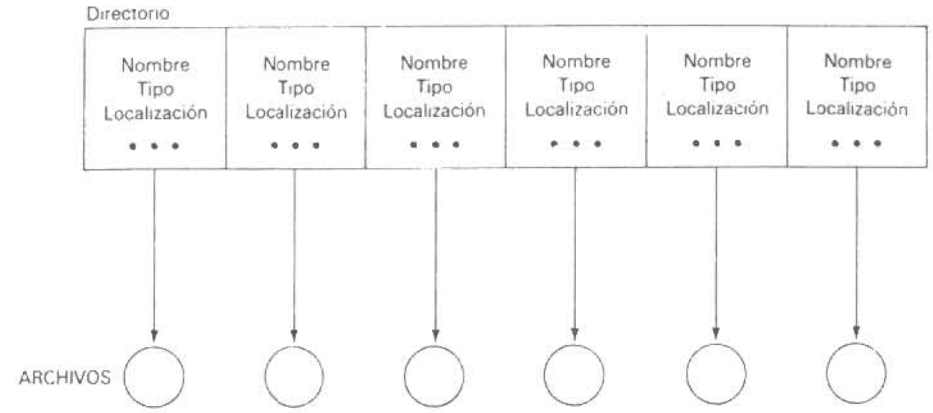


Figura 9.20. Directorio de un nivel.

En definitiva, el esquema de directorio de un nivel es sencillo y útil en sistemas pequeños, monousuario y con pocos archivos. Según va creciendo el número de usuarios y, por tanto, de archivos, se hacen necesarias estructuras de directorios más potentes.

▪ Directorios de dos niveles

Una solución más eficiente que la anterior es definir un directorio para cada usuario donde quedarán reflejados los archivos que le pertenecen. Por su parte, los diferentes directorios de usuario se controlan mediante un **directorio maestro del sistema.** En este esquema, la acción combinada de ambos directorios definen la relación de pertenencia respecto a los usuarios y su organización lógica (Figura 9.21).

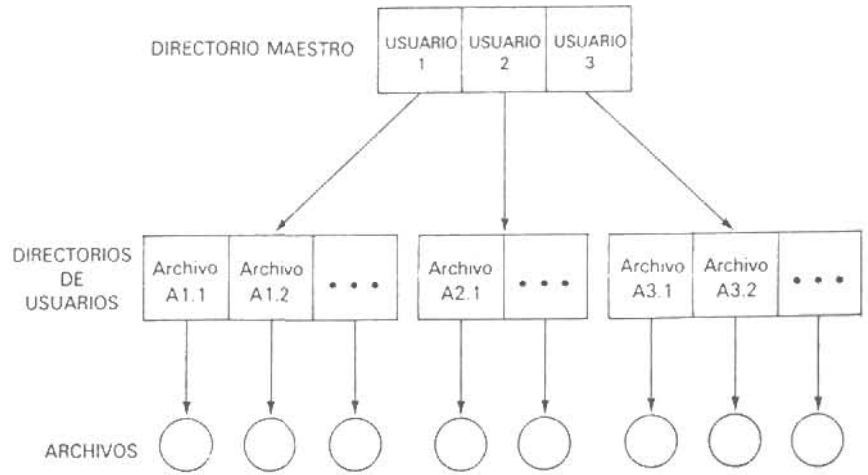


Figura 9.21. Directorios de dos niveles.

Ahora pueden existir varios archivos con el mismo nombre en directorios de usuarios diferentes.

Para localizar un archivo solicitado por el usuario, el subsistema lo buscará en el directorio del peticionario. Si crea un nuevo archivo, éste se añadirá en su directorio de igual forma que si borra un archivo, la acción sólo afectará al que tenga el nombre referenciado en dicho directorio y no en el de otros usuarios.

En este tipo de directorios es necesario disponer de una utilidad cuyo acceso será restringido al administrador del sistema, que permita dar de alta o baja a un usuario, es decir, incluir o eliminar el directorio del usuario y sus entradas en el directorio maestro.

Con esta organización de directorios, para localizar un archivo sólo necesitamos conocer su nombre y el del directorio de usuario que lo contiene. Estos dos nombres forman lo que se denomina **nombre** de **camino (path name)** exclusivo del archivo. La relación archivo-nombre de camino es biunívoca.

La utilización de los caminos para localizar los archivos permite que un usuario acceda a los archivos de otro usuario, indicando el camino adecuado. Esta flexibilidad la podrá controlar el subsistema mediante mecanismos de protección que veremos más adelante.

▪ Estructuras multinivel. Arboles de directorios

El esquema anterior puede considerarse como una organización en árbol de dos niveles, que generalizándola podemos pensar en estructuras en árbol de varios niveles.

Cada usuario podrá crear subdirectorios en su directorio, con lo cual podrá agrupar los archivos de la forma más adecuada. En general, al directorio maestro se denomina **raíz.**

Este esquema es muy utilizado en los sistemas operativos actuales donde cada directorio de usuario contiene entradas que apuntan a archivos o a subdirectorios que a su vez pueden apuntar a archivos o a subdirectorios de menor nivel.

Un usuario que se encuentre trabajando con el sistema en un momento determinado, tiene definido un **directorio actual** que será el punto de partida del camino de cualquier búsqueda pedida por él, además podrá en cualquier momento cambiar de directorio, para lo cual tendrá un comando a su disposición (Figura 9.22).

Algunos sistemas utilizan una **lista de búsqueda de directorios** en lugar del directorio actual. Cada usuario puede establecer su lista de búsqueda actual, que contendrá los directorios que el sistema deberá recorrer en cada petición, recorrido que realizará en el orden marcado por dicha lista. Mediante comandos se podrá cambiar la lista actual.

Con esta estructura de directorios un usuario puede agrupar sus archivos según sus criterios particulares (datos, programas en desarrollo, programas en explotación, etc.), logrando una organización más adecuada a sus necesidades.

▪ Otras estructuras de directorios

En los grandes sistemas multiusuario es muy corriente que varios usuarios deseen compartir archivos y directorios porque se encuentren en un mismo proyecto de trabajo o por cualquier otra causa. Un archivo o directorio compartido deberá estar apuntado en los directorios de los usuarios que lo comparten. Se obtiene así una estructura llamada **gráfico acíclico** (sin ciclos o anillos) más flexible que la de árbol, pues posibilita a los usuarios a compartir sus programas y datos (Figura 9.23).

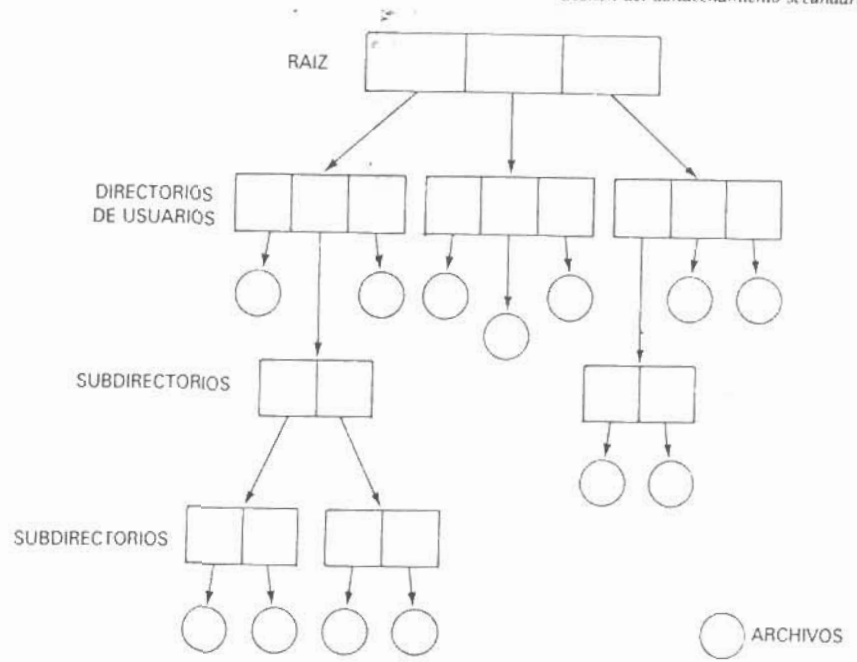


Figura 9.22. Directorios multinivel.

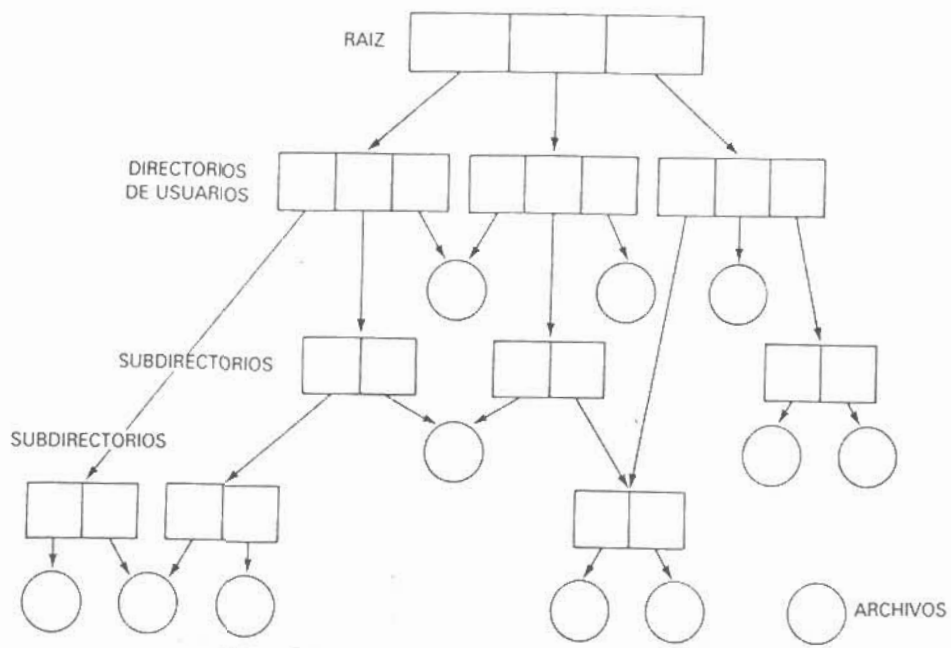


Figura 9.23. Estructura de directorios acíclica.

En aquellos directorios que vayan a compartir un archivo ajeno (cuyo propietario es otro) se crea una entrada especial llamada **enlace** o **link** que apunta a dicho archivo.

Compartir archivos introduce nuevos problemas. Dos directorios que comparten un archivo tienen reflejada la dirección física del mismo. Si uno de los usuarios realiza operaciones de actualización modificará el tamaño del archivo y estas operaciones deben quedar reflejadas en ambos directorios. Este tipo de problemas se pueden resolver de dos formas:

- **Direccionar indirectamente los archivos.** Los directorios apuntan a una estructura de datos que contiene la dirección física y el tamaño del archivo.
- **Enlazar los archivos entre los usuarios.** Si un usuario desea compartir un archivo de otro usuario, se crea en el directorio del primero un nuevo archivo llamado **link** que contiene el camino a dicho archivo.

9.9. SEGURIDAD DE LOS ARCHIVOS

La finalidad principal de una computadora es el tratamiento de la información así como el almacenamiento de los datos para posteriores utilizaciones. La pérdida o alteración de estos datos causaría trastornos que en ocasiones pueden ser irresolubles.

Pensemos por ejemplo, en una computadora en cuyos archivos se haya almacenada la información operativa de una empresa (clientes, proveedores, contabilidad, nóminas, previsiones, etc.). Si por cualquier razón, accidental o no, se destruye dicho equipo, sería fácil reponer el hardware (comprando otro equipo) pero el contenido de sus archivos (lo más importante para la empresa) sería imposible de recuperar si previamente no se han tomado las medidas adecuadas.

La seguridad de los archivos, de su contenido, se debe enfocar bajo dos aspectos: disponibilidad de la información (los archivos contienen la información prevista y se puede acceder a ella) es uno de ellos, y su privacidad (control de acceso a los mismos) es el otro.

9.9.1. Disponibilidad de los archivos

El objetivo fundamental es lograr una buena disponibilidad de la información contenida en los archivos, es decir, que pueda accederse a ella en el momento en que se precise.

La primera técnica utilizada consiste en la realización periódica de **copias de seguridad (back-up)** del contenido de los archivos para que en caso de destrucción de éstos se pueda recuperar dicho contenido a partir de las copias. Existen en los sistemas operativos actuales programas de utilidad para la realización automática de estas copias así como para la recuperación de su información.

Estas copias se realizan generalmente, en cinta magnética que se guarda en lugar separado de la computadora. De esta forma, a partir de ella se podría restaurar los archivos en otras instalaciones si la propia sufriera algún tipo de accidente (Figura 9.24).

El esquema de esta primera técnica cubrirá todas las necesidades de disponibilidad en una instalación en la que sólo se procesaran trabajos batch. Si durante la ejecución de un

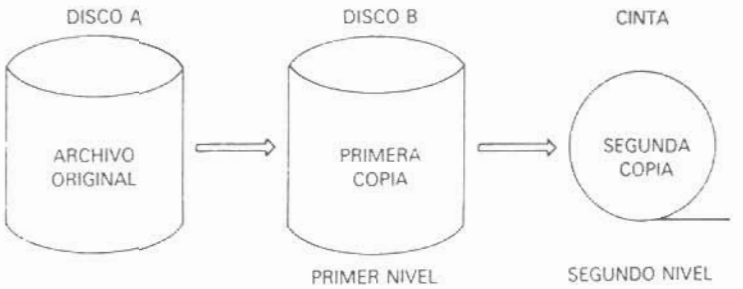


Figura 9.24. Copias de seguridad.

trabajo que se encuentra actualizando algún archivo se produjera algún accidente, bastará con restaurar la copia disponible y repetir los trabajos de actualización necesarios. Sin embargo, en los sistemas interactivos esto no basta.

Los sistemas de tiempo compartido atienden a numerosos usuarios que, entre otras operaciones, pueden estar actualizando sus archivos. En estos sistemas se recurre a la utilización de archivos auxiliares llamados comúnmente archivos **LOG**, donde se va reflejando cada operación que actualice los archivos y guardando la información añadida o borrada. Existen programas de utilidad en los sistemas operativos que pueden recuperar el estado actual de los archivos partiendo de una copia anterior y realizando las actualizaciones que indique el archivo LOG.

Esta segunda técnica permite asegurar la consistencia del contenido de los archivos ante caídas inesperadas del sistema, evitando que una información se queda a medias de escribir.

Para solucionar problemas de consistencia, algunos sistemas no dan la operación de escritura por realizada hasta que no se refleja en el LOG, y esto se hace una vez confirmada la escritura en el disco. Al volver a arrancar, el sistema inspecciona el LOG buscando operaciones iniciadas y no acabadas, finalizándolas antes de permitir de nuevo el trabajo de los usuarios.

9.9.2. Privacidad de los archivos. Protección

El segundo aspecto para la seguridad de los archivos es el que debe controlar que la información contenida en los mismos no sea accedida por usuarios no autorizados.

El primer control consiste en la identificación del usuario por medio del **nombre de usuario** *(username)* y una **palabra clave** *(password)* intimamente ligada al anterior. El sistema validará o no el binomio anterior consultando la tabla de claves concediendo o denegando el solicitado acceso al sistema.

Generalmente, la información del citado archivo de claves se encuentra criptografiada desde que se guarda por primera vez, para evitar el descubrimiento de dichas claves ante listados de memoria *(dumps)* o accesos indeseados a dicho archivo.

Un problema importante es la tendencia de los usuarios a definir contraseñas triviales (nombres, iniciales, fechas, etc.), haciendo más fácil su descubrimiento. Algunos sistemas in-

tentan solucionar este riesgo exigiendo periódicamente el cambio de la clave al usuario, que no podrá ser igual, por ejemplo a las 10 últimas utilizadas. Por otro lado, despúes de recibir un número determinado de contraseñas erróneas durante un intento de conexión, el sistema anula temporalmente al usuario.

Una vez introducido en el sistema, el usuario puede acceder a diferentes recursos (programas, archivos, directorios, dispositivos, etc.) y aquí debe establecer el sistema un nuevo nivel de control.

Para enfocar este segundo nivel de protección se recurre al concepto de **dominio,** que es un conjunto de recursos y de operaciones permitidas sobre los mismos. Por ejemplo, un dominio podría estar constituido por una impresora y el derecho de realizar operaciones de salida sobre ella (Figura 9.25).

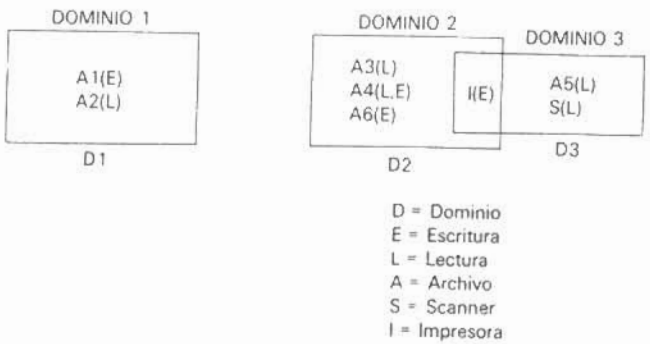


Figura 9.25. Dominios de protección.

Cualquier proceso está, en todo momento, ejecutándose en un dominio de protección, o lo que es lo mismo, podrá realizar determinadas operaciones sobre algunos recursos. Dependiendo de los diferentes sistemas, los procesos pueden cambiar de dominio en determinadas circunstancias.

Un factor importante de diseño es la forma en la que el sistema va a almacenar toda la información relativa a sus dominios. En principio se puede optar por una **matriz de dominios** cuyas filas indiquen los dominios existentes, y sus columnas los recursos. Cada elemento de esa matriz es la lista de derechos del dominio correspondiente sobre el recurso indicado por su columna.

La Figura 9.26 seria la matriz correspondiente a los dominios de la Figura 9.25.

	A1	A2	A3	A4	A5	A6	I	S
D1	E	L						
D2			L	L,E		E	E	
D3					L		E	L

Figura 9.26. Matriz de accesos.

- **Listas de acceso.**

Esta técnica almacena la información de las columnas de la matriz, asociando con cada objeto de protección o recurso una lista ordenada de los dominios que pueden utilizarlo y cómo pueden hacerlo. Esta lista se denomina **lista de acceso.**

En el caso de archivos se puede almacenar dicha lista en la entrada correspondiente del directorio al que pertenezcan. Los directorios como archivos que son, también tendrán su lista de acceso en el directorio de nivel superior.

- **Listas de capacidades**

Se puede recurrir también a almacenar las filas de la matriz de accesos. Cada dominio se asocia con una lista de recursos a los que está capacitado para acceder y en qué modalidad. Son las denominadas **listas de capacidades** (Figura 9.27).

DOMINIOS	TIPO	DERECHOS	OBJETOS
0	A	L	Apuntador a A3
1	A	L,E	Apuntador a A4
2	A	L,E	Apuntador a A5
3	I	E	Apuntador a I

Figura 9.27. Lista de capacidades.

Toda la gestión de estas listas de control (crearlas, borrarlas, modificarlas, etc.) es de uso exclusivo del administrador del sistema.

En los sistemas grandes es necesario controlar el uso de muchos recursos por un gran número de usuarios (dominios), y ello exige, cada vez más, la existencia de personas dedicadas a administrar la seguridad del sistema y la eficacia de la protección utilizada. Para apoyar esta labor, los sistemas suelen mantener un archivo de incidencias en el que quedan reflejados los intentos de acceso ilegal, altas y bajas de nuevos recursos y dominios, etc.

9.10. DISEÑO DEL SUBSISTEMA DE ARCHIVOS

La realización de un subsistema de archivos debe afrontar dos aspectos diferentes:

- La visión del usuario. Los servicios que éste puede solicitar.
- La realización mediante algoritmos y estructuras de datos de la visión del usuario.

Por ello, el subsistema de archivos se compone generalmente de varios niveles organizados jerárquicamente (Figura 9.28).

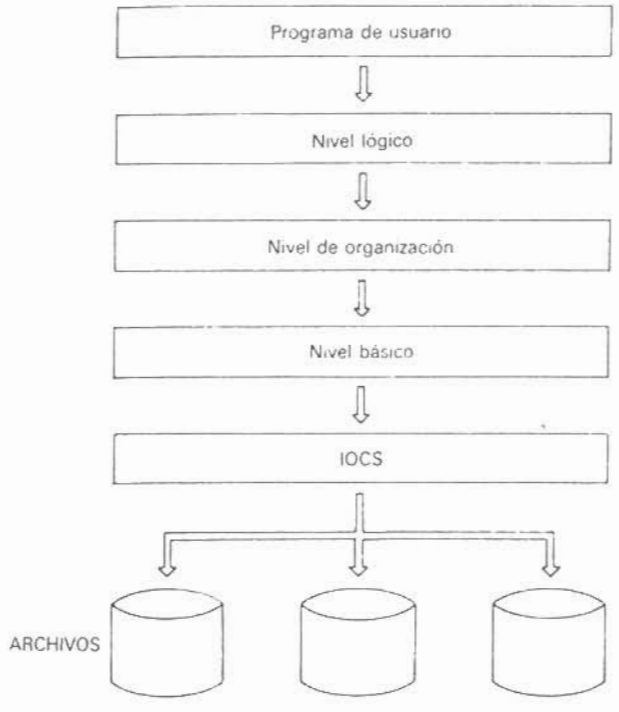


Figura 9.28. Jerarquía del subsistema de archivos.

Ante una petición del programa de usuario, el nivel lógico localiza el archivo deseado a través de la estructura de directorios. El nivel de organización genera las direcciones de los bloques correspondientes a la petición hecha y las pasa al nivel básico. Este último suministra a los IOCS los comandos necesarios para realizar las lecturas o escrituras de los mencionados bloques de información. El IOCS es un elemento que contiene el gestor de interrupciones y los drivers de dispositivos.

9.11. TENDENCIAS ACTUALES

La utilización cada vez más extendida de grandes archivos exige agilizar al máximo la velocidad de acceso a los datos, pues sólo así es posible gestionar informáticamente con eficacia grandes cantidades de información.

Para lograrlo se recurre a la utilización de memorias cache en los controladores de los discos. En dichas memorias el subsistema mantiene los bloques más buscados y, ante cada petición de un registro, busca en primer lugar en la memoria cache. En caso de encontrarlo

se habrá evitado un acceso al disco, y si no es así, deberá acceder al disco y actualizar el contenido de la memoria cache. En estos casos se plantean algoritmos para el reemplazamiento similares a los analizados para la sustitución de páginas en la memoria principal.

En atención al usuario, la tendencia apunta a independizarlo cada vez más de la organización y manejo del contenido de los archivos. Las bases de datos relacionales como conjunto de archivos interrelacionados y organizados por un software específico, permiten que el usuario disponga de la información almacenada con mucha facilidad y variedad de opciones, y todo ello a través de lenguajes coloquiales.

Esta mayor flexibilidad y transparencia para el usuario final se traduce en una mayor complejidad del sistema operativo.

CUESTIONES

1. Comentar brevemente la justificación de la necesidad de almacenar información a lo largo del tiempo en un sistema informático.
2. Definir conceptos de bases de datos, archivo, registro, campo, campo clave, cadena de caracteres y carácter.
3. Realizar un esquema simplificado de la jerarquía existente entre los distintos elementos de la información.
4. Realizar una definición de soporte de información y unidad de entrada/salida e indique sus diferencias.
5. ¿En qué se diferencian los conceptos de registro físico y registro lógico?
6. ¿Cuáles son las tres operaciones que tiene que realizar el hardware de una unidad de disco ante una petición de acceso a un bloque determinado?
7. ¿En qué consiste el algoritmo de planificación FCFS para las peticiones de acceso a un disco?
8. ¿Cuáles son las ventajas e inconvenientes del algoritmo SSTF frente al FCFS en la planificación de los accesos a disco?
9. ¿En qué se basan los algoritmos Scan y C-Scan de planificación de accesos a disco?
10. ¿Qué es el subsistema de archivos y cuáles son sus misiones?
11. ¿Qué métodos utiliza el subsistema de archivos para controlar el espacio ocupado y disponible en un disco?
12. Definir qué es el directorio de dispositivo.
13. Enumerar la información habitual que sobre un archivo se encuentra en el directorio al que pertenece.
14. ¿Cuáles son las estrategias que utiliza el subsistema de archivos para la asignación de espacio en un disco?
15. Comente brevemente los distintos métodos de acceso que se utilizan para recuperar la información contenida en un disco.
16. ¿Qué operaciones básicas debe permitir realizar sobre los archivos el directorio correspondiente?
17. Realizar un esquema de los distintos tipos de directorios de archivos existentes.
18. ¿Por qué se hace necesario disponer mecanismos de protección de la información contenida en los archivos?

CAPITULO **10**

Seguridad en los sistemas operativos

10.1. INTRODUCCION

En este capítulo se pretenden analizar los aspectos que intervienen en la seguridad de los sistemas informáticos en general y en los sistemas operativos en particular.

El estudio comienza haciendo balance de los requerimientos y mecanismos necesarios para poseer una buena seguridad informática tanto de los equipos como de los programas y datos.

A continuación se analizan cuestiones de seguridad desde dos perspectivas diferentes, la seguridad externa y la interna (Figura 10.1).

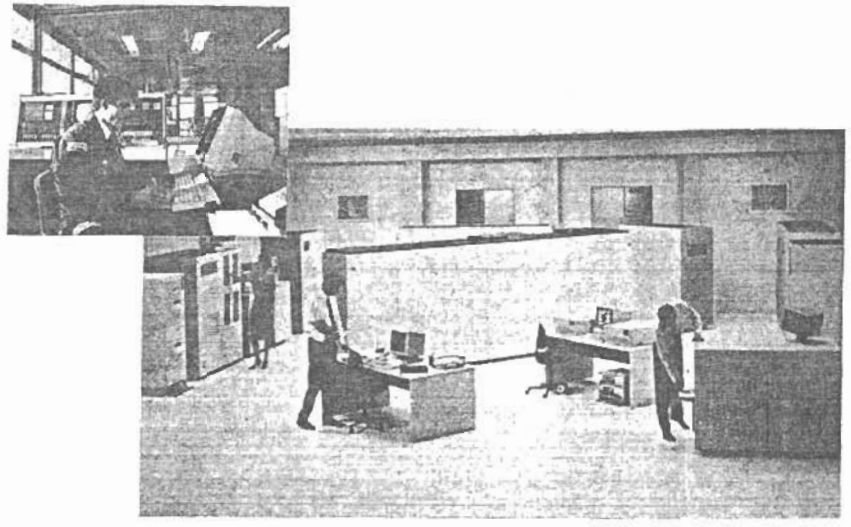


Figura 10.1. Seguridad en los sistemas operativos. (Cortesia de Grupo 4 Securitas España.)

10.2. DIRECTRICES Y MECANISMOS DE SEGURIDAD

En un sistema informático, con el fin de asegurar la integridad de la información contenida en él, se precisa describir las directrices necesarias y los mecanismos capaces de satisfacerlas para lograr dicho fin. Dependiendo de los mecanismos utilizados y de su grado de efectividad, se puede hablar de sistemas seguros e inseguros.

En primer lugar, deben imponerse ciertas características en el entorno donde se encuentra la instalación de los equipos, con el fin de impedir el acceso a personas no autorizadas, mantener un buen estado y uso todo el material y equipos, así como eliminar los riesgos de causas de fuerza mayor (por ejemplo incendios, inundaciones, etc.) que puedan destruir la instalación y la información contenida.

En la actualidad son muchas las violaciones que se producen en los sistemas informáticos, en general por accesos de personas no autorizadas que obtienen información confidencial, pudiendo incluso manipularla. En ocasiones, este tipo de incidencias resulta grave por la naturaleza de los datos; por ejemplo, si se trata de datos bancarios, datos oficiales que pueden afectar a la seguridad de los Estados, etc.

Ahora bien, no todas las violaciones se deben a accesos no permitidos, sino que pueden producirse por muy diversas causas. Entre éstas, una que últimamente se ha puesto de moda con el progreso de la microinformática es el **software malintencionado**, es decir, pequeños programas que poseen una gran facilidad para reproducirse y ejecutarse, cuyos efectos son destructivos y el daño en la mayoría de los casos es irreversible. Nos estamos refiriendo a lo que en términos populares se ha dado en llamar **virus informático** (Figura 10.2).

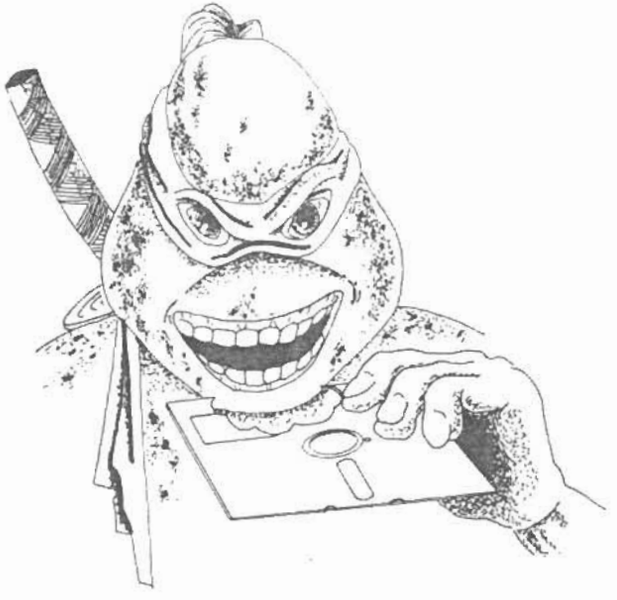


Figura 10.2. Virus informático.

Por todo ello, los gobiernos de los distintos países han dictado leyes y normas para asegurar una racional seguridad en los sistemas de información y proteger el derecho a la intimidad de la información de las personas.

Por último, un aspecto que no debemos olvidar es el de evaluar el nivel de seguridad que una instalación necesita. Dependerá fundamentalmente de la importancia de los datos, su grado de confidencialidad, etc.; por tanto, no será igual la seguridad que necesita una instalación bancaria que un equipo de uso doméstico.

10.3. SEGURIDAD EXTERNA

En un sistema informático todos los mecanismos de seguridad tienen que complementarse entre sí, de tal forma que si una persona logra saltarse alguna de las protecciones, se encuentre con otras que le hagan el camino difícil.

Todos los mecanismos dirigidos a asegurar el sistema informático sin que el propio sistema intervenga en el mismo se engloban en lo que podemos denominar **seguridad externa** (Figura 10.3).

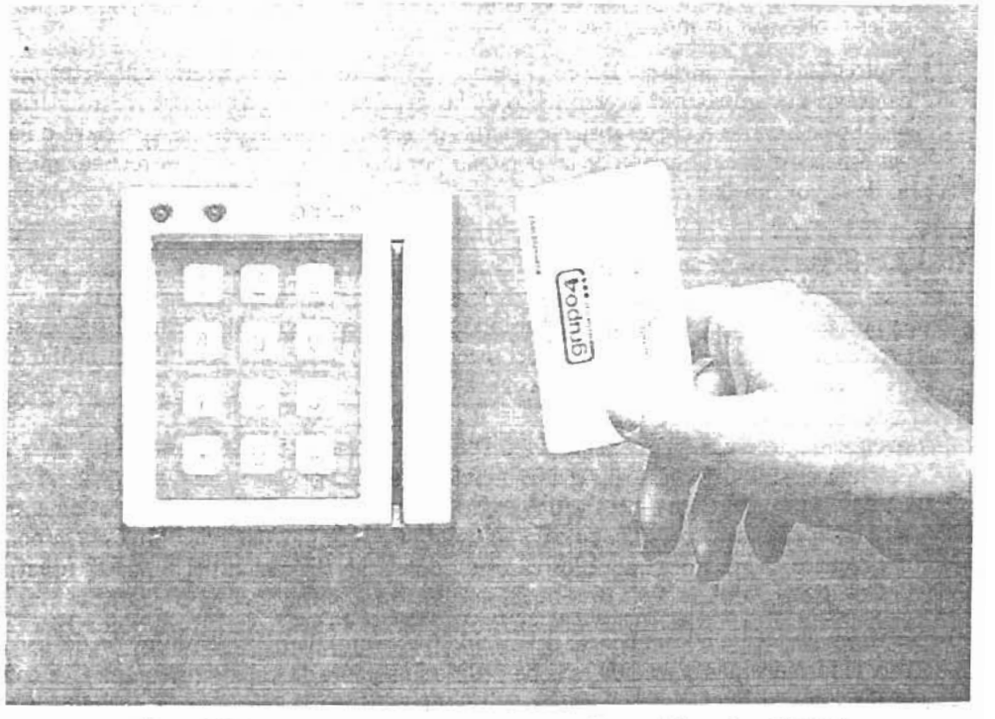


Figura 10.3. Seguridad externa. (Cortesía de Grupo 4 Securitas España.)

La seguridad externa puede dividirse en dos grandes grupos:

- **Seguridad física.** Engloba aquellos mecanismos que impiden a los agentes físicos la destrucción de la información existente en el sistema; entre ellos podemos citar el fuego, humo, inundaciones, descargas eléctricas, campos magnéticos, acceso físico de personas con no muy buena intención, etc.
- **Seguridad de administración.** Engloba los mecanismos más usuales para impedir el acceso lógico de personas físicas al sistema.

10.3.1. Seguridad física

Como ya hemos mencionado, se trata de eliminar los posibles peligros que originan los agentes físicos o la presencia física de personas no autorizadas. Para ello podemos considerar los siguientes aspectos:

- **Protección contra desastres.** Consta de elementos de prevención, detección y eliminación que actúan contra incendios, humos, sobretensiones, fallos en el suministro de energía, etc. También es necesario controlar la temperatura y limpieza del medio ambiente en que se encuentran los equipos, instalando aire acondicionado, falso suelo, ventilación, y, en definitiva, tomando en consideración todo aquello que pueda causar cualquier problema a la instalación.
- **Protección contra intrusos.** Desde el punto de vista físico, es necesario establecer mecanismos que impidan el acceso físico de las personas no autorizadas a las instalaciones. Suele llevarse a cabo mediante puertas de seguridad con apertura por clave o llaves especiales, identificación de las personas por tarjetas de acceso o por reconocimiento de la voz, huellas digitales, etc.

10.3.2. Seguridad de administración

Comprende aquellos mecanismos cuya misión es dar acceso lógico al sistema. Este acceso puede realizarse a través de un terminal del sistema o bien desde otro sistema por medio de una red de comunicación a la que estén conectados ambos sistemas.

▪ Protección de acceso

Se trata de un mecanismo para el control de los intentos de entrada o acceso al sistema, de tal forma que permita la conexión cuando un usuario lo solicite y pase el control correspondiente y rechace el intento en aquellos casos en que la identificación del supuesto usuario no sea satisfactoria.

- **Palabra de acceso o identificador del usuario (password).** Para la identificación del usuario, la fórmula más extendida es la de pedirle su **nombre de usuario (username)** y a continuación la palabra clave tal que el mecanismo accede al archivo correspondiente para contrastar los datos recibidos y aceptar o rechazar el intento. Los intentos fallidos de acceso son registrados por el sistema con el fin de que el administrador del sistema pueda estudiar cada cierto tiempo si se está o no intentando transgredir la seguridad del sistema.

El sistema operativo dota al administrador del sistema para que en cualquier momento se pueda dar de alta o de baja a un usuario, asignándole en el primer caso, además de un username, la correspondiente contraseña o password inicial. Mientras que el nombre de usuario es público, la password no lo es, siendo recomendable su cambio cada cierto tiempo, así como no tenerla escrita en ninguna otra parte que en la propia mente del usuario.

La password cuando se escribe en un terminal, tanto para acceder al sistema como para su cambio, no aparece en la pantalla como ocurre en el resto de datos que se teclean, para así conservar el secreto de la misma. Además, esta palabra se graba en los archivos de administración del sistema codificada o encriptada para que no sea fácilmente reconocible por las personas.

Al proceso de petición de entrada a un sistema, contestación a las preguntas de identificación, contrastación de los datos recibidos y dar el correspondiente acceso se denomina **login** Figura 10.4). Asimismo, al proceso de despedida del sistema se le llama **logout.**

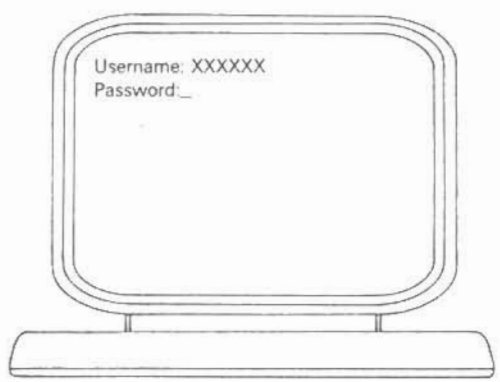


Figura 10.4. Proceso de acceso al sistema.

- **Criptografía.** Es un proceso de transformación que se aplica a unos datos para ocultar su contenido. El proceso al que hay que someter la información para conseguir que sea secreta se conoce con el nombre de **encriptado** o **cifrado,** denominándose la información antes del proceso como **texto claro** y después del mismo **texto cifrado** (Figura 10.5).

La mayoría de los sistemas que utilizan algoritmos de cifrado exigen que el texto cifrado pueda convertirse en texto claro. Para ello existen diversas técnicas, algunas de las cuales describimos brevemente a continuación:

a) **Or-exclusivo.** Es un método sencillo que ofrece una gran seguridad. Consiste en tomar la información a cifrar y aplicar a cada octeto la operación or-exclusivo con

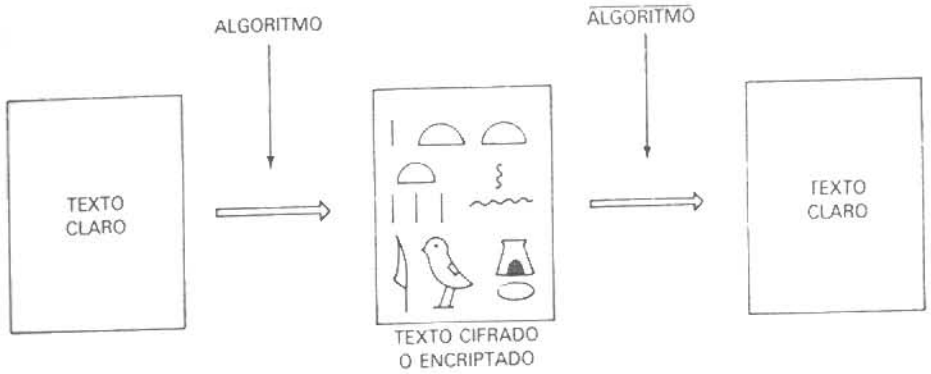


Figura 10.5. Criptografiado de la información.

una clave cuya longitud debe ser, en número de caracteres, tan larga como el mensaje a cifrar. El algoritmo de descifrado es similar al de cifrado utilizando la misma clave. Esta clave se tiene que cambiar cada cierto tiempo para mantener un buen grado de seguridad.

b) **Estándar de Encriptado de Datos (Data Encryption Standard-DES).** Es un método desarrollado en la Oficina Nacional de Estándares de Estados Unidos, siendo uno de los más utilizados actualmente. El algoritmo lleva asociado un chip especialmente construido para este fin, aunque puede ser simulado por software, y se basa en claves de 56 bits de longitud.

c) **Método de Rivest, Shamir y Adelman (RSA).** Es un algoritmo que utiliza distinta clave para el cifrado y descifrado, ofreciendo con ello un alto índice de seguridad.

■ Seguridad funcional

Engloba aspectos relativos al funcionamiento del sistema y a la seguridad que de las instalaciones se pretende tener.

- **Seguridad en la transmisión de datos.** En las líneas de transmisión de datos existen diversos problemas de seguridad debido a lo fácilmente violables que son dichas líneas. Por esta razón, para enviar datos a través de líneas de comunicación entre computadoras se siguen diversas técnicas, como son:

  a) **Compactación de datos.** Consiste en comprimir los datos para que ocupen el menor espacio posible y así conseguir en principio que la duración de la transmisión sea menor, y que para entenderla haya que descompactarla; por tanto, la información va relativamente cifrada. Existen muchos métodos de compactación de datos, de los cuales los más utilizados son:

  1. **Reducción de espacios en blanco.** Un archivo de información puede tener muchos espacios en blanco que pueden ser sustituidos por un número que indique cuántos de ellos están de forma consecutiva en un determinado punto.

  2. **Codificación por diferencia.** En ella se transmiten únicamente las diferencias existentes entre la información que se quiere enviar y la misma información ya enviada previamente, de tal forma que en el destino se puede reconstruir la información sin grandes dificultades. Se trata de un caso similar a las copias de seguridad (Backup) incrementales, donde cada nueva copia sólo registra las diferencias que existan entre el estado actual de la información y el original, con lo que se logra un importante ahorro de memoria.

  b) **Criptografía.** Similar al proceso ya mencionado para ocultar la información en una transmisión.

  c) **Fiabilidad.** Además de las medidas anteriores, se suelen tomar otras para asegurar el correcto estado de la información al llegar a su destino. Se pueden presentar problemas debidos a causas accidentales, como la influencia de fuertes campos magnéticos, perturbaciones eléctricas, etc., así como por motivos de intrusión en las comunicaciones con el fin de destruirlas o modificarlas. También pueden producirse errores por colisiones entre mensajes en redes locales y un sinfín de otras causas de diversa naturaleza.
  Para evitar todo tipo de incidencias, se suele añadir a la información una pequeña parte que nos permitirá saber si los datos recibidos coinciden con los enviados o no. Los métodos más utilizados para dotar de fiabilidad a una transmisión de datos son mecanismos hardware o software que permiten detectar errores ocurridos en una comunicación e incluso recuperar algunos de ellos. Citaremos los siguientes métodos:

  1. **Bit de paridad.** Consiste en añadir un bit a cada octeto o palabra que se transmita para con él conseguir que la suma de unos sea par (paridad par) o impar (paridad impar). Con este método se detectan errores al variar un bit o un número impar de ellos sin que se detecten variaciones de un número par de bits. Se sabe que la mayoría de errores que se producen en condiciones normales sólo afectan a un bit.

  2. **Códigos de Hamming.** Añaden varios bits de control al octeto o palabra a transmitir, de tal forma que detectan errores de uno o más bits y los corrigen.

  3. **Código de redundancia cíclica (CRC).** Si se prevé que los daños esperados en una transmisión no sean de un bit en un octeto o palabra, sino en una secuencia de ellos, se puede utilizar un algoritmo que permita realizar una suma denominada **suma de chequeo (Checksum)** y aplicar el método denominado de redundancia cíclica durante la transmisión, de tal forma que al terminar ésta se repite en el destino el mismo algoritmo de suma, comprobándose si el valor final de la suma es el mismo.

- **Sistemas tolerantes a fallos.** Se utilizan en sistemas donde se pueda perder información debido a un mal funcionamiento de los mismos. Este aspecto es muy importante en los sistemas de control y supervisión en tiempo real. Existen mecanismos que ante situaciones de mal funcionamiento consiguen recuperar y controlar el entorno, prote-

giendo fundamentalmente la información. Este tipo de mecanismos se basa en redes de dos o más computadoras conectadas entre sí de manera que, ante el mal funcionamiento de una de ellas, éste se pondrá en situación de inactivo, tomando el control cualquiera de los otros que estén conectados.

10.4. SEGURIDAD INTERNA

Todos los mecanismos dirigidos a asegurar el sistema informático, siendo el propio sistema el que controla dichos mecanismos, se engloban en lo que podemos denominar **seguridad interna.**

10.4.1. Seguridad del procesador

Los mecanismos de protección del procesador son varios ya estudiados y que pasamos a enumerar:

- **Estados protegidos (Kernel) o no protegido (Usuario).**
- **Reloj hardware para evitar el bloqueo del procesador.**

10.4.2. Seguridad de la memoria

Se trata de mecanismos para evitar que un usuario acceda la información de otro sin autorización. Entre ellos citaremos dos:

- **Registros límites o frontera.**
- **Estado protegido y no protegido del procesador.**

Además se emplean para la memoria métodos como el de utilizar un bit de paridad o el checksum ya mencionado.

10.4.3. Seguridad de los archivos

La finalidad principal de las computadoras es la del tratamiento de la información que se almacena permanentemente en los archivos. La pérdida o alteración no deseada de dicha información causaría trastornos que podrían ser irreparables en algunos casos. Por eso es necesario tomar las correspondientes medidas de seguridad, que, como ya se ha comentado en el Capítulo 9, se deben enfocar desde dos aspectos diferentes: la disponibilidad y la privacidad de los archivos.

▪ Disponibilidad de los archivos

Un archivo debe tener la información prevista y estar disponible en el momento que un usuario la necesite. Hay que tener presente la necesidad de asegurar tal circunstancia y para ello se suelen realizar las siguientes acciones:

- **Copias de seguridad (backup).** Consiste en que cada cierto tiempo (hora, día, semana...) se realice una copia del contenido de los archivos, de forma que si se destruyen éstos, es posible la recuperación de los datos a partir de la última de las copias. La operación de realizar copias de seguridad, así como la recuperación de los datos a partir de las mismas, se suele hacer por medio de programas de utilidad del sistema operativo.
  La fiabilidad de las copias de seguridad dependerá fundamentalmente de la periodicidad con que se realicen y del índice de actividad de los archivos, es decir, del ritmo al que se actualicen.
  Las copias de seguridad suelen realizarse sobre cinta magnética, guardándose en dependencias alejadas del sistema y en armarios protegidos incluso contra incendios.
  Al margen de las copias de seguridad, en muchos casos es conveniente mantener los archivos duplicados en el mismo o distinto disco, para que en caso de problemas locales en el archivo original se pueda tener una rápida recuperación (Véase Figura 9.24).
  En los grandes sistemas se tiende a automatizar los procesos de copias de seguridad por medio de un software que periódicamente revisa la fecha de la última copia de cada archivo, así como su último proceso de actualización, y a través de unos parámetros prefijados decide en qué archivos deben ser procesadas sus copias.
- **Archivos LOG.** En sistemas de tiempo compartido donde trabajan simultáneamente muchos usuarios, que entre otras operaciones llevan a cabo numerosas actualizaciones y modificaciones de archivos, no son suficientes las periódicas copias de seguridad para afrontar la pérdida de la información. Si la computadora falla por cualquier motivo en medio de una sesión donde hay un gran número de usuarios trabajando, se puede recuperar la información de los archivos desde la última copia de seguridad; pero esto puede no ser suficiente, por lo cual se recurre en estos sistemas a archivos auxiliares donde se registran todas las operaciones que realiza un usuario sobre sus archivos, almacenándose la nueva información o aquella que difiera de la ya existente. Estos archivos reciben el nombre de archivos LOG y son tratados por utilidades del sistema operativo conjuntamente con las copias de seguridad para los procesos de recuperación.

▪ Privacidad de los archivos

El contenido de los archivos se debe proteger de posibles accesos no deseados. Entre el peligro de permitir a todos los usuarios el acceso a cualquier archivo, y la rigidez de que cada usuario sólo pueda acceder a los suyos, el sistema de protección debe permitir accesos de forma controlada, según unas reglas predefinidas y con las consiguientes autorizaciones.

Cada usuario, al comenzar la sesión en un sistema tras su identificación, tiene asignado por el sistema de protección un dominio compuesto de una serie de recursos y de operacio-

nes permitidas, por ejemplo, una serie de archivos a los que acceder, no teniendo permitido el acceso al resto de archivos. En general, los sistemas operativos almacenan la información relativa a los dominios en lo que se denomina **matriz de dominios,** cuyas filas indican los dominios existentes y las columnas los recursos. Cada elemento de la matriz indica el derecho a utilizar el recurso correspondiente en el dominio (Véase Figura 9.26).

Si la matriz anterior tiene poca información, se recurre a otro tipo de almacenamiento de información sobre dominios, consistente en asociar a cada recurso una lista de dominios que pueden utilizarlo, denominándose este vector **lista de acceso.** También se puede obtener otro vector donde a cada dominio se le asigna una lista de recursos a los que puede acceder, denominándose en este caso **lista de capacidades** (Véase Figura 9.27).

En todos estos casos, la gestión de las listas de control se realiza mediante comandos de uso restringido, estando éstos únicamente disponibles para el administrador del sistema.

10.5. LEGISLACION SOBRE PROTECCION DE LA INFORMACION

Por último, debemos señalar que la información contenida en los archivos de las distintas instalaciones necesita, además, ser protegida para no infringir las leyes que aseguran el derecho a la intimidad de las personas y a su vida privada.

La legislación española contempla la protección de la información en el artículo 18, apartado 4, de la Constitución, aprobada por las Cortes y mediante referéndum en el año 1978. El texto del artículo es el siguiente:

> **«La ley limitará el uso de la informática para garantizar el honor y la intimidad personal y familiar de los ciudadanos y el pleno ejercicio de sus derechos.»**

Existen otros principios legales que hacen referencia indirecta a la protección de la información en cuanto al secreto de las comunicaciones, libertad de expresión, etc.

CUESTIONES

1. ¿Por qué se hace necesario tomar medidas de seguridad en un sistema informático?
2. ¿A qué se denomina seguridad externa de un sistema?
3. ¿Cuáles son las características que tiene y para qué sirve la palabra de acceso o password?
4. ¿En qué consiste el proceso LOGIN?
5. Explicar qué es, para qué sirve y cómo se puede llevar a la práctica la criptografía de datos.
6. Señalar las razones por las que se producen errores en líneas de transmisión de datos y cómo pueden corregirse.
7. ¿A qué se denomina seguridad interna de un sistema?
8. ¿En qué consisten las copias de seguridad de archivos (backup)?
9. ¿Para qué se utilizan los archivos comúnmente denominados LOG?
10. ¿Con qué fin utiliza un sistema operativo los elementos denominados matriz de dominios, lista de acceso y lista de capacidades?
11. ¿Qué agentes físicos tienen encomendados custodiar los mecanismos de seguridad física de un sistema?
12. ¿A qué se denomina «virus informático»?
13. Indicar en qué archivos y cómo se encuentran las password de los usuarios de un sistema informático.
14. ¿En qué consiste el método Or-exclusivo para encriptar información?
15. ¿Cuáles son los métodos más utilizados para dotar de fiabilidad a los datos que se transmiten por una línea de comunicación?

CAPITULO 11

Compiladores e intérpretes

11.1. INTRODUCCION

Todo sistema operativo en su entorno tiene un conjunto de **programas de proceso** encargados de la ayuda a los programadores en la realización y puesta a punto de los programas.

Entre los programas de proceso de un sistema operativo se encuentra un conjunto bien definido denominado **traductores** cuya misión es permitir el diseño de programas en lenguajes alejados de la máquina para en el momento en que se les solicite, realicen una traducción de dichos programas a lenguaja máquina para así poder ejecutarlos directamente por el hardware.

El proceso de traducción parte de un programa escrito en lenguaje generalmente de alto nivel y que recibe el nombre de **programa fuente** para producir otro equivalente (que represente el mismo trabajo) en lenguaje máquina correspondiente al procesador donde se va a ejecutar dicho programa, a este último se le denomina **programa objeto** que en ocasiones también recibe el nombre de **programa ejecutable** existiendo entre ambos alguna pequeña diferencia como puede ser alguna preparación o agrupamiento de varios programas objeto para configurar un sólo programa, misión que realiza el editor de enlace o montador.

El esquema general del proceso de traducción aparece en la Figura 11.1.

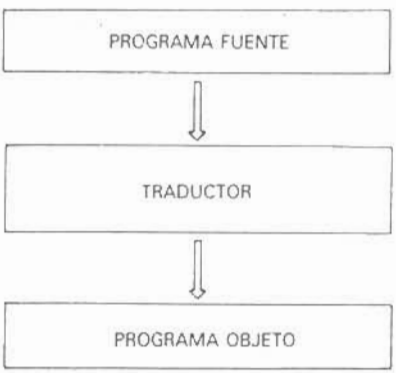


Figura 11.1. Esquema general del proceso de traducción.

Existen tres tipos de traductores:

■ Ensambladores

Son traductores que transforman programas fuente escritos en lenguaje simbólico de bajo nivel (denominados **lenguajes ensambladores** o **assemblers),** en programas objetos equivalentes escritos en lenguaje máquina. La traducción se realiza de tal forma que cada instrucción escrita en ensamblador se transforma en una única instrucción en lenguaje máquina. En definitiva se puede decir que el lenguaje ensamblador es una simplificación simbólica del lenguaje máquina y el **programa ensamblador** es su traductor.

Existen varios tipos de ensambladores entre los que citamos los siguientes:

- **Ensamblador cruzado** *(cross assembler).* Es un traductor de lenguaje ensamblador a lenguaje máquina que traduce en una computadora y ejecuta en otra distinta. La ventaja que ofrece este tipo de traductores es utilizar una computadora de características potentes para desarrollar programas que van dirigidos a otro cuya potencia y facilidades para el programador están ciertamente limitadas.
- **Macroensamblador** *(macroassembler).* Es un ensamblador que posee la característica de permitir el uso de lo que se denomina **macroinstrucción.** Una macroinstrucción no es más que un grupo de instrucciones que de forma global reciben un nombre simbólico al que se puede hacer referencia en un programa tantas veces como se desee. El macroensamblador colocará en la traducción el mencionado grupo de instrucciones en cada una de las referencias (expansión de macros).
- **Microensamblador** *(microassembler).* Es un traductor utilizado en la microprogramación que algunas computadoras tienen. Estos microprogramas permiten cierta flexibilidad al repertorio de instrucciones máquina de la computadora.
- **Ensamblador de una pasada o incrementales.** Son ensambladores que traducen en una sola pasada construyendo la tabla de símbolos a medida que van apareciendo las variables y etiquetas. Tienen el pequeño inconveniente de no permitir lo que se denomina referencias adelantadas, es decir, referencias a líneas de programa posteriores no traducidas.
- **Ensamblador de dos pasadas.** Realizan la traducción, a diferencia del anterior, en dos pasadas. En la primera leen el programa fuente construyendo la tabla de símbolos y asignando las correspondientes direcciones, y en la segunda vuelven a leer el programa traduciéndolo a lenguaje máquina. En este caso sí están permitidas las referencias adelantadas, siendo este tipo de ensambladores los más utilizados en la actualidad.

■ Compiladores

Son traductores encargados de transformar programas fuente escritos en lenguajes simbólicos de alto nivel en programas objetos escritos en lenguaje máquina. La traducción no suele ser directa, apareciendo un paso intermedio situado en un nivel similar al ensamblador. Una característica fundamental de este tipo de traductores es que se realiza la traducción completa, y en el caso de no existir errores se pasa a la creación del programa objeto. La traducción del programa se efectúa además de forma que cada instrucción del programa fuente se transforma en una o más instrucciones en el programa objeto.

■ Intérpretes

Son programas traductores que transforman programas fuente escritos en lenguaje de alto nivel en programas objeto escritos en lenguaje máquina. En estos programas intérpretes la traducción se realiza de forma que después de transformar una instrucción del programa fuente en una o varias instrucciones en lenguaje máquina, no esperan a traducir la siguiente instrucción, sino que inmediatamente la ejecutan.

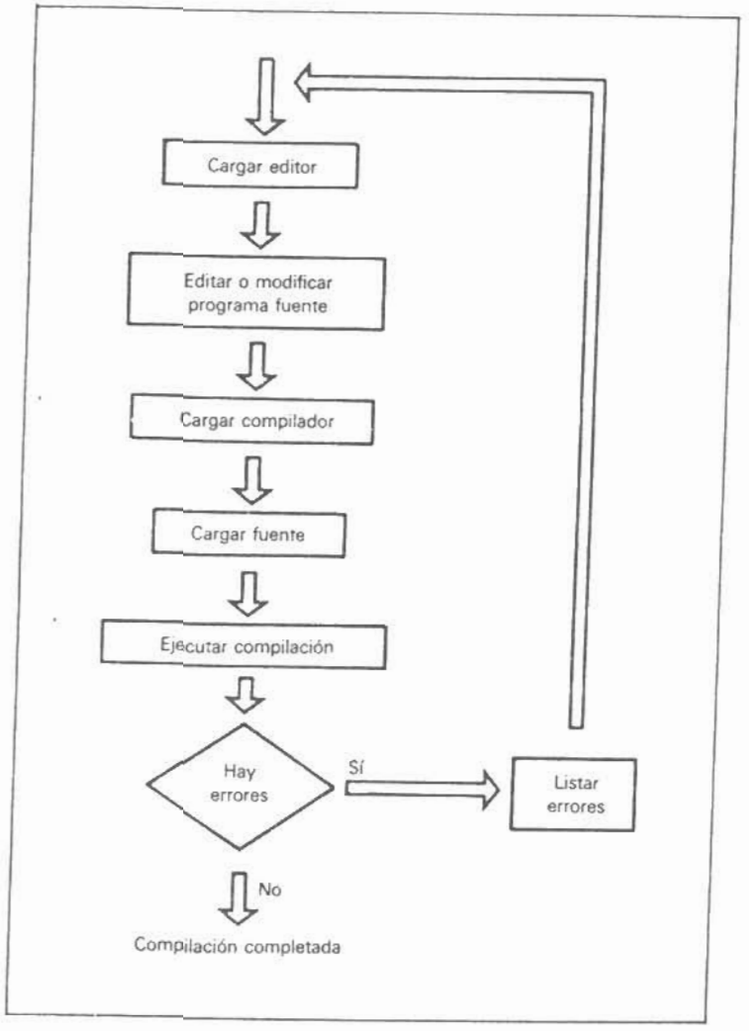


Figura 11.2. Proceso de compilación.

11.2. CONCEPTOS BASICOS

Por ser los compiladores los traductores más utilizados en la actualidad, desarrollamos el proceso de compilación, que consiste en la traducción de un programa fuente escrito en lenguaje de alto nivel en su correspondiente programa objeto escrito en lenguaje máquina, dejándolo listo para la ejecución con poca o ninguna preparación adicional. La Figura 11.2 representa el esquema general del proceso de compilación de un programa.

En primer lugar se crea el mencionado programa fuente, normalmente mediante un programa de utilidad denominado **editor** o con cualquier **procesador de texto** de que disponga la computadora que estemos utilizando.

Para ejecutar la compilación de un programa, éste debe estar en memoria principal simultáneamente con el compilador. El resultado de la misma puede ser que no se hayan producido errores, en cuyo caso aparecerá el programa objeto, o si se han producido errores obtendremos un listado de los mismos para proceder a su corrección por medio del editor y volver a empezar nuevamente el proceso.

En general, para obtener el programa ejecutable es necesario someter al programa objeto a un proceso de montaje donde se enlazan los distintos módulos en caso de trabajar con subprogramas y además se incorporan al mismo las **funciones intrínsecas** del lenguaje (SIN, LOG, etc.) procedentes de las **librerías** (Figura 11.3).

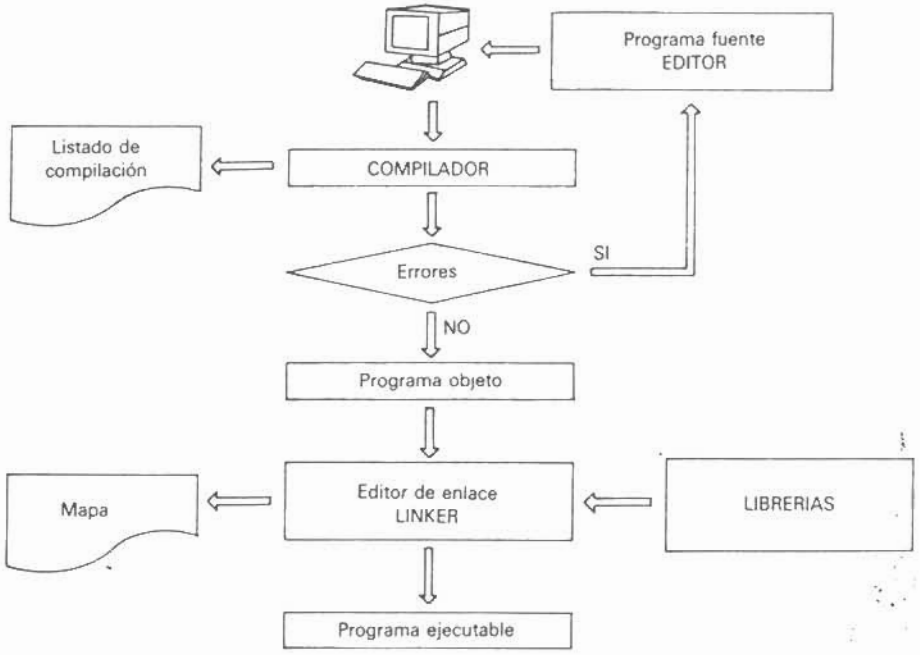


Figura 11.3. Compilación y montaje de un programa.

11.3. ESTRUCTURA GENERAL DE UN COMPILADOR

Un compilador, además de traducir, realiza otra serie de operaciones que en su mayoría están enfocadas a la detección de errores en la escritura del programa fuente. Por lo general, está constituido por los siguientes bloques:

- **Analizador lexicográfico** *(Scanner).*
- **Analizador sintáctico** *(Parser).*
- **Generador de código intermedio.**
- **Optimizador de código.**
- **Generador de código final.**
- **Tabla de símbolos.**
- **Módulo de tratamiento de errores.**

El compilador utiliza internamente una tabla de símbolos para introducir determinados datos que necesita, y está relacionada con todos sus elementos; asimismo, posee un módulo de tratamiento de errores también relacionado con sus elementos (Figura 11.4).

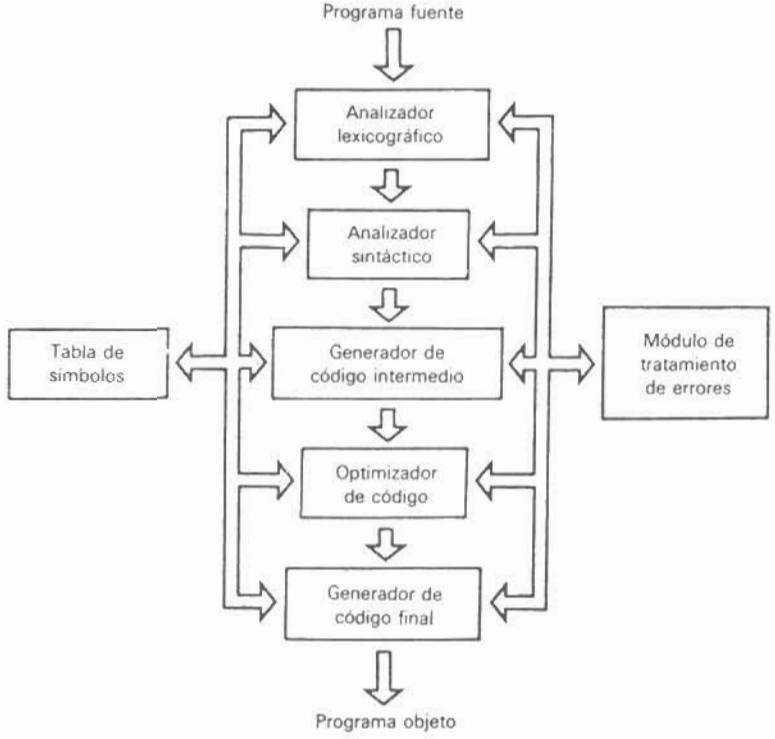


Figura 11.4. Estructura general de un compilador.

11.3.1. Análisis lexicográfico

El **analizador lexicográfico** también se denomina **scanner,** detecta en el programa fuente unidades básicas de información que pertenecen al lenguaje. Estas unidades básicas se denominan **tokens** o **unidades léxicas.** En el caso de no existir errores realiza una primera traducción a un código propio del compilador (generalmente de tipo numérico), eliminando a su vez toda información superflua como pueden ser los comentarios y blancos no significativos del programa fuente.

Un **token** es un elemento o cadena del programa fuente que tiene significado propio, como pueden ser las palabras reservadas del lenguaje, los identificadores, los operadores, etc.

El resultado de esta traducción es lo que se llama una **tira de tokens** que es la información que recibe el siguiente elemento del compilador.

Ejemplos de errores lexicográficos pueden ser palabras reservadas mal escritas, identificadores mal escritos, etc.

ENVIROMENT DIVISION
↑
N

11.3.2. Análisis sintáctico

El **analizador sintáctico** o **parser** toma la tira de tokens que recibe del scanner e investiga en ella los posibles errores sintácticos que aparezcan, estos errores suelen ser de formato de instrucciones, duplicidad de identificadores de distintas variables, etc.

Tanto en un analizador como en otro, todo error detectado es comunicado al programa por medio de un listado de errores, en el que se indican donde están situados y qué tipo de error es.

En ocasiones, se indican al programador determinados posibles errores que pueda haber sin que perjudiquen el resto del proceso de compilación. Estos mensajes se denominan **warning** o **advertencias.**

11.3.3. Tabla de símbolos

La **tabla de símbolos** es el elemento que almacena todos los datos referidos a variables y estructuras de datos, del programa que se está compilando. Estos datos suelen ser el tipo de la variable, dimensiones en los casos de tablas internas, direcciones de memoria, etc. También se denomina **diccionario.**

11.3.4. Generación de código

El **generador de código intermedio** traduce el resultado del análisis (en caso de ausencia de errores) a un código intermedio propio del compilador, para con él permitir la portabilidad del lenguaje (posibilidad de utilización en distintas computadoras).

Debido a que el lenguaje máquina es distinto entre una computadora y otra, se hace común todo el proceso de análisis, particularizándose a partir del código intermedio para cada familia de procesadores, con lo que además se abaratan los costes al aplicar un mismo lenguaje a una gran variedad de máquinas (Figura 11.5).

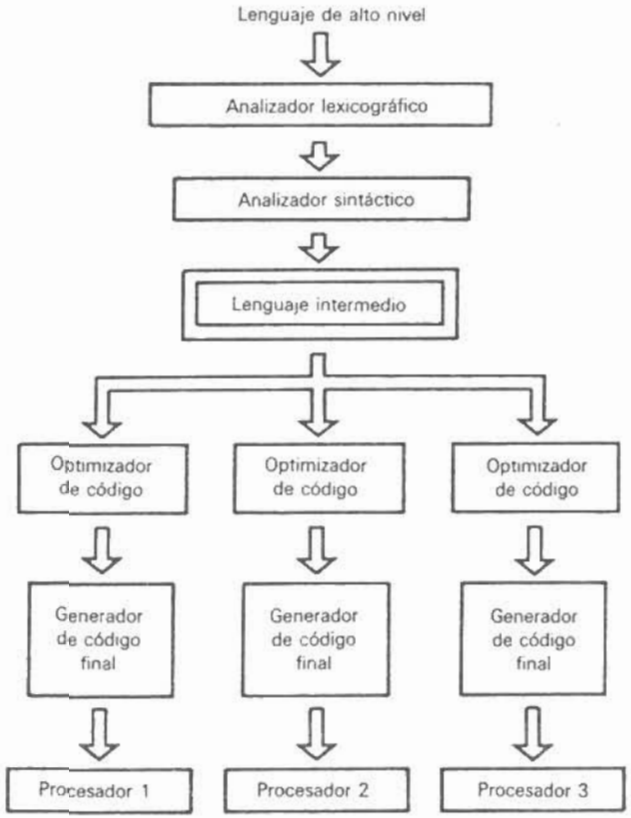


Figura 11.5. Portabilidad de un lenguaje de alto nivel.

La misión del **optimizador de código** consiste en tomar el código intermedio y optimizarlo, adaptándolo a las características de cada procesador.

Es conveniente tener en cuenta que el código intermedio está enfocado a que un programa en dicho código pueda ser, con algunas modificaciones, interpretado por cualquier procesador. Por tanto, podemos decir que el optimizador de código, además de su misión, realiza la de adaptar a las características propias de cada procesador.

La misión del **generador de código final** es la de traducir el código intermedio optimizado al código final, en el que quedará escrito el programa objeto.

11.4. GESTION DE LA MEMORIA

Al ejecutar la compilación de un programa, todas las variables y estructuras de datos que se utilicen deben quedar de una forma u otra con una determinada asignación o referencia a la memoria. Existen tres tipos de asignación que relacionamos a continuación.

- **Asignación estática**

Es un tipo sencillo, pero es necesario que en el momento de la compilación se conozcan todos los objetos que va a manejar el programa. Se basa en una gran reserva de memoria donde se van asignando los elementos del programa de forma contigua.

- **Asignación dinámica**

Se realiza en determinados lenguajes modernos donde se manejan estructura dinámicas que en la ejecución del programa pueden variar el número de sus elementos. En general queda asignada para cada elemento una dirección base a partir de la cual se va configurando la estructura de datos.

- **Asignación mixta**

Trata de aprovechar las ventajas de las dos anteriores realizando una asignación estática-dinámica.

11.5. ERRORES. TIPOS, DETECCION Y RECUPERACION

La misión del **módulo de tratamiento de errores** es la de facilitar la detección y, en algún caso, la recuperación de errores en las distintas fases de la compilación. Un compilador, cuando detecta un error, trata de buscar la sentencia y el lugar donde se ha producido así como de determinar la causa probable para poder ofrecérsela al programador en lo que se denomina un **mensaje de diagnóstico** de error que incluirá en el correspondiente listado de errores. En la mayoría de los casos, cuando se detecta un error, el compilador continúa realizando el diagnóstico de errores pero no genera el programa objeto ya que no serviría para nada.

Los tipos de errores que puede tener un programa son los siguientes:

- **Errores lexicográficos.** Son errores que se producen por la aparición de tokens no reconocibles, los detecta el scanner en el tiempo en que actúa el analizador lexicográfico. Los más frecuentes son: aparición de un carácter que no pertenece al lenguaje, palabras reservadas y obligatorias del lenguaje mal escritas (por ejemplo ENVIROMENT en lugar de ENVIRONMENT), identificador declarado de una forma y utilizado de otra, comentarios sin orlar con los correspondientes apóstrofos y más de un punto decimal en valores reales.

- **Errores sintácticos.** Son los que no cumplen las reglas de sintaxis del lenguaje, detectándolos el parser al no reconocer una tira completa de tokens como un formato válido de instrucción. Los errores más típicos de este tipo son: expresiones incompletas (A+B/), caracteres no permitidos en una determinada instrucción, equivocación de caracteres, palabras reservadas mal construidas y, en general, construcciones incorrectas de la estructura del programa.

- **Errores semánticos.** Se detectan en alguna fase de la compilación y en algún caso en la de ejecución y son aquellos que no interrumpen el proceso pero se detecta en ellos alguna incorrección (por ejemplo variables no declaradas) son los denominados **warning.**

- **Errores lógicos.** Son los debidos a la utilización de un algoritmo o fórmula incorrecta para el problema que se trata de resolver. Se detectan en la fase de pruebas de un programa por medio de la utilización de juegos de ensayo.

- **Errores de ejecución.** Son errores relacionados con desbordamientos, operaciones matemáticamente irresolubles, etc. Ejemplos de este tipo de errores son: división por 0, cálculo de la raíz cuadrada de un número negativo, desbordamientos en general, leer de un archivo no abierto, salir del rango de una tabla, bucles infinitos, etc. La mayoría de los lenguajes ofrecen algunas facilidades para la detección de dichos errores y sobre todo para prevenirlos.

```
ON SIZE ERROR
ERR=
EOF
```

11.6. INTERPRETES

Los programas traductores que realizan lo que podríamos llamar traducción simultánea, es decir, traducir y ejecutar simultáneamente se denominan como ya hemos comentado **intérpretes.** La ventaja frente a un compilador es que la ocupación en memoria es menor ya que suelen ser programas más reducidos en tamaño que un compilador. El gran inconveniente es el tiempo de ejecución que tarda un programa sujeto a este tipo de traducción (Figura 11.6).

En la actualidad existen intérpretes que incorporan una pequeña fase de análisis antes de la interpretación para evitar los problemas originados por la suspensión del programa en ejecución al aparecer el primero de los errores, con lo cual todo el proceso transcurrido se pierde.

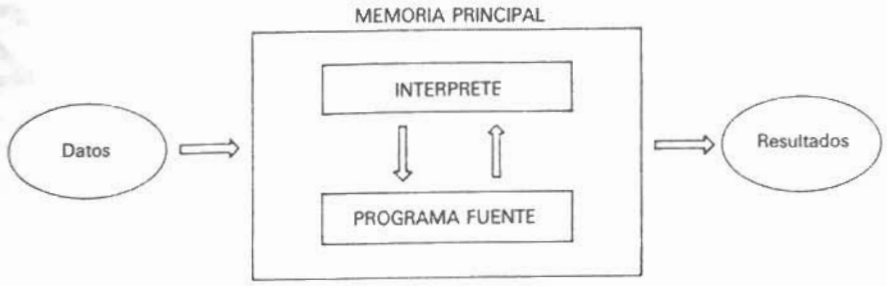


Figura 11.6. Intérprete.

Una buena aplicación de los intérpretes es la de depurar programas que una vez en funcionamiento correcto se compilan ejecutándose el producto de dicha compilación en el uso normal del programa.

La estructura de un intérprete es la reflejada en la Figura 11.7.

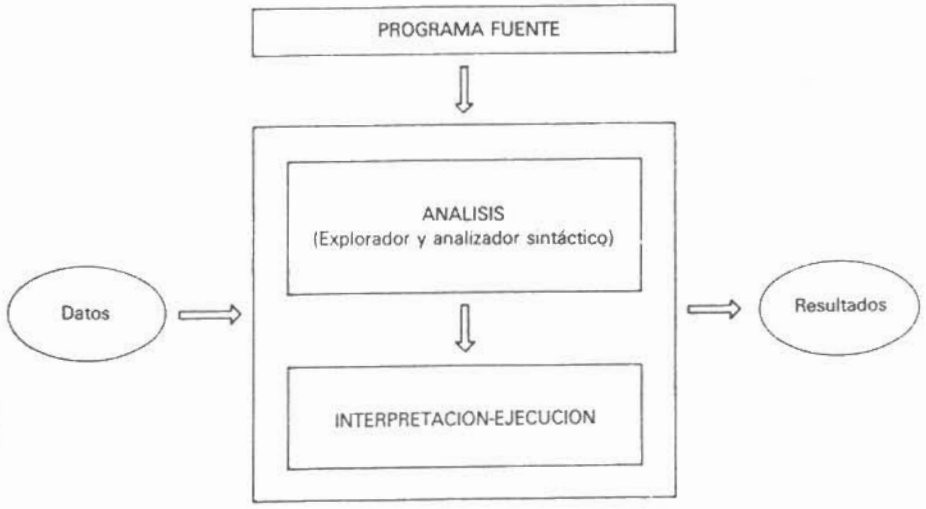


Figura 11.7. Estructura general de un intérprete.

11.7. LIBRERIAS

Los pequeños programas que realizan cálculos de tipo matemático o el control de determinadas operaciones sobre archivos son generalmente suministrados por el compilador en lo que se denomina **librería.** Estos programas ya están compilados y los incorpora el montador al programa que reclama los servicios de alguno de ellos.

En ocasiones las librerías se encuentran indexadas para permitir su ampliación en cualquier momento con nuevos módulos.

Con el uso de estas librerías se ha logrado dotar a los lenguajes de programación de alto nivel de un gran número de funciones que permiten al programador reducir notablemente sus programas.

11.8. DEPURADORES *(DEBUGGERS)*

Mientras se realiza el proceso de desarrollo de un programa, resulta muy interesante conocer cómo se ejecuta y qué ocurre en cada momento en nuestro programa. Para ello, algunos sistemas poseen unos programas depuradores cuya misión es permitir la ejecución paso a paso o por tramos del programa manteniendo el entorno que se va produciendo (valores de variables). El programador en cada parada de la ejecución de su programa puede comprobar e incluso modificar valores de las variables, con lo cual resulta muy interesante este tipo de ejecución para la detección de errores y comprobación del buen o mal funcionamiento del programa. Los puntos de parada en la ejecución son determinados por el propio programador mediante sentencias destinadas a tal fin.

11.9. EDITORES DE ENLACE O MONTADORES *(LINKERS)*

En general, el producto de la compilación suele ser un conjunto de módulos independientes denominados **segmentos** en lenguaje máquina que necesitan ser montados para formar el programa completo para el que han sido creados. La misión del editor de enlace o montador es unir los distintos módulos que configuran un programa asignando la memoria al conjunto e incorporando aquellas rutinas del propio lenguaje a las que se hacen referencia.

Algunos editores de enlace permiten el montaje de módulos escritos en distintos lenguajes y compilados con los correspondientes compiladores.

CUESTIONES

1. Explicar lo más detalladamente posible por qué razón es necesario en la actualidad someter todo programa a un proceso de traducción.
2. ¿En qué consiste la traducción de un programa?
3. ¿Cuáles son las diferencias de los tres tipos de traductores que se utilizan en la actualidad?
4. ¿Qué ventajas tiene un ensamblador cruzado frente al resto de programas ensambladores?
5. ¿Cuál es la ventaja de un ensamblador de dos pasadas en comparación con uno de una sola pasada?
6. ¿Cómo definiría un macroensamblador?
7. ¿En qué cuestiones esenciales se diferencian los compiladores de los intérpretes?
8. Dibujar un esquema que represente la estructura general de un compilador.
9. ¿En qué consiste el análisis lexicográfico del proceso de compilación de un programa?
10. ¿Qué misión realiza el analizador sintáctico de un compilador?
11. ¿Cuáles son los datos que refleja la tabla de símbolos que se genera en todo proceso de compilación de un programa?
12. ¿Cómo se consigue que un determinado lenguaje de programación sea portable, es decir, que se pueda implantar en distintas computadoras?
13. Enumerar y comentar los tipos de errores que pueden presentarse en un programa y, en su caso, dónde suelen detectarse.
14. Dibujar la estructura general de un intérprete.
15. ¿En qué consisten las librerías de un traductor?
16. ¿Para qué se utiliza un depurador en el diseño de un programa?
17. ¿Qué funciones realiza un editor de enlace o montador?
18. Realizar una reflexión sobre los procesos de traducción que en su experiencia haya realizado y relaciónelos con todo lo visto en este capítulo.

CAPITULO 12

Sistema Operativo DOS

12.1. INTRODUCCION

El sistema operativo DOS comúnmente denominado MS-DOS *(MicroSoft-Disk Operating System)* es un sistema monousuario y monotarea creado para la gama de computadoras personales que en 1981 lanzó IBM (International Business Machines). Posteriormente fue instalado en una gran gama de marcas y modelos de computadoras compatibles en cierta medida con la de IBM (Figura 12.1).

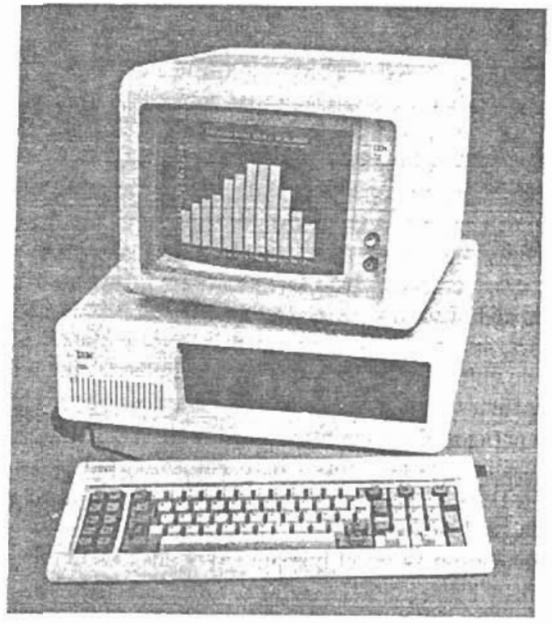


Figura 12.1. Computadora personal IBM. (Cortesía de IBM.)

El sistema operativo DOS es el interfaz que permite al usuario de una computadora personal acceder al hardware, ejecutar programas y sacar rendimiento a un equipo capaz de realizar una gran variedad de trabajos. Proporciona el medio para indicar al hardware qué programa o mandato se quiere ejecutar, dónde se encuentra el programa o mandato, etc.

Para realizar estas misiones, el DOS trabaja en dos niveles: el primero para establecer el control necesario sobre el hardware, y el segundo para ofrecer al usuario la posibilidad de dar órdenes para que a través de ellas pueda realizar los distintos trabajos.

12.2. HISTORIA

Con la aparición de la computadora personal de International Business Machines (IBM) en 1981, se desarrolló el sistema operativo Disk Operating System (DOS), siendo sus creadores las empresas Microsoft Corporation e IBM. En aquel año se utilizaban dos versiones similares, una de cada empresa denominadas MS-DOS 1.0 y PC-DOS 1.0, que permitían a los usuarios del PC (Personal Computer) una serie de órdenes básicas para su funcionamiento. La característica principal de estas primeras versiones era la utilización de un subsistema de archivos que ha permitido compatibilizar éstos en las sucesivas versiones así como en otros sistemas operativos como es el OS/2. Otra característica de esta primera versión era la utilización de disquetes de 5.25 pulgadas de una sola cara con una capacidad de 160 K.

En el año 1982 la empresa Microsoft mejoró el sistema operativo permitiendo la utilización de disquetes de doble cara con una capacidad de 360 K. Esta versión fue la MS-DOS 1.25.

En 1983, las empresas Microsoft e IBM mejoraron notablemente el sistema operativo dotándole de una buena gestión de archivos (aparición de subdirectorios, etc.), ampliando el conjunto de órdenes y mejorando algunas de las existentes. Las versiones surgidas en aquel año fueron las 2.0, 2.01, 2.10, 2.11 y 2.25.

En el año 1984 y con la aparición del modelo PC-AT, se desarrollaron dos nuevas versiones. La DOS 3.0, que incorpora la utilización de discos de 5.25 pulgadas de alta densidad (1.4 Mbytes) y discos fijos también de gran capacidad y la versión DOS 3.1, que permite la conexión de computadoras personales bajo DOS en redes de área local y su utilización como servidores en entornos multiusuarios.

La versión DOS 3.2, aparecida en 1986, además de la mejora en cuanto a órdenes, introdujo la posibilidad de utilización de disquetes de 3.5 pulgadas tanto en baja (720 K) como en alta densidad (1.2 Mbytes).

En 1987, con la aparición de otras arquitecturas de computadoras personales (80286, 80386, etc.) compatibles con las anteriores, pero con importantes mejoras tecnológicas, se comercializó la versión DOS 3.3, capaz de ser implantado en la nueva serie de computadoras personales PS/2 de IBM.

Actualmente se encuentran ya en el mercado versiones DOS 4.X, de características superiores a las anteriores y están anunciadas las características de las futuras versiones DOS 5.X.

La Figura 12.2 muestra esquemáticamente la evolución del sistema operativo DOS.

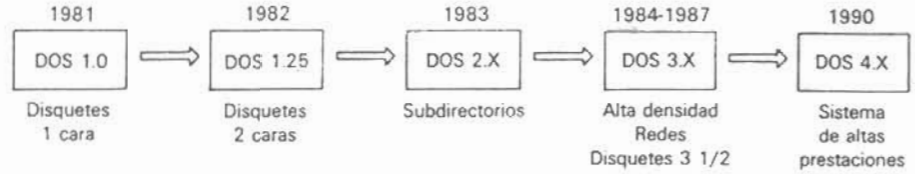


Figura 12.2. Evolución del sistema operativo DOS.

12.3. ESTRUCTURA DEL SISTEMA OPERATIVO DOS

La estructura del sistema operativo DOS es la representada en la Figura 12.3.

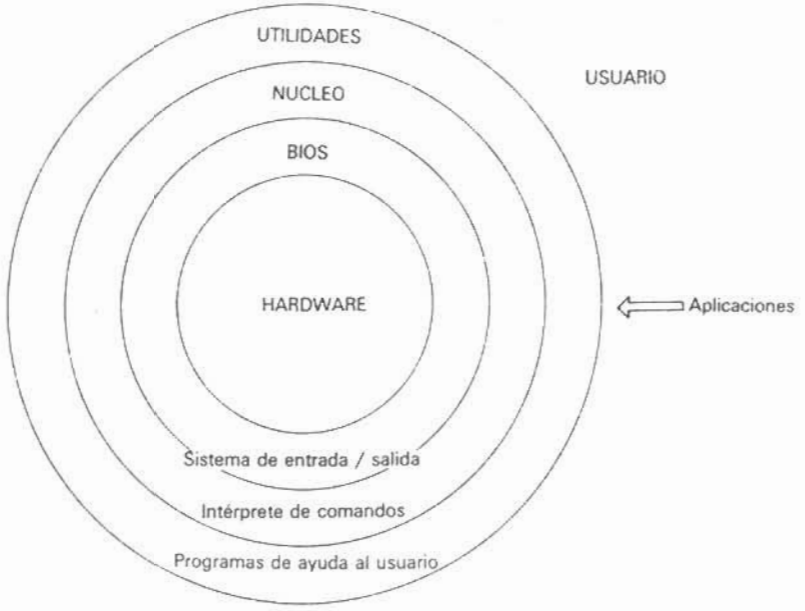


Figura 12.3. Estructura del sistema operativo DOS.

Como puede verse en primer término, rodeando al hardware se encuentra una parte software denominada BIOS *(Basic Input Output System)* residente en memoria de sólo lectura *(Read Only Memory-ROM)*, cuyas misiones son las siguientes:

- Realizar un test de todo el equipo en cada proceso de arranque donde se examinan todos los elementos conectados y en qué estado se encuentran.
- Hacer de interfaz entre el software de los niveles superiores y el hardware a través de una serie de rutinas, cada una de ellas con una función específica.

El siguiente nivel corresponde al núcleo del sistema operativo, que permanece constantemente en memoria desde que se enciende el equipo. Está compuesto por el programa intérprete de comandos (COMMAND.COM) que lleva consigo la carga de una serie de comandos residentes permanentemente en memoria y dos archivos de los denominados ocultos, pues no aparecen en el directorio del disco que los contiene pero que se encuentran presentes. Estos archivos tienen rutinas que permiten ampliar y actualizar (evolución de las versiones DOS) las rutinas de la ROM-BIOS.

12.4. CONCEPTOS BASICOS

▪ Gestión del procesador

Al tratarse de un sistema monousuario y monotarea, el procesador en cada momento está dedicado en exclusividad a la ejecución de un proceso. Por tanto, la planificación del procesador es simple, se dedicará al único proceso activo que puede existir en un momento dado.

▪ Direccionamiento de memoria principal

El direccionamiento de la memoria en el DOS se realiza en modo real, es decir, se accede directamente a posiciones físicas de la memoria. Se utiliza un esquema de direccionamiento basado en un segmento y un desplazamiento. Las direcciones de un segmento son de 16 bits, por tanto se pueden referenciar 64 K, pero se pueden desplazar a la izquierda hasta 4 bits sin perder los más significativos, con lo cual dispondremos de 1.024 K o 1 Mbyte. Como los cuatro últimos bits son cero, las posiciones que se pueden direccionar van de 16 en 16 direcciones.

▪ Organización de archivos en disco

Un disco formateado y preparado para ser utilizado bajo DOS posee dos elementos fundamentales, la tabla de asignación de archivos *(File Allocation Table-FAT)*, donde aparecen las indicaciones de sectores del disco ocupados, libres y defectuosos y como segundo elemento el directorio donde aparecen informaciones referentes a los archivos existentes con sus nombres, direcciones y una serie de datos complementarios.

La asignación del espacio a ocupar por un archivo en disco se realiza de forma contigua mientras exista espacio, pero en el momento que al ampliar un archivo no existan sectores contiguos, el sistema le asigna sectores libres de forma no contigua. De esta forma se evitan problemas de fragmentación interna pero se complica la acción de asignar espacio a los archivos.

En un disco se pueden crear subdirectorios, en cuyo caso, físicamente se genera una entrada en el directorio raiz que indica dónde se encuentra el archivo que va a contener la información de archivos del subdirectorio.

El sistema operativo DOS, al igual que la mayoría de sistemas actuales, cuenta con un conjunto de órdenes para mantener una estructura multinivel de directorios en forma de árbol con la cual se realiza una gestión y organización de archivos muy eficiente. La Figura 12.4 muestra un ejemplo de estructura de directorios.

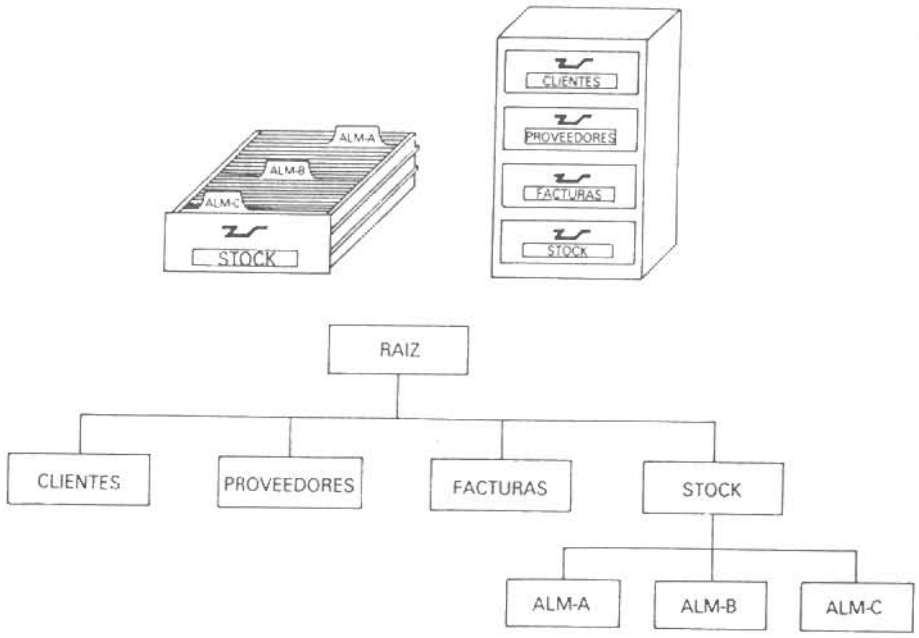


Figura 12.4. Estructura de directorios.

12.5. EL INTERPRETE DE COMANDOS

El intérprete de comandos del DOS es un programa denominado COMMAND.COM que realiza funciones de interfaz entre el usuario y la computadora. Se encarga de mantener el indicador del DOS (prompt) y está en todo momento en espera de recibir una orden para iniciar el proceso que lleve a su resolución, para volver nuevamente al estado inicial de espera.

Cuando estamos bajo el control del intérprete de comandos, podemos ejecutar órdenes (.COM o residentes) o programas ejecutables (.EXE) de dos formas:

- **Modo interactivo.** Con el cual el comando o el programa se ejecuta en ese momento.
  Ejemplo: C>Date
  La fecha actual es Mie 25-12-1991
  Introduzca la nueva fecha (dd-mm-aa):

- **Proceso por lotes.** Se reúnen un conjunto de comandos y programas que deben ejecutarse uno tras otro en un archivo (.BAT) y el procesador cuando recibe la orden así lo hace. Este tipo de procesos es muy flexible ya que admite condicionamientos para algunos de sus elementos, repeticiones y una gran variedad de combinaciones posibles de igual forma que estando en ejecución un proceso por lotes se puede interrumpir en cualquier momento.

Existe un archivo por lotes característico en el DOS cuya misión es la de inicializar el entorno del equipo e incluir todo aquello que se desea ejecutar al arrancar el sistema. Se trata del archivo AUTOEXEC.BAT que se ejecuta, siempre que exista, automáticamente al iniciar una sesión (al conectar el equipo).

12.6. HERRAMIENTAS DEL DOS

El sistema operativo ofrece en todas sus versiones una serie de herramientas para facilitar al usuario una serie de operaciones básicas.

Entre las más utilizadas podemos citar:

- **Editor de texto** *(EDLIN o EDIT).* Permite al usuario crear archivos de texto, escribir archivos batch, escribir código fuente en cualquier lenguaje, etc.
- **Ordenación de archivos** *(SORT).* Se trata de un programa cuya misión es la de ordenar archivos.
- **Realización de copias de seguridad** *(BACKUP y RESTORE).* Permiten de forma automática la realización de copias de seguridad del disco fijo.
- **Depurador** *(DEBUG).* Permite la ejecución de un programa paso a paso estableciéndose puntos de parada donde puede examinarse el entorno.
- **Intérprete de lenguaje BASIC** *(BASICA, BASIC o GWBASIC).* Se trata de un intérprete muy actualizado del lenguaje BASIC.
- **Configuradores lógicos de teclado** *(KEYB).* Son programas que adaptan el teclado físico común a todos los países, al lenguaje y alfabeto propio de cada país.
- **Otros.** Existe además una serie de programas para realizar operaciones que en ocasiones son una buena ayuda al usuario.

12.7. SOFTWARE ESTANDAR PARA COMPUTADORAS PERSONALES

Actualmente existen programas y paquetes de software dirigidos a computadoras personales bajo el sistema DOS que permiten todo tipo de trabajos. Entre ellos citaremos los siguientes:

- **Compiladores de lenguajes de alto nivel.** Permiten el desarrollo de programas de usuario en lenguajes de alto nivel como son el COBOL, FORTRAN, Pascal, C, Logo, Ada, Pilot, LISP, RPG, BASIC y un sinfín de lenguajes que cada día se incrementa.
- **Procesadores de texto.** Son programas que convierten la computadora en una máquina de escribir evolucionada, donde se pueden escribir todo tipo de textos (documentos, cartas, programas fuente, etc.), existiendo una gran variedad de tipos de letras, correctores ortográficos y otros elementos de una gran ayuda.

- **Hojas de cálculo.** Son programas que permiten realizar una serie de cálculos sobre una planilla compuesta de celdas donde pueden ser almacenados tres tipos de datos: números, cadenas de caracteres o expresiones cuyos operandos son casillas de la propia planilla.
- **Sistemas de gestión de bases de datos.** Son programas que realizan toda la gestión de una base de datos diseñada por el propio usuario. Permiten el diseño de la misma, la carga de datos en los archivos, su actualización y la extracción de todo tipo de consultas e informes sobre la información contenida.
- **Paquetes de software integrado.** Son conjuntos de programas para la realización de la gestión completa en pequeñas empresas. Poseen en general un procesador de texto, una hoja de cálculo, un gestor de bases de datos y ayudas para la edición de gráficos y el establecimiento de comunicaciones con la gran ventaja de poder pasar información de un elemento a otro dentro del propio paquete.
- **Enseñanza asistida por computadora.** Existen en el mercado programas que permiten con facilidad la realización de aplicaciones sobre computadoras personales encaminadas a la enseñanza, es decir, a convertir la computadora en un profesor que enseña y pregunta generalmente a través de figuras con animación.
- **Programas de ayuda al DOS.** También existen una serie de programas que automatizan la gestión del propio sistema operativo y permiten la realización de órdenes y ejecución de programas de tal forma que el usuario apenas tiene que recordar nada. Entre otras funciones, estos programas suelen tener la posibilidad de ofrecer información completa de todo el sistema.
- **Diseño asistido.** De igual forma existen programas para facilitar el trabajo a todo tipo de diseñadores, delineantes o cualquier persona que necesite realizar diseños de cualquier tipo.
- **Otros.** Existen, además de los anteriores, miles de programas que permiten la confección de gráficos, comunicaciones con otros equipos, realización de contabilidades, etc.

CUESTIONES

1. Explicar brevemente qué tipo de sistema operativo es el DOS y para qué tipo de computadoras fue diseñado.
2. Realizar un esquema de la evolución que ha ido teniendo el sistema operativo DOS.
3. Dibujar esquemáticamente la estructura del sistema operativo DOS.
4. ¿Qué es y para qué sirve el software denominado BIOS en el entorno del sistema operativo DOS?
5. ¿Qué elementos componen el núcleo del sistema DOS?
6. ¿Cómo se gestiona el procesador en sistemas operativos monotarea?
7. ¿Cómo se realiza la asignación de espacio en disco para los archivos bajo control del sistema operativo DOS?
8. ¿Qué funciones realiza el intérprete de comandos del sistema operativo DOS?
9. Indicar algunas de las herramientas que el sistema operativo DOS ofrece al usuario para la realización de operaciones básicas.
10. Indicar algunos programas de software estándar que pueden utilizarse en las computadoras personales bajo DOS.
11. ¿Qué es un paquete de software integrado? ¿Conoce alguno de ellos? Si es afirmativo indicar algunas de sus características.
12. Si ha utilizado alguna vez el sistema operativo DOS, ¿qué opinión particular le merece?

CAPITULO 13

Sistema Operativo UNIX

13.1. INTRODUCCION

El objetivo de este capítulo es el de realizar una descripción del sistema operativo UNIX de tal forma que sea suficiente para conocer sus características.

Se trata de un sistema operativo de los más utilizados y con más futuro debido a que son muchos organismos oficiales y particulares los que defienden su utilización, así como muchas firmas de fabricación y comercialización de computadoras que lo incorporan en sus productos. Podemos citar como ejemplo el de la Comunidad Económica Europea, que impone el sistema operativo UNIX en todas las aplicaciones que se desarrollan bajo sus auspicios.

Comenzaremos la descripción del sistema operativo UNIX haciendo una breve reseña histórica para llegar a su situación actual e indicar los posibles motivos de su fuerte expansión. Seguiremos dando unas pequeñas indicaciones de su estructura interna para pasar a hablar del Intérprete de Comandos *(Shell)*, finalizando con una descripción de las herramientas de ayuda en el desarrollo de software.

13.2. HISTORIA

La historia del sistema operativo UNIX es la que se describe a continuación, en consonancia con la evolución representada en la Figura 13.1.

- **1965:** Las empresas Bell Telephone Laboratories y General Electric Company intervienen en el proyecto MAC del Massachusets Institute Tecnology (MIT) para desarrollar un nuevo sistema operativo denominado MULTICS, cuyo objetivo fue el ofrecer un sistema multiusuario (acceso simultáneo de gran número de usuarios) de gran potencia de proceso, gran capacidad de almacenamiento y con grandes facilidades para compartir datos entre procesos.
- **1969:** Vistos los resultados poco satisfactorios del MULTICS, la Bell Telephone Laboratories se retira del proyecto y desarrolla un sistema de tiempo compartido con paginación por demanda para uso interno de la empresa sobre la computadora PDP-7 de Digital, siendo los artífices del mismo un equipo encabezado por Ken Thompson y Dennis Ritchie. Este sistema operativo constituyó la primera versión del UNIX, que sólo permitía la explotación en monoprogramación.

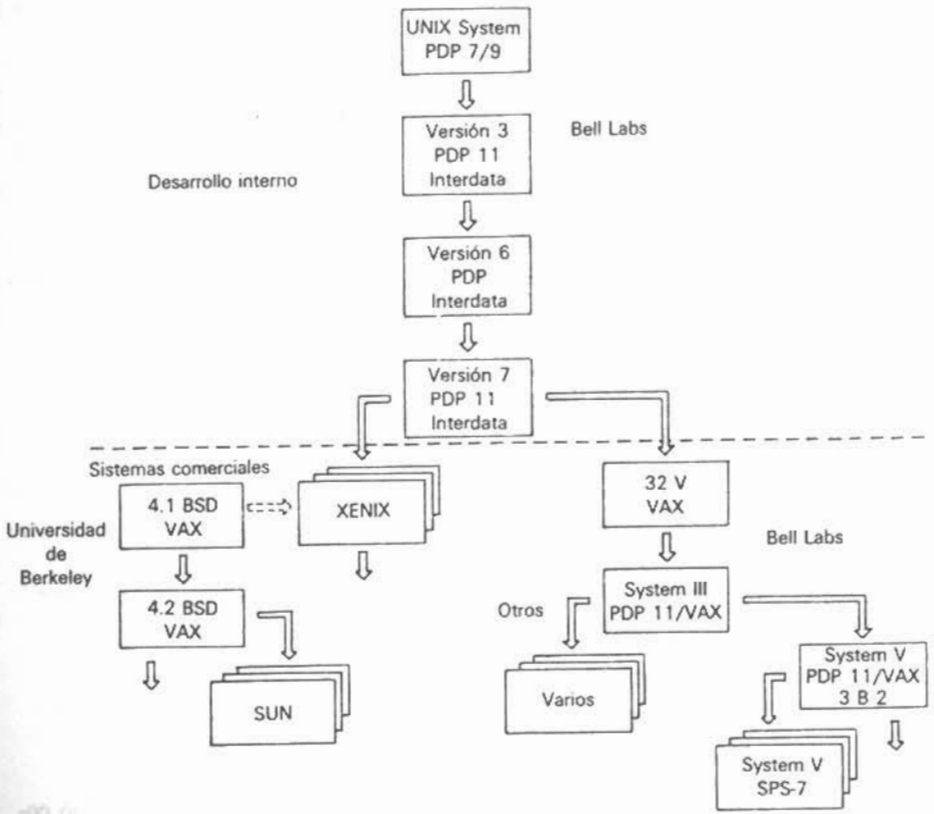


Figura 13.1. Evolución del sistema operativo UNIX.

- **1971:** El resultado del sistema anterior tuvo tanto éxito que la compañía puso a disposición de Thompson y Ritchie una computadora más potente, que fue la PDP-11 de Digital. En ella, Thompson desarrolló el lenguaje de programación B inspirándose en el BCPL y en el FORTRAN, y a continuación Ritchie creó el lenguaje C, con el que consiguió la generación de código máquina, descripción de tipos de datos y de estructuras de datos.
- **1973:** Se reescribe en C la versión de UNIX desarrollada en ensamblador y que prácticamente es la que se ha mantenido hasta hoy. Aparece una versión de UNIX conocida por Programmer's Workbench (PWB).
- **1974:** Se introduce el sistema operativo UNIX en las Universidades norteamericánas con fines educativos.
- **1977:** Se construye la primera versión comercial del UNIX, conocida como versión 6, implantándose por primera vez en una computadora distinta de la PDP, que fue la INTERDATA 8/32.

- **1979:** Aparece la versión 7 de UNIX para PDP y una versión para la computadora VAX de Digital (32 bits), conocida como 32V.
- **1981:** Nace la primera versión de UNIX para computadoras personales con el nombre de XENIX.
- **1982:** Para la distribución externa, los laboratorios Bell desarrollan el UNIX System III, que no es más que el original con algunas pequeñas variantes. Por otra parte, la Universidad de Berkeley desarrolla una variante del UNIX 32V para computadoras VAX con mejoras en cuanto a comandos y gestión de la memoria virtual paginada, denominada 4.1 BSD.
- **1983:** La empresa AT&T anuncia una nueva versión denominada UNIX System V, que es el sistema actual y que presenta importantes mejoras de rendimiento, comunicaciones, etc.
- **1984:** La Universidad de Berkeley presenta la versión 4.2 BSD para computadoras VAX, que también se aplica en estaciones de trabajo SUN 2/3 de SUN MICROSYSTEMS.

En la actualidad se utilizan fundamentalmente dos versiones del sistema operativo:

- UNIX System V.
- XENIX System V.

En la mayoría de los casos, cada fabricante tiene su propia versión de UNIX; entre ellas citaremos:

• Digital Equipment Corporation	ULTRIX.
• IBM	AIX.
• Data General	DG/UX.
• National Semiconductor	GENIX.
• Gould Concept 32/6750	UTX.
• Hewlett Packard	HP-UX.

13.3. ESTRUCTURA DEL SISTEMA OPERATIVO UNIX

El sistema operativo UNIX es un sistema de tiempo compartido y, por tanto, multiusuario, en el que existe portabilidad para la implantación en distintas computadoras.

Está formado por una serie de elementos que pueden representarse en forma de capas concéntricas donde, en primer lugar, alrededor del hardware de la máquina se encuentra el núcleo *(kernel)*, que interactúa, directamente con el hardware, aislando a éste de los usuarios, además de adaptar el resto del sistema operativo a la máquina debido a la portabilidad que existe en el mismo (Figura 13.2).

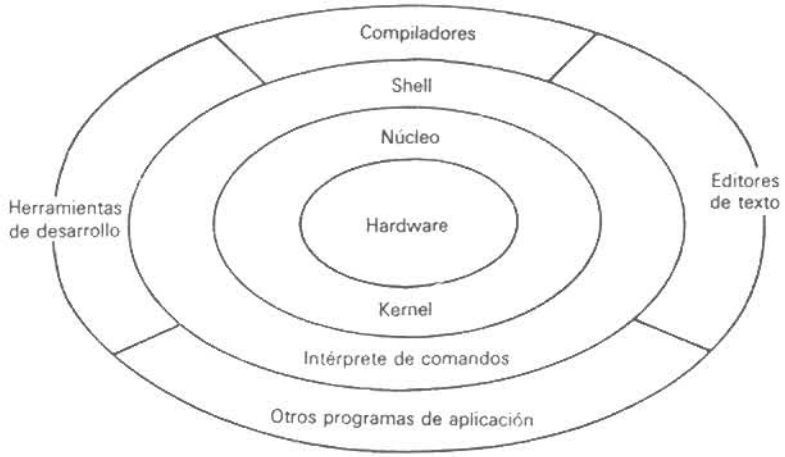


Figura 13.2. Estructura del sistema operativo UNIX.

En una segunda capa se encuentran los comandos, que no son otra cosa que el interface entre los programas de aplicación y el núcleo del sistema operativo.

La última de las capas contiene los programas de aplicación.

El sistema operativo UNIX se compone de bloques funcionales que se encuentran representados en la Figura 13.3 y serán comentados en los apartados que siguen.

13.4. VENTAJAS E INCONVENIENTES

Entre las ventajas que permiten al sistema operativo UNIX gozar de la popularidad que en la actualidad tiene, se pueden citar las siguientes:

- Resulta de fácil lectura por estar escrito en alto nivel (C).
- Es un sistema jerárquico de procesos y archivos.
- Posee un interface con periféricos consistente y uniforme.
- Es un sistema multiusuario y multiproceso.
- Adapta los programas *(fuente)* a cualquier máquina con sistema operativo UNIX.
- No está ligado a una marca comercial fabricante de computadoras.
- Permite la combinación de utilidades para producir otras nuevas.

Por otra parte, los inconvenientes que presenta este sistema son:

- Comandos poco claros y con demasiadas opciones.
- Escasa protección entre usuarios.
- Sistema de archivos lento.

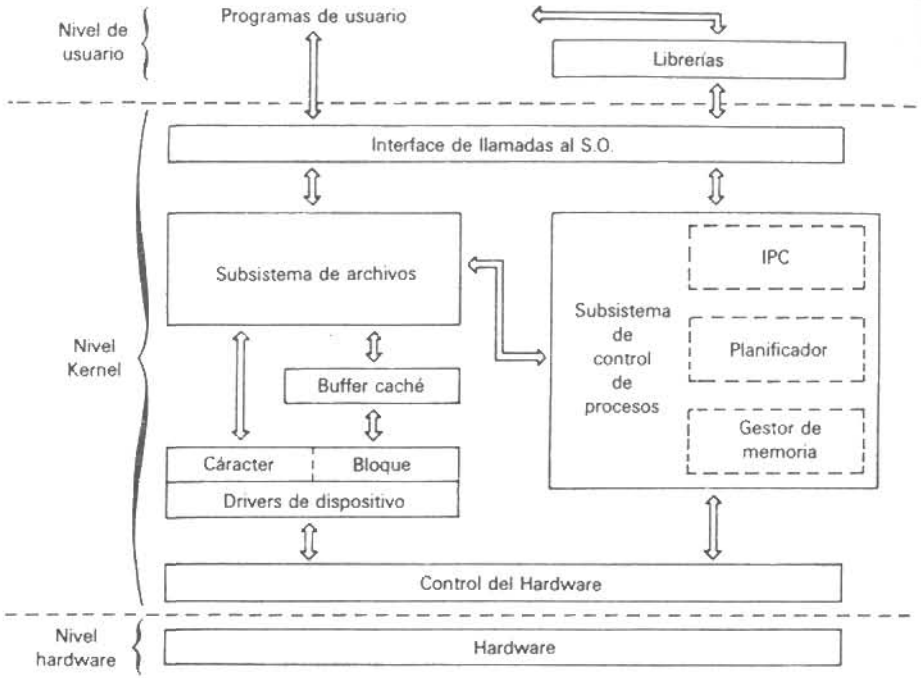


Figura 13.3. Bloques funcionales del sistema operativo UNIX.

13.5. CONCEPTOS BASICOS

Para abordar un pequeño recorrido a través del sistema operativo UNIX, es necesario conocer ciertos conceptos y terminología común a todo el sistema.

La comunicación entre la computadora y el terminal teclado-pantalla que utilizará el usuario para trabajar bajo el control del sistema operativo es siempre Full-Duplex y podrán ser tecleados todos los caracteres que se deseen, más los caracteres de control que tienen significado especial (Tabla 13.1).

Tabla 13.1

Carácter de control	Significado
RETURN	Fin de E/S
Ctrl-m	Fin de E/S
Ctrl-d	Fin de la sesión
Ctrl-g	Acciona la campana
Ctrl-h	Retroceso (Backspace)
Ctrl-i	Tabulador
DELETE	Borra un carácter
BREAK	Produce una interrupción
Ctrl-c	Produce una interrupción

13.5.1. Sesión UNIX

El trabajo de un usuario en el sistema operativo UNIX se organiza por sesiones. Una sesión UNIX comprende todo el trabajo realizado por el usuario en la computadora, desde que se identifica hasta que se despide.

Vamos a exponer brevemente cuáles deben ser las acciones que ha de llevar a cabo el usuario para dar comienzo a una sesión, operar a lo largo de ella y finalizarla correctamente.

La Figura 13.4 muestra un ejemplo de inicio y terminación de una sesión de usuario. En ella puede observarse que el primer paso es el de identificación del usuario, en el que además el sistema pide la correspondiente palabra o clave de usuario para proteger el acceso de otros usuarios. Una vez reconocida la clave por el sistema, dará comienzo la sesión.

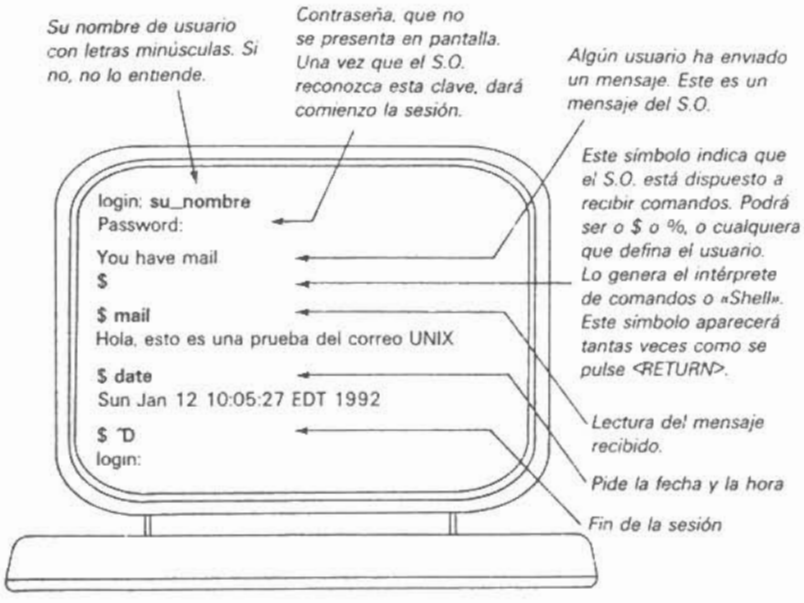


Figura 13.4. Inicio y fin de una sesión UNIX.

Aparece en el terminal el carácter que indica espera de mandato o comando, que será interpretado por el intérprete de comandos o Shell y que se denomina prompt, representándose en general por los caracteres $ o %. Cuando se termina el trabajo que se pretendía hacer, basta con despedirse de la sesión por medio del carácter de control Ctrl-d (^D).

13.5.2. Estructura de la línea de comandos

Una vez iniciada una sesión UNIX y estando presente el prompt $, el intérprete de comandos Shell está preparado para recibir un comando, cuya estructura es la siguiente (Figura 13.5):

$ nombre [calificador] [argumentos] [terminador]

nombre:	Nombre del comando.
calificador:	Posibles variaciones de actuación del comando.
argumentos:	Nombre del elemento (archivo, directorio...) sobre el que se quiere aplicar el comando.
terminador:	Delimitador que sirve para separar comandos (;).

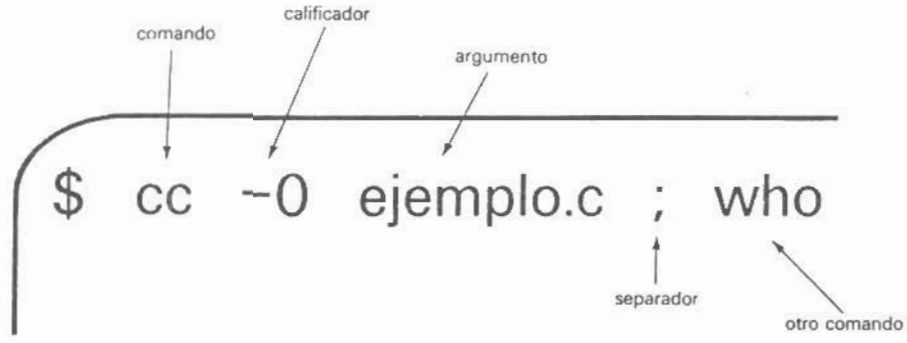


Figura 13.5. Estructura de la línea de comandos.

13.5.3. Archivos y directorios

La estructura de archivos y directorios es jerárquica y está constituida de forma arborescente, donde los nodos son directorios y las hojas son archivos normales. El árbol tiene un directorio raíz único root que se identifica por / (Figura 13.6).

El sistema de archivos UNIX gestiona varios tipos de archivos:

- **Archivos normales o regulares:** Son los archivos de usuario que contienen programas, textos, etc.
- **Directorios:** Son particiones lógicas que a su vez son archivos que contienen la información necesaria para poder encontrar un archivo en el disco.
- **Archivos especiales:** Se utilizan para designar periféricos de entrada y salida.
  - *Pipes:* Archivos que permiten la transferencia de datos entre procesos.
  - *Dispositivos organizados por bloques.*
  - *Dispositivos organizados por caracteres.*

En el sistema de archivos, cada archivo tiene asociado un conjunto de permisos que determinan qué puede hacerse con él y quiénes tienen acceso.

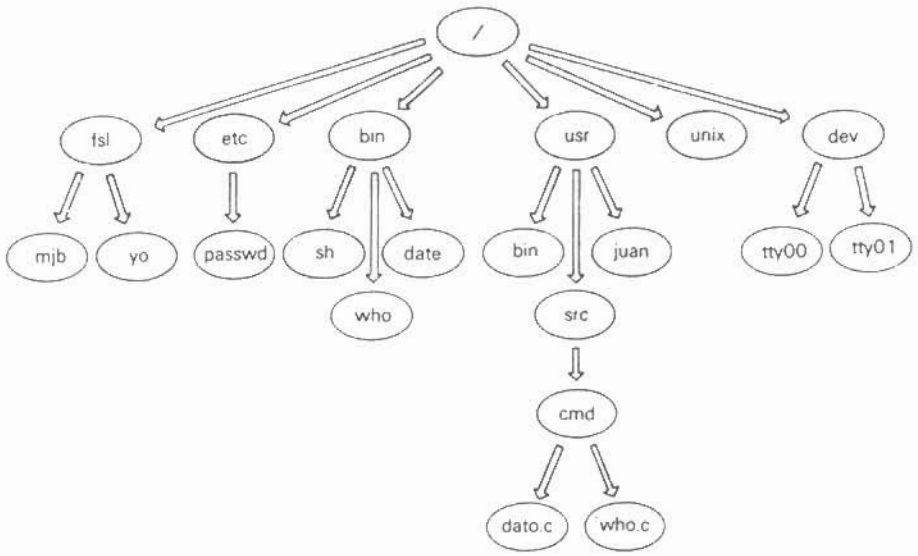


Figura 13.6. Estructura del sistema de archivos.

13.5.4. Control de procesos

El núcleo del sistema operativo UNIX conoce la existencia de un proceso a través de su bloque de control del proceso, donde se describe el proceso y su entorno, constituyendo un contexto consistente en:

- **Espacio de direccionamiento y entorno de ejecución:** Variables que utiliza el proceso.
- **Contenido de los registros hardware:** Contador de programa, registro de estado del procesador, puntero de la pila y registros de propósito general.
- **Contenido de las estructuras del núcleo relacionadas con el proceso:** Tablas de proceso, áreas, regiones, etc.

Si congelamos el estado del procesador y del proceso que está en ejecución en un determinado momento, obtendríamos lo que se conoce como imagen estática del programa. En caso de producirse una interrupción o cambio de proceso, se almacena la imagen del que está en ejecución en ese mismo instante.

Cada proceso se reconoce dentro del sistema por un número que lo identifica unívocamente y que se conoce como Identificador del Proceso o PID.

Todos los procesos, excepto el proceso 0, son creados por otro proceso; es decir, el sistema de creación y gestión de procesos en el sistema operativo UNIX es jerárquico.

El proceso que se genera con el PID 0 es un proceso especial creado en el momento de arrancar el sistema. A continuación se genera un proceso init que será el antecesor de todos los procesos que se generen en el sistema. A partir de aquí se generan tantos procesos como terminales existan.

Los procesos que atienden a los terminales crearán otros con el fin de identificar y controlar el acceso de los usuarios al sistema, los cuales, una vez que inicien la sesión, ejecutarán un proceso intérprete de comandos Shell que será el que genere el resto de programas solicitados por el usuario.

13.5.5. Gestión de la memoria

La gestión de la memoria en el sistema operativo UNIX se basa en intercambio *(Swapping)* y paginación. La paginación de la memoria se lleva a cabo si el hardware de la computadora la soporta. La política de carga y descarga de un proceso en la memoria depende del tiempo que lleve en la misma, de su actividad y del tamaño.

13.6. EL SHELL

Se conoce con el nombre de Shell al programa que interpreta las peticiones del usuario para ejecutar comandos, utilidades o programas; es decir, es el intérprete de comandos del sistema operativo UNIX y es el interface entre el usuario y el sistema.

El Shell es un programa del sistema operativo, pero no forma parte del núcleo del mismo. Se ejecuta cada vez que un usuario se identifica ante el sistema y comienza una sesión.

Es también un lenguaje de programación que soporta todas las estructuras propias de los lenguajes modernos. Además permite la utilización de todas las primitivas del sistema operativo de control de procesos, interrupciones y utilidades para diseñar programas de comandos por el usuario.

Existen varios tipos de Shell con diferentes características:

- **Bourne Shell:** Es el intérprete de comandos básico.
- **C-Shell:** Es el intérprete de comandos creado en Berkeley para el sistema operativo BSD y para el XENIX, un poco más completo que el anterior. Su programación es prácticamente lenguaje C.
- **Korn Shell:** Se basa en los dos anteriores, siendo compatible con el Bourne en un 95 por 100. Añade posibilidades de programación avanzada, facilidades aritméticas y mayor rapidez de ejecución.

13.7. HERRAMIENTAS DE DESARROLLO SOFTWARE

El sistema operativo UNIX proporciona al usuario una serie de herramientas o programas de utilidad para el desarrollo del software. Estas herramientas son:

- **Editor de pantalla (VI):** Tiene como misión principal la creación de archivos fuente.

- **CC:** Es el compilador del lenguaje C.
- **LINT:** Es un analizador sintáctico de lenguaje C más estricto que el compilador.
- **AR:** Es un programa para mantener librerías o bibliotecas de programas objeto o cualquier otro tipo de archivos.
- **MAKE:** Es un programa de utilidad de UNIX que acepta especificaciones de las dependencias existentes entre módulos de un programa y establece mecanismos para mantener las versiones del programa, trasladando al resto de módulos los cambios que se realicen en uno de ellos.
- **SCCS** *(Sistema de Control de Código Fuente):* Son un conjunto de utilidades para el desarrollo de aplicaciones con muchos módulos que se desarrollan en distintos equipos.
- **AWK:** Es el intérprete de un lenguaje de programación cuyo fin es tabular, dar formato y preprocesar archivos de datos.
- **LEX:** Es un programa de utilidad que sirve para crear un analizador léxico que basa su análisis en ciertas reglas de lenguaje definidas por el usuario.
- **YACC:** Es un analizador semántico cuya misión es transformar un archivo con instrucciones de un lenguaje especificado, que cumpla las reglas semánticas dadas, en un módulo compilable en C.

13.8. ADMINISTRACION DEL SISTEMA

En computadoras que funcionan bajo el sistema operativo UNIX, existe un usuario que se distingue de los demás por ser el encargado de realizar la administración del sistema. Las funciones propias del administrador del sistema son:

- Actualización y mantenimiento del sistema:
  - Mantenimiento del sistema de archivos.
  - Determinación de altas y bajas de usuarios.
  - Control de periféricos.
- Realización periódica de copias de seguridad *(Backups).*
- Suministro de soporte técnico al resto de usuarios.
- Gestión de los recursos de la computadora.
- Etcétera.

En el sistema operativo UNIX existe un directorio de uso exclusivo del administrador del sistema donde se encuentra una serie de comandos para la realización de dichas funciones, que no pueden ser utilizadas por el resto de usuarios.

El administrador del sistema, denominado más comúnmente **superusuario,** tiene asociada una cuenta que se identifica en el terminal por un símbolo diferente al del resto de usuarios, normalmente # en lugar de $.

En general, el sistema operativo UNIX puede tener además una serie de usuarios que gocen de cierta libertad para la gestión del sistema; éstos son los denominados **usuarios especiales.**

El superusuario es quien organiza y designa el arranque del sistema editando un archivo de comandos de uso exclusivo, en el que se ordenan las siguientes acciones:

- Comprobar si el sistema de archivos es correcto, para que si encuentra archivos erróneos proceda a su restauración.
- Limpiar el registro de usuarios activos.
- Montar el sistema de archivos.
- Limpiar los directorios temporales.
- Arrancar los procesos iniciales (update y cron).
- Presentar la fecha en la consola maestra.

De igual forma, el superusuario establece el proceso de apagado del sistema cuando termina una jornada. Este proceso suele comprender las siguientes acciones:

- Enviar mensajes de aviso a los terminales activos.
- Terminar o interrumpir los procesos activos excepto el de la consola maestra.
- Asegurar que toda la actividad sobre el sistema de archivos ha terminado.
- Desmontar el sistema de archivos.

Otras actividades de la administración del sistema son:

- Cambiar las características de los terminales. Símbolos de teclado, velocidad de transmisión, etc.
- Ejecutar periódicamente determinados procesos. Existe un comando *(cron)* que analiza en un archivo especial *(crontab)* determinados procesos que tienen que ejecutarse automáticamente cada minuto, hora, día, etc.

CUESTIONES

1. Definir con su propia terminología el sistema operativo UNIX.
2. ¿Qué se entiende por sesión UNIX?
3. Indicar qué es un comando del sistema operativo UNIX.
4. ¿Qué ventajas tiene la distribución de archivos de usuario en una estructura jerárquica con directorios?
5. ¿Qué se conoce como imagen estática de un programa?
6. ¿Qué función realiza el intérprete de comandos Shell del sistema operativo UNIX?
7. ¿Qué funciones tiene asignadas el Administrador del Sistema?
8. ¿Qué opciones se realizan en el arranque del sistema y quién edita el programa que controla dicho arranque?
9. ¿Qué acciones se realizan en el apagado del sistema?
10. Indicar qué empresa desarrolló el sistema operativo UNIX, quienes fueron sus principales autores y qué ordenadores fueron los primeros en utilizarlo.
11. Describir brevemente la historia del sistema operativo UNIX y qué versiones son las que actualmente se utilizan.
12. ¿Cuál es la principal ventaja del sistema operativo UNIX?
13. ¿Cuál es su principal inconveniente?
14. ¿Qué símbolos utiliza el sistema operativo UNIX para indicar el prompt?
15. Dibujar y comentar la estructura de la línea de comandos del sistema operativo UNIX.

CAPITULO 14

Sistema Operativo OS/2

14.1. INTRODUCCION

Con el avance tecnológico de los últimos años el hardware de las computadoras personales, en 1987 y con la aparición del microprocesador 80286 de Intel, las empresas Microsoft (MS) e International Business Machines (IBM) desarrollaron un nuevo sistema operativo para microcomputadoras con la intención de implantarlo en los modelos PC-AT y PS/2 50, 60 y 80 y con total compatibilidad con el software desarrollado en el entorno DOS. Este sistema operativo llamado Operating System/2 (OS/2) permite la multitarea a diferencia del DOS y tiene mayores posibilidades de direccionamiento de memoria ya que posee un mayor rango de direccionamiento de memoria física y además puede utilizar la técnica de la memoria virtual con lo cual el espacio de direccionamiento virtual disponible es aún mayor.

Otra de las características fundamentales del OS/2 es la posibilidad de ejecutar simultáneamente distintas partes independientes de un mismo programa.

El sistema operativo OS/2 se diseñó para cubrir los siguientes objetivos:

- Sobrepasar la limitación de direccionamiento de memoria existente en aquel momento (640 K).
- Permitir la conexión de computadoras en redes de área local para facilitar el intercambio de programas e información entre usuarios y puedan compartirse recursos comunes.
- Rentabilizar la potencia de los microprocesadores aparecidos en aquel momento (80286 y 80386).
- Debe permitir una gran flexibilidad a todos los cambios que puedan presentarse en el futuro referentes a evolución del propio sistema operativo.
- Debe mantener la misma flexibilidad ante la aparición de nuevos microprocesadores, dispositivos periféricos y sus controladores, así como para la evolución de los ya existentes.
- Debe ser en todos los sentidos un sistema con futuro.
- Minimizar las situaciones de error y que éstos no afecten a otros procesos.

Para que exista compatibilidad con el software del DOS, el sistema OS/2 posee dos modos de ejecución de sus programas:

- **Proceso en modo real.** En él, un programa puede acceder al hardware directamente, realizar operaciones de entrada/salida de bajo nivel y controlar todos los recursos de

la computadora, no existiendo interferencias entre un programa y otro, es decir, se ejecuta un solo programa.

- **Proceso en modo protegido.** Es el modo que permite la ejecución de programas en multiprogramación o multitarea, es decir, varios programas pueden ejecutarse simultáneamente.

El sistema operativo OS/2 precisa una configuración mínima del hardware para su funcionamiento que se encuentra representada en la Figura 14.1, donde además y entre paréntesis aparecen los valores más recomendados para una mayor rentabilidad del sistema.

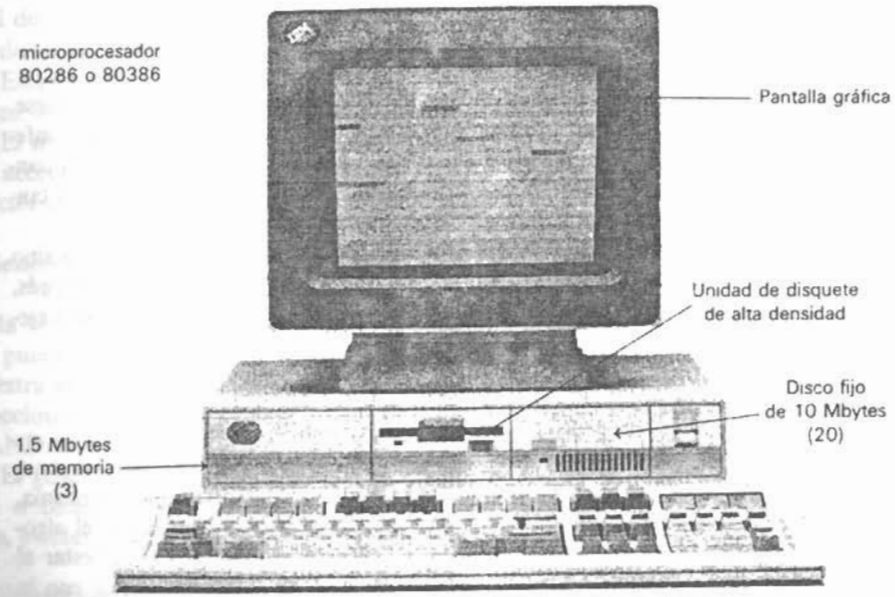


Figura 14.1. Configuración hardware para soportar OS/2. (Cortesía de IBM.)

En resumen, las nuevas incorporaciones del OS/2 frente al DOS se encuentran en aspectos relacionados con:

- Multiprogramación o multitarea.
- Gestión de memoria virtual.
- Utilización de dispositivos virtuales.
- Comunicación entre procesos concurrentes.
- Interconexión en redes.
- Gráficos avanzados.
- Interfaz potente con el usuario.

14.2. HISTORIA

En el año 1987 las empresas Microsoft e IBM anunciaron la aparición de un nuevo sistema operativo para microcomputadoras adaptado al momento presente y con un futuro que permita el desarrollo de las microcomputadoras en la década de los 90.

En principio, el punto de partida fue la adaptación al nuevo microprocesador 80286 y se utilizaron muchas rutinas similares a las existentes en versiones del DOS como la 3.3.

En el momento actual sigue desarrollándose el sistema operativo OS/2 y empieza lentamente a tener simpatizantes, aunque la gran popularidad del DOS hace que no termine de introducirse plenamente en el mercado.

14.3. ESTRUCTURA DEL SISTEMA OPERATIVO OS/2

El sistema operativo OS/2 al trabajar en modo real para la ejecución de programa DOS ya existentes, deja libertad al proceso para acceder a la memoria, utilizar dispositivos, etc. La estructura de este modo se encuentra representada en la Figura 14.2.

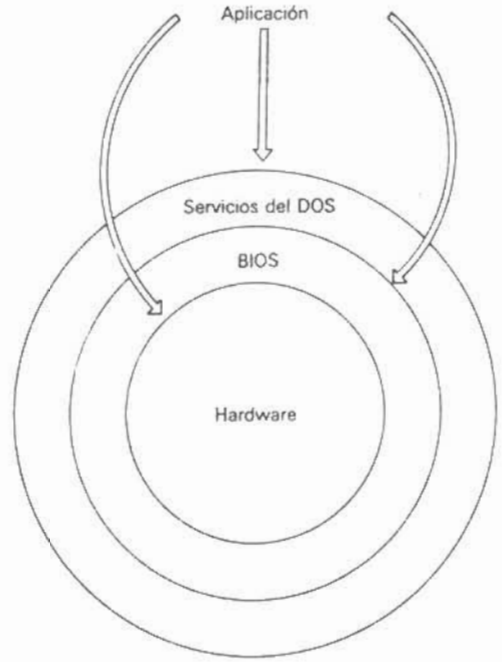


Figura 14.2. Estructura del modo real del OS/2.

El modo protegido para los programas de OS/2 en el que varios de ellos pueden estar coexistiendo en la memoria y compartiendo recursos del sistema, tiene una estructura a través de la cual el propio sistema ejerce el control de los procesos (Figura 14.3).

Figura 14.3. Estructura del modo protegido del OS/2.

En esta figura, el API *(Application Program Interface)* es el interfaz de los servicios del sistema OS/2.

14.4. CONCEPTOS BASICOS

■ Gestión del procesador

Como sistema operativo multitarea, el OS/2 distribuye el tiempo del procesador entre los distintos procesos existentes en cada momento dedicándoles un quantum de forma ciclica, es decir, cada proceso dispone de un pequeño tiempo de procesador cuando le llega el turno que se establece de forma circular. Además, OS/2 permite la asignación de prioridad a determinados procesos frente a los demás, por tanto, la planificación del procesador se realiza comenzando por el grupo de procesos de mayor prioridad de forma cíclica para continuar con el mismo procedimiento en los niveles inferiores. Como es sabido, pueden producirse problemas de postergación indefinida en estos casos.

Cuando en cualquier nivel de prioridades el OS/2 termina el quantum asignado a un proceso, comprueba la lista de procesos de mayor prioridad y si existe alguno en estado preparado le concede el siguiente quantum. Si se queda en el mismo nivel de prioridad le será asignado al que por turno circular le corresponda (Figura 14.4).

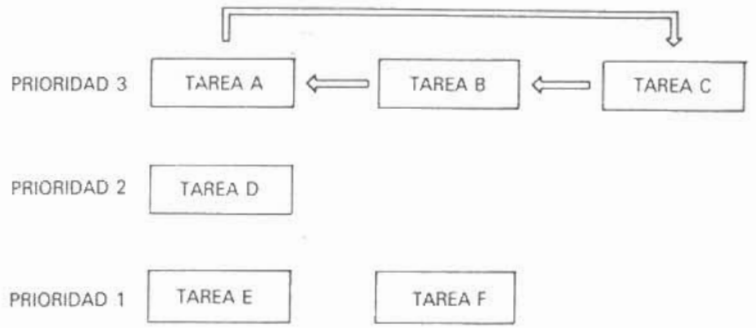


Figura 14.4. Planificación circular por niveles de prioridad.

Para eliminar los posibles problemas de postergación indefinida que puedan presentarse, el sistema OS/2 dispone de prioridad dinámica, con la cual los procesos de prioridad inferior van adquiriendo cada cierto tiempo mayor prioridad hasta que consiguen tomar el control del procesador en cuyo momento vuelven a su prioridad inicial repitiéndose el ciclo tantas veces como sea necesario.

En realidad el control del procesador no se concede directamente a cada proceso sino que lo hace a las distintas tareas independientes de cada proceso que se denominan **threads,** con lo que se consigue que varias partes o threads de un mismo proceso puedan estar ejecutándose simultáneamente.

■ Gestión de memoria principal

Para poder ejecutar varios programas simultáneamente, el sistema operativo OS/2 los mantiene ubicados en memoria de tal forma que cuando los programas no caben en la misma, utiliza la técnica del intercambio o swapping sustituyendo los segmentos mediante el algoritmo LRU *(Less Recently Used).* Por otra parte, cada programa no tiene por qué estar al completo en la memoria ya que además OS/2 utiliza la técnica de memoria virtual con la que pueden ser ejecutados programas de mayor tamaño que la memoria física.

El OS/2 realiza una correspondencia indirecta entre las direcciones de los programas y la memoria física. Este proceso de correspondencia indirecta es la mencionada memoria virtual en la que el concepto de segmento-desplazamiento del DOS se sustituye por un manipulador de direcciones que se denomina **selector** que actúa sobre una tabla de segmentos. Con todo ello se generan direcciones de 24 bits, con lo cual se pueden direccionar 16 Mbytes. La memoria se divide en segmentos de 64 K.

■ Gestión de memoria secundaria

Para la gestión del espacio disponible en disco, OS/2 utiliza una tabla de asignación de archivos (FAT) donde se encuentra la información correspondiente a sectores libres y ocupados, así como de los sectores defectuosos por alguna causa del propio soporte.

El sistema operativo OS/2 utiliza para el almacenamiento de datos en disco el mismo formato que el sistema DOS, es decir, un disco creado en OS/2 puede ser utilizado en DOS y viceversa.

De igual forma se admite el mismo tipo de estructura de subdirectorios en árbol a partir del directorio raiz.

■ Gestión de entrada/salida

Cada dispositivo posee un controlador para ocultar al sistema y a los procesos las características particulares del mismo. La misión de estos controladores es, además de la propia comunicación, la gestión total del dispositivo al estar en un entorno multitarea. Deben decidir si el dispositivo puede ser accedido por dos procesos simultáneamente o no, si un proceso puede acceder directamente a un puerto de entrada/salida del dispositivo, etc.

Existen en el entorno OS/2 dos tipos de dispositivos con sus correspondientes controladores: los que se comunican en modo carácter y los que lo hacen en bloques.

El mecanismo de prioridades facilita que un proceso pueda, en un momento determinado, acceder directamente a un dispositivo con la consiguiente velocidad que esto puede proporcionar.

■ Sesiones del OS/2

Cada vez que se ejecuta un programa en OS/2 se dice que se realiza una sesión. Las sesiones pueden producirse en modo real y en modo protegido, para lo cual el gestor de sesiones muestra al comenzar cada sesión un pequeño menú donde ofrece la opción de elegir el modo seleccionado entre el intérprete de comandos del DOS para el modo real (COMMAND.COM) o su equivalente para el modo protegido (CMD.EXE).

El gestor de sesiones es el encargado de iniciar la ejecución de los programas en modo real o compartido, ejecutar varios simultánea y concurrentemente, conmutar de uno a otro, etcétera.

■ Dispositivos virtuales

En la multiprogramación, el OS/2 tiene que controlar todas las operaciones de entrada/salida que produzcan los procesos que se están ejecutando, por tanto, si tenemos en un determinado momento dos programas en ejecución que hacen uso del teclado y la pantalla, el sistema operativo debe hacer ver a los procesos que disponen de todos los recursos del sistema en exclusividad, es decir, aparentemente se supone que existen dos teclados y dos pantallas, una de cada para los procesos activos. Como estos dispositivos no son reales tal como se muestra, reciben el nombre de dispositivos virtuales.

En estos casos, las salidas de cada programa (en pantalla o impresora) se guardan en colas para que no se produzcan mezclas de datos de salida de procesos (Figura 14.5).

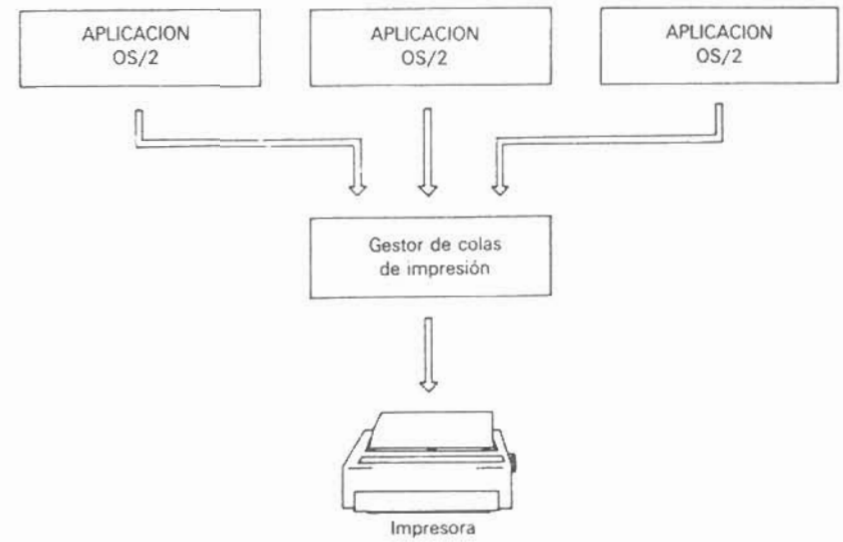


Figura 14.5. Cola de salidas en impresora.

■ Comunicación entre procesos

Al ser el OS/2 un sistema multitarea, los procesos que se ejecutan en un determinado momento deben ser capaces de compartir información y por consiguiente puede existir entre ellos concurrencia. Por este motivo, es necesario que existan mecanismos de comunicación y sincronización entre dichos procesos. OS/2 ofrece a tal fin los siguientes:

- **Memoria compartida.** Permite que dos o más programas compartan ciertan porción de memoria a través de la cual puedan sincronizarse por medio de un protocolo particular.
- **Semáforos.** Son indicadores o contadores compartidos entre dos o más programas para que a través de ellos se establezca la sincronización de dichos programas concurrentes.
- **Colas de mensajes.** Con esta técnica un programa puede enviar un mensaje a otro programa para que este último lo procese antes de tomar el control del procesador.

■ Comunicaciones

El sistema operativo OS/2 posee un gestor de LAN que le permite la conexión a redes de área local estandarizadas. Proporciona comunicación entre programas, compartición de archivos y utilización de recursos remotos, además permite la coexistencia con equipos bajo otros sistemas operativos (Figura 14.6).

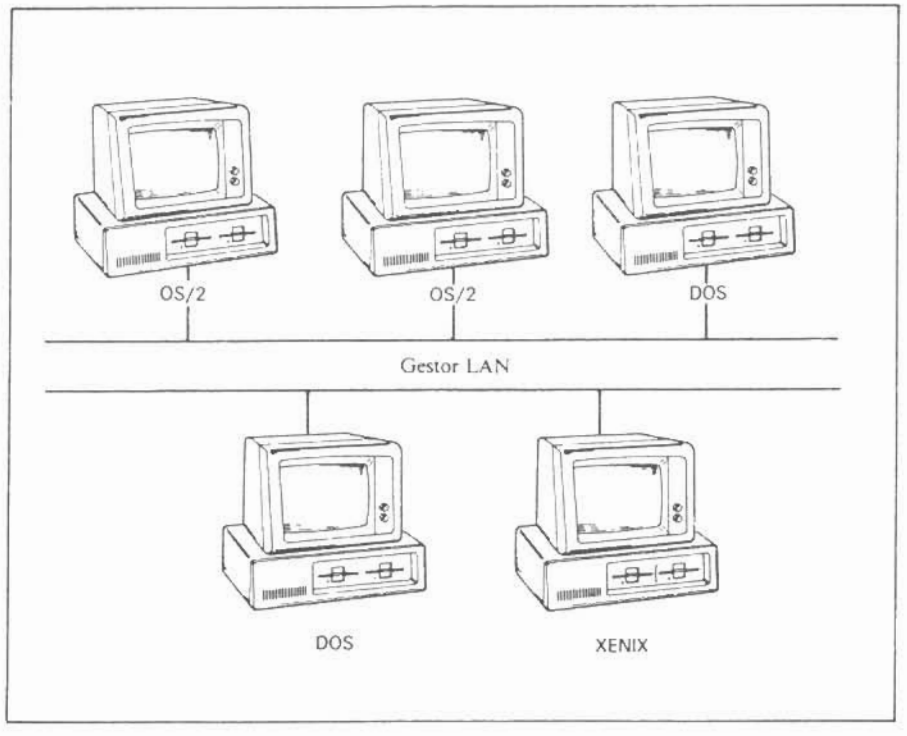


Figura 14.6. Gestor LAN del OS/2.

14.5. EL INTERPRETE DE COMANDOS

El intérprete de comandos del OS/2 es el CMD.EXE, que carga al inicio en memoria una serie de comandos internos (CLS, ECHO, DATE, EXIT, etc.) residentes permanentemente en memoria, y permite la ejecución del resto de comandos externos residentes en disco (LABEL, COMP, XCOPY, etc.).

También existe para el modo real el intérprete COMMAND.COM similar al del sistema operativo DOS.

Al igual que en el DOS, el sistema OS/2 tiene la posibilidad de procesar comandos y programas por lotes. En este caso, se dispone de algunas mejoras sobre el proceso por lotes del DOS como puede ser por ejemplo la posibilidad de llamadas entre procesos por lotes (CALL-sólo válido en modo protegido). Cuando se trabaja en modo real el proceso de arranque es el AUTOEXEC.BAT, mientras que si se trabaja en modo protegido el sistema busca y ejecuta el STARTUP.CMD, realizando ambos la misma misión.

14.6. GESTOR DE PRESENTACION (PRESENTATION MANAGER)

Se trata de un interfaz para la presentación en pantalla de toda la gestión y proceso realizado en un equipo bajo OS/2. A través de él puede verse en pantalla la ejecución de varios programas simultáneamente, se pueden compartir datos entre programas, etc. (Figura 14.7).

Figura 14.7. Gestor de presentación de OS/2.

El gestor de presentación posee una serie de rutinas para facilitar el desarrollo de programas en los siguientes aspectos:

- Interfaz de ratón.
- Gráficos.
- Ventanas múltiples.
- Rutinas temporizadoras.
- Recursos compartidos.

Desde el gestor de presentación se puede ejecutar un programa, controlar la pantalla y en definitiva realizar la coordinación correspondiente a un conjunto de programas que se ejecutan concurrentemente, compartiendo recursos, además dispone de un conjunto de menús de ayuda que facilitan el uso de la computadora a cualquier persona experta o no.

El gestor de presentación permite la exposición en pantalla de todos los programas que están en un determinado momento activos en el sistema a través de superposición de ventanas en las que el usuario puede cambiar de lugar una ventana, superponer en una posición o en otra, etc.

CUESTIONES

1. Describir brevemente las razones por las cuales apareció el sistema operativo Operating System/2 (OS/2).
2. ¿Qué ventajas presenta OS/2 frente al sistema operativo DOS?
3. ¿Cuál es el grado de compatibilidad entre los sistemas operativos DOS y OS/2?
4. Dibujar de forma esquematizada la estructura del sistema operativo OS/2 en sus dos modos de trabajo (real y protegido).
5. ¿En qué consisten y cuáles son las diferencias de los modos de trabajo de sistema OS/2 real y protegido?
6. ¿Cómo se realiza la gestión del procesador en este sistema operativo?
7. ¿Cómo se gestiona la memoria principal en este sistema operativo?
8. ¿Qué diferencias existen entre las formas de gestionar los discos en los sistemas DOS y OS/2?
9. ¿A qué se denomina dispositivo virtual?
10. ¿Cuáles son los mecanismos de comunicación entre procesos que ofrece OS/2 en el entorno multitarea para la sincronización de procesos?
11. ¿Cuáles son las ventajas que ofrece la posibilidad de conexión de un equipo bajo OS/2 a una red de área local?
12. ¿Qué es y para qué sirve en OS/2 el denominado Gestor de Presentación (Presentation Manager)?

CAPITULO 15

Sistema Operativo MVS

15.1. INTRODUCCION

El Sistema Operativo MVS *(Memory Virtual System)* fue comercializado por International Business Machines (IBM) a mediados de los setenta, siendo el utilizado en sus grandes sistemas (sistemas 370 y 390).

Los objetivos principales de diseño del MVS fueron suministrar un alto rendimiento, disponibilidad y compatibilidad entre entornos de grandes sistemas, suministrando una estabilidad mayor a los sistemas comercializados en aquella época. Para ello, MVS dispone de rutinas de recuperación funcional mediante las que se recuperan errores de Hardware y Software.

Se trata de un sistema orientado a la tolerancia de fallos, por ejemplo en un sistema multiprocesador, el fallo de uno de sus procesadores no tiene por qué provocar la caída de todo el sistema. En este caso, MVS detecta el fallo y posibilita al resto de procesadores para realizar las funciones del procesador del fallo.

15.2. HISTORIA DEL MVS

El Sistema Operativo MVS es un sucesor del MVT *(Multiprogramming with a Variable number of Tasks)*, que fue un sistema orientado a procesos batch que permitía el tratamiento concurrente de hasta 15 trabajos. Este sistema MVT, a pesar de su orientación inicial, fue introduciendo con el tiempo componentes de tiempo compartido *(TSO-Time Sharing Option)* y gestión de la entrada/salida de los diferentes trabajos *(HASP-Houston Automatic Spooling Priority)*.

En 1972, IBM introdujo el OS/SVS *(Single Virtual Storage)* como un sistema intermedio en el tiempo para aprovechar las ventajas de la arquitectura IBM/370 y la memoria virtual; en realidad SVS fue el sistema operativo MVT con memoria virtual.

El siguiente paso fue la aparición del MVS/370 que incorporaba como facilidades estándar las de los sistemas anteriores y ofrecía a cada usuario un espacio de direccionamiento de 16 Mbytes, siendo su principal novedad el gestor de recursos.

Posteriormente, el sistema ha ido evolucionando y en 1983 apareció la versión XA *(Extended Addressing)* del MVS cuyas diferencias principales fueron la ampliación del direccio-

namiento de memoria de usuario a 2 Gbytes y la introducción del subsistema de canales que gestiona todas las operaciones de entrada/salida liberando de ellas a los procesadores.

La última versión aparecida en 1988 ha sido la 3, conocida como MVS/ESA *(Enterprise Systems Architecture)*, caracterizada por posibilitar la utilización de los espacios de direcciones para datos con la consiguiente ganancia en tiempo (Figura 15.1).

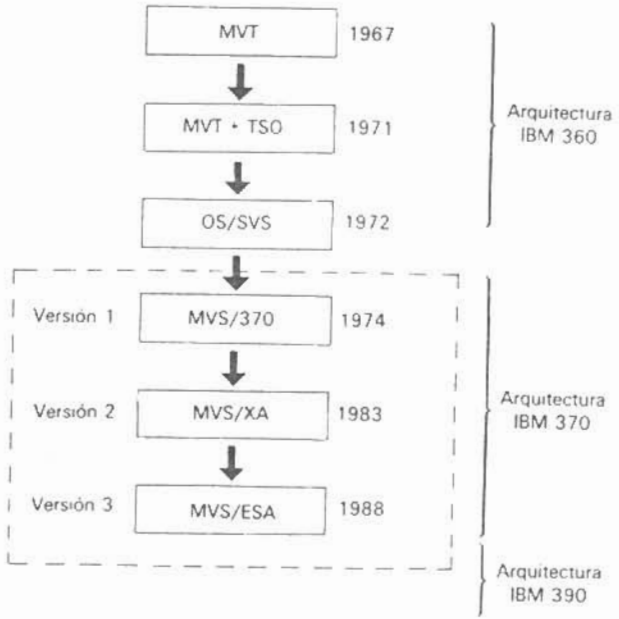


Figura 15.1. Historia del MVS.

15.3. ESTRUCTURA DEL SISTEMA OPERATIVO MVS

Se trata de un sistema gestionado por interrupciones cuyo componente principal es el núcleo encargado de gestionar el o los procesadores y atender las interrupciones.

El núcleo se encuentra residente en memoria y contiene los bloques principales que definen las librerías del sistema, las rutinas de recuperación y los bloques de control de los dispositivos de entrada/salida.

En la Figura 15.2 podemos ver un mapa de la estructura de los diferentes componentes del MVS.

El Sistema Operativo MVS dispone de un planificador maestro *(Master Scheduler)* que, en respuesta a comandos del operador y a parámetros de inicialización del sistema, arranca los diferentes componentes del Sistema Operativo como son:

- **Programa de intercomunicación de proceso** *(Program Call/Authorization)*. Este es el primer componente que se inicializa y permite la comunicación entre los diferentes espacios de comunicaciones (usuarios u otros componentes del sistema).

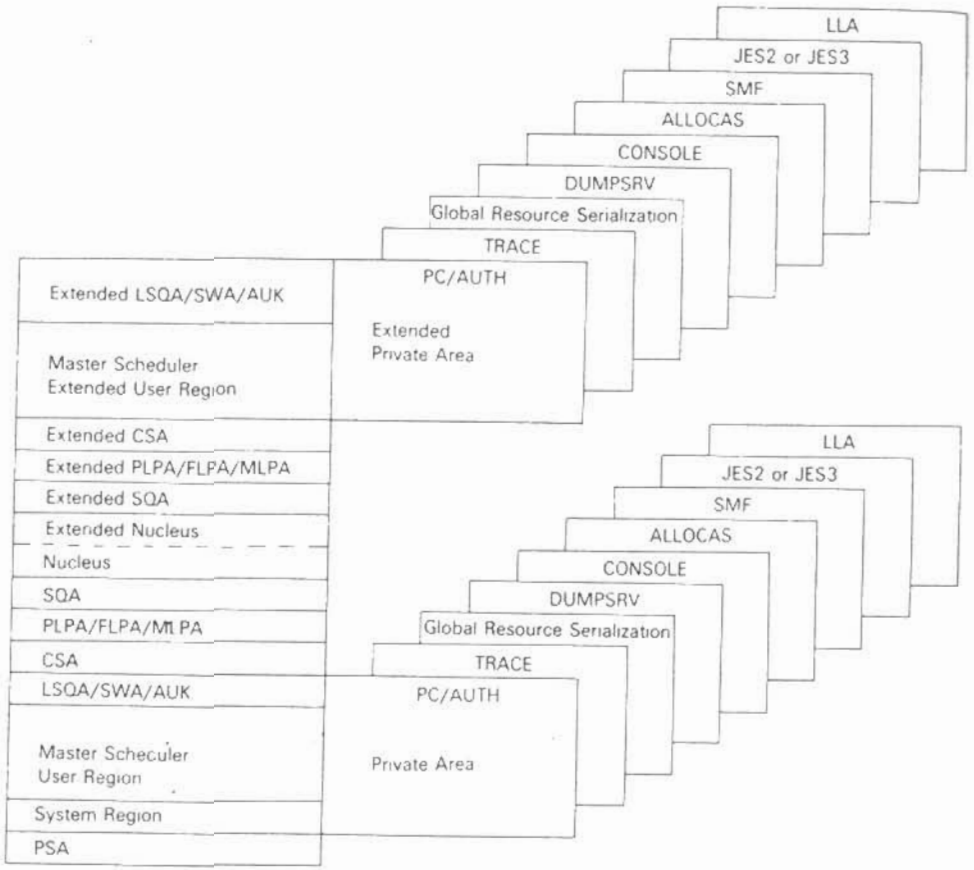


Figura 15.2. Estructura del Sistema Operativo MVS (Cortesía de IBM).

- **Programa de seguimiento** *(TRACE)*. Permite controlar el seguimiento de las operaciones realizadas por cualquier elemento de software.
- **Componente para la serialización de recursos** *(Global Resource Serialization)*. Encargado de asegurar la utilización serializada (en serie) de recursos.
- **Servicios de volcado** *(DUMPSRV)*. Posibilita la obtención de vuelcos del contenido de la memoria de los diferentes espacios de direccionamiento.
- **Tarea de comunicaciones** *(Communications Task-CONSOLE)*. Permite la comunicación con el operador.
- **Servicio de asignación** *(ALLOCAS)*. Realiza toda la asignación de recursos.

- **Facilidades de gestión del sistema** *(SMF).* Crea registros de las actividades del sistema que permiten contabilizar la utilización de los diferentes recursos del sistema.
- **Subsistema de entrada de trabajos** *(JES2* o *JES3).* Es el planificación de la entrada y salida de trabajos.
- **LNKLST Lookaside** *(LLA).* Es el componente del sistema que acelera la búsqueda de módulos cargables.

Además, el sistema operativo MVS poseerá tantos espacios de direccionamiento como usuarios y trabajos estén procesándose en un determinado momento.

15.4. CONCEPTOS BASICOS

MVS como el resto de los sistemas operativos posee tres categorías de recursos:

- Procesadores.
- Memoria principal.
- Dispositivos de entrada/salida.

La gestión de estos recursos la realiza el gestor de recursos del sistema *(SRM-System Resources Manager),* que tiene dos objetivos principales:

- Lograr un uso óptimo de estos recursos desde el punto de vista del sistema (realizar el mayor número de trabajos posible).
- Conseguir que los usuarios reciban un tiempo de respuesta óptimo.

▪ Gestión del procesador

Se realiza mediante técnicas Round Robin (RR) con prioridades.

▪ Gestión de la memoria

Se realiza con técnicas de memoria virtual mediante la utilización de la paginación segmentada. La sustitución de páginas se realiza por medio de algoritmos del tipo LRU.

▪ Gestión de la entrada/salida

Una configuración con MVS puede incluir más de 4.000 dispositivos de entrada/salida y procesar muchos programas concurrentemente. Para manejar las operaciones de entrada/salida de todo este entorno, a partir de MVS/XA, el sistema utiliza un procesador independiente.

Este procesador y sus componentes es lo que se conoce como el **Subsistema de canal.**

15.5. SERVICIOS DEL SISTEMA Y FACILIDADES

Además de los componentes del sistema anteriormente citados, MVS ofrece programas de utilidad, componentes para la gestión de archivos y componentes para la gestión de teleproceso.

Entre ellos podemos citar los siguientes:

- **Compiladores.** MVS admite todo tipo de lenguajes, siendo el más utilizado hoy en día el lenguaje COBOL.
- **Utilidades para la gestión y manejo de archivos.** Posee programas para ordenación, copiado, borrado, etcétera, de archivos.
- **Facilita diferentes métodos de acceso a archivos.** Admite:
  - Secuencial *(SAM).*
  - Indexado *(ISAM).*
  - Virtual indexado *(VSAM).*
  - Directo *(BDAM).*
- **Software de comunicaciones.** Contiene un conjunto de métodos de acceso para el soporte del teleproceso admitiendo distintos protocolos (SNA, OSI, BSC, X.25, etcétera.)
- **Facilidades de tiempo compartido.** Admite la utilización compartida del sistema por un gran número de usuarios (TSO, CICS).

CUESTIONES

1. ¿Qué es MVS, cuándo se comercializó y por quién?
2. ¿Cuáles fueron los principales objetivos de diseño del sistema operativo MVS?
3. ¿Cómo actúa ante fallos el sistema operativo MVS?
4. Describir brevemente la historia del MVS.
5. ¿Cuáles son las características principales de la versión XA del sistema operativo MVS?
6. ¿En qué se basa la estructura del sistema operativo MVS?
7. ¿Cuál es la misión del planificador maestro?
8. Enumerar los diferentes componentes del sistema operativo MVS.
9. ¿Cuáles son los objetivos principales del gestor de recursos del sistema?
10. ¿Cómo se gestiona el procesador en el sistema MVS?
11. ¿Cómo se gestiona la memoria en el sistema MVS?
12. ¿Quién realiza la gestión de las operaciones de entrada/salida en las versiones actuales del sistema operativo MVS?
13. ¿Cuáles son los diferentes métodos de acceso a ficheros que admite el sistema MVS?
14. Enumerar los distintos servicios del sistema y facilidades que ofrece al usuario el sistema operativo MVS.

CAPITULO 16

Sistema Operativo VMS

16.1. INTRODUCCION

VAX *(Virtual Address Extension)* es una familia de computadoras de 32 bits con memoria virtual de Digital Equipment Corporation (DEC). Los sistemas VAX-11 son los sucesores de los populares PDP-11 que trabajaban con 16 bits.

Esta serie de minicomputadoras, debido al descenso en el coste del hardware y a las altas prestaciones que permiten los sistemas diseñados con la tecnología actual, han conseguido transformar pequeñas computadoras en sistemas complejos capaces de soportar multiprogramación, tiempo compartido, tiempo real y proceso por lotes. Los sistemas operativos actuales proporcionan una funcionalidad que hace unos años sólo era posible en grandes computadoras y hoy en día se puede encontrar en computadoras de menor tamaño y, lo que es más importante, de menor precio.

El sistema operativo de esta familia de computadoras es el **VAX/VMS** o Sistema de Memoria Virtual *(Virtual Memory System),* que es un sistema operativo de tiempo compartido, permitiendo que pueda ser utilizado también como uno de tiempo real y, además, permitiendo la utilización de una amplia gama de lenguajes de alto nivel y programas de utilidad. Aunque VMS es un sistema operativo multiusuario, no es como un sistema de tiempo compartido tradicional. Podemos decir que es un sistema orientado a las aplicaciones que optimiza el rendimiento total del sistema así como las actividades de alta prioridad.

Estas minicomputadoras han necesitado para cubrir los objetivos descritos, una mayor capacidad de ejecución de procesos simultáneamente y de gran tamaño, basándose para ello en la gestión de la memoria virtual. Por tanto, la gama VAX-11 es una extensión de la familia PDP-11 con gestión de memoria virtual.

Existe una prácticamente completa compatibilidad entre el software desarrollado en computadoras PDP-11 para que sean ejecutados en los de la gama VAX-11; por tanto, es posible la migración de dicho software.

La estructura básica de estas minicomputadoras se compone de tres elementos interrelacionados; éstos son:

- **Unidad Central de Procesos.** Posee los siguientes elementos:
  - **Procesador.** Multiprogramado de 32 bits.
  - **Memoria cache.** Con capacidad de 4 a 64 Kbytes (según el modelo) de alta velocidad.
  - **Acelerador de coma flotante.** Realiza operaciones más rápidamente que la unidad aritmético lógica del procesador.

- **Memoria principal.** Para el almacenamiento de programas y datos con una capacidad de hasta 16 Mbytes en los primeros modelos y 64 Mbytes en los más modernos.
- **Subsistema de entrada/salida.** Se encarga de la conexión con los periféricos a través del bus interno (SBI) y de los buses normalizados UNIBUS y MASSBUS.

La Figura 16.1 muestra la estructura general de una minicomputadora VAX-11.

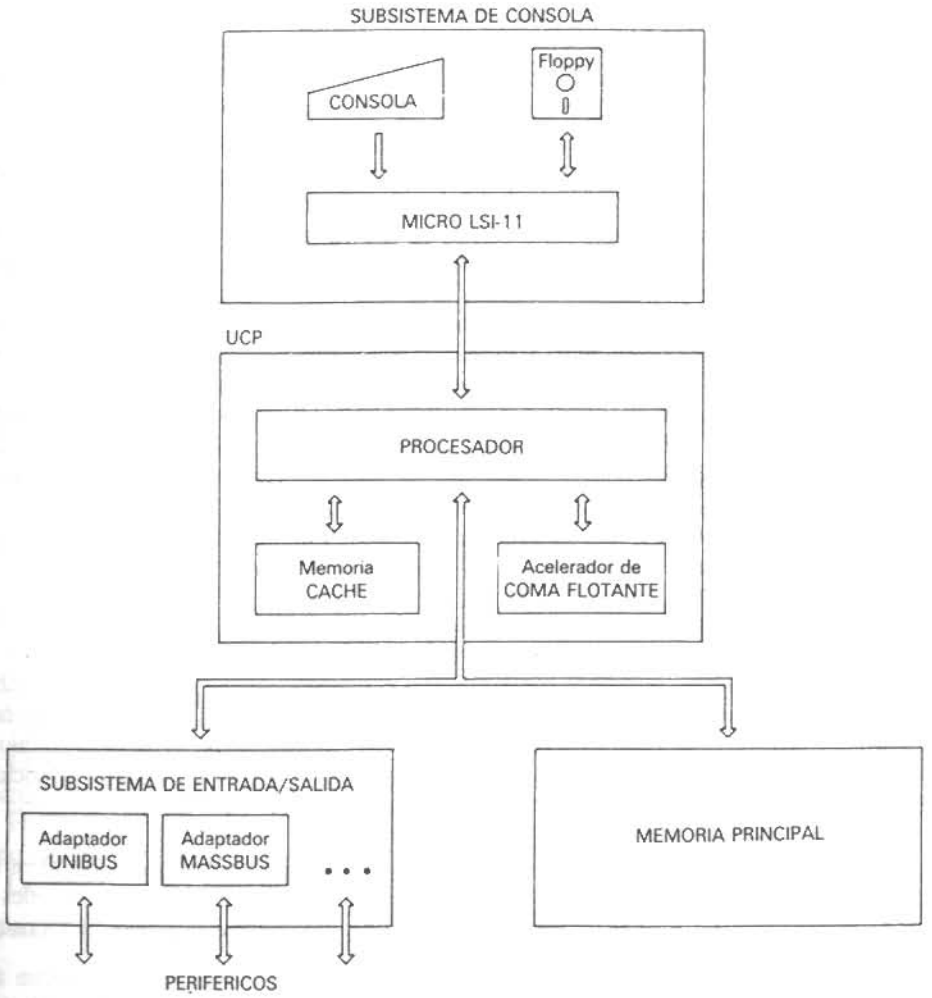


Figura 16.1. Estructura básica del VAX-11.

Además, cuenta con una consola inteligente que posee una microcomputadora LSI-11 con memoria local y unidad de disquete. Esta unidad se utiliza para la instalación del sistema y actualización del software interno.

16.2. HISTORIA DE LA FAMILIA VAX

Como ya se ha dicho, las computadoras PDP-11 quedaron en desuso cuando aparecieron las computadoras de 32 bits con mayores capacidades de almacenamiento y proceso.

Los tres primeros sistemas VAX-11 fueron:

- **VAX-11/730.** Es un modelo relativamente sencillo y barato con grandes posibilidades en cuanto a utilización de periféricos se refiere.
- **VAX-11/750.** Es el modelo medio construido con tecnología LSI que posee características intermedias entre los otros dos.
- **VAX-11/780.** Es el más potente de la gama y se utiliza para el manejo de grandes bases de datos con una gran potencia en el manejo y gestión de procesos.

Otros modelos más actuales y potentes son: El VAX 8200, VAX 8300, VAX 8600, VAX 8650, MicroVAX-32, VAX VLSI y el MicroVAX-1, todos ellos con características muy superiores a los tres primeros.

El sistema operativo de todos ellos es el VMS en sus distintas versiones.

16.3. ESTRUCTURA DEL SISTEMA OPERATIVO VMS

El sistema operativo VAX/VMS tiene una estructura similar a la del sistema operativo RSX de la familia PDP-11. Muchos de los elementos funcionales internos de ambos sistemas son idénticos, como por ejemplo el sistema de archivos.

El sistema tiene una estructura en cuatro niveles: núcleo, ejecutivo, supervisor y nivel de usuario (Figura 16.2).

Figura 16.2. Estructura del sistema operativo VMS.

Es posible, por tanto, trabajar en cuatro modos diferentes que definimos a continuación:

- **Modo kernel.** Ocupado por el núcleo del sistema operativo y se compone de los elementos: planificador, gestor de páginas y controladores de entrada/salida. Es el modo de mayor prioridad.
- **Modo ejecutivo.** Se utiliza en las llamadas al sistema operativo y además contiene el sistema de gestión de registros. Posee la siguiente prioridad al modo kernel.
- **Modo supervisor.** Se utiliza para dar distintos servicios, entre los que se encuentra el intérprete de comandos. Es el siguiente modo en prioridad.
- **Modo usuario.** Desde el que pueden utilizarse una amplia gama de compiladores, utilidades y depuradores de programas. Tiene menor prioridad que los anteriores.

16.4. CONCEPTOS BASICOS

16.4.1. Planificación

El sistema operativo VAX/VMS asegura una alta prioridad a los procesos de tiempo real con respecto al resto.

Los procesos se pueden encontrar en distintos estados (Figura 16.3). Las transiciones se producen como resultado de un evento del sistema. El estado de un proceso queda reflejado en su PCB (bloque de control de proceso).

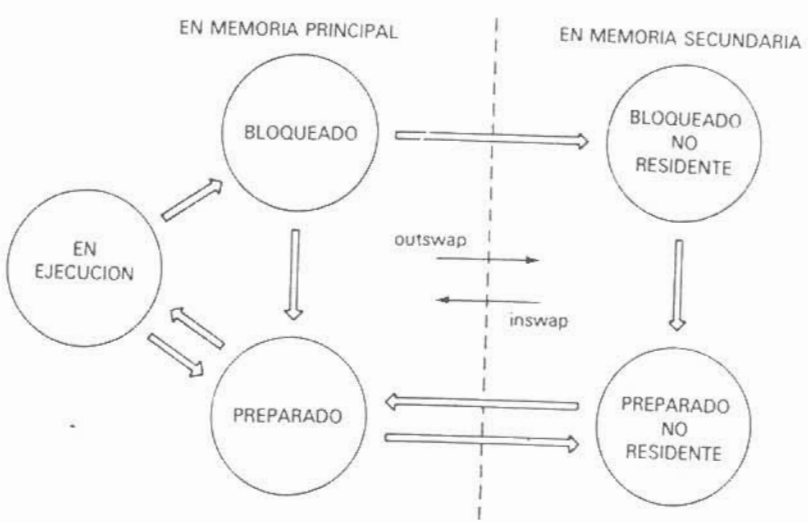


Figura 16.3. Transiciones de estados de procesos.

El planificador del procesador reconoce 32 prioridades, siendo la 31 la más alta y la 0 la más baja. El rango comprendido entre la 31 y la 16 está destinado para los procesos de tiempo real y el resto para los procesos normales.

Cuando se crea un proceso, el sistema le asigna una prioridad que se considera como base.

El planificador mantiene una cola por cada prioridad de manera que los procesos de igual prioridad estarán en una misma cola, ordenada por orden de llegada a la misma.

El planificador no altera en ningún caso la prioridad de un proceso de tiempo real durante su ejecución, mientras que puede incrementarla en el caso de los procesos normales pero nunca disminuirla por debajo de la prioridad base del proceso correspondiente.

Siempre que se produzca un evento cuyo efecto sea la ejecución de un proceso, éste se ejecutará si no existe otro de mayor prioridad haciendo uso del procesador. Si es un proceso de tiempo real, retendrá el control del procesador hasta que termine su ejecución, realice una operación de entrada/salida, entre en un estado de espera por alguna causa, o sea expulsado del procesador por otro proceso de tiempo real de mayor prioridad.

Cuando no se estén ejecutando procesos de tiempo real, se podrán ejecutar procesos normales distribuyendo el tiempo del procesador entre los procesos existentes de mayor prioridad.

La selección del proceso a ejecutar es igual al procedimiento descrito para los de tiempo real, con la diferencia de que el planificador modifica la prioridad de estos procesos cuando se produce un evento que afecta a un proceso normal y cuando uno de ellos sea seleccionado para su ejecución en el procesador. Dicha modificación consiste en elevar la prioridad pero sin pasar del valor 15, colocando el proceso al final de la cola de la nueva prioridad. El incremento de prioridad dependerá del tipo de evento que se produzca, siendo máxima para aquellos relacionados con acciones interactivas desde un terminal de usuario.

Cada vez que un proceso normal toma el control del procesador se disminuye su prioridad en una unidad, siempre y cuando no haya llegado a su prioridad base.

16.4.2. Gestión de la memoria

La gestión de la memoria se realiza por el proceso combinado de paginación e intercambio. La paginación se realiza cuando aparece una petición que produce la correspondiente falta de página y cuando es necesario dejar hueco para una página entrante. El intercambio se realiza eliminando un proceso sustituyéndolo por otro en la memoria principal. Funciona en paralelo con el planificador determinando qué procesos intercambiar entre memoria externa y principal y viceversa.

Las páginas en este sistema son de 512 bytes, menores a las de la mayoría que otros sistemas. Las tablas de páginas son la base para la transformación de direcciones virtuales en direcciones físicas y especifican el permiso de acceso (nulo, sólo lectura, lectura/escritura) para cada uno de los cuatro modos de trabajo.

16.4.3. ENTRADA/SALIDA EN EL SISTEMA OPERATIVO VAX/VMS

El sistema de entrada/salida se realiza en varias fases representadas en la Figura 16.4. La entrada/salida es atendida del nivel más lógico al nivel más físico. En el más alto nivel el

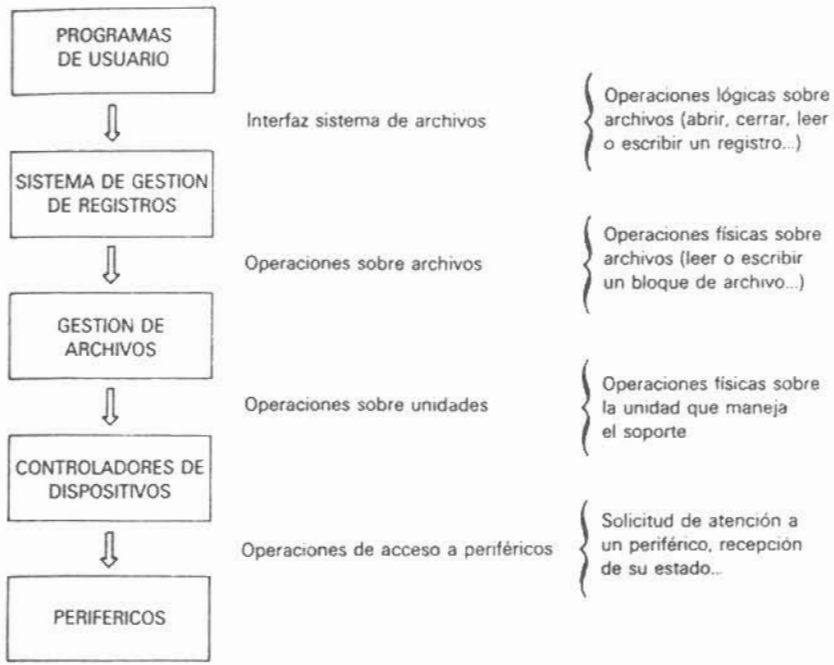


Figura 16.4. Sistema de entrada/salida.

sistema VMS atiende a los archivos, registros y campos de registros. A continuación se ocupa de las operaciones físicas sobre dispositivos periféricos así como la organización de sus entradas y salidas.

Las herramientas de programación de entrada/salida son:

- El sistema de gestión de registros (RMS).
- El sistema de servicios de entrada/salida.

El RMS proporciona acceso independiente de los dispositivos a aquellos que están estructurados como archivos, permitiendo al usuario el procesamiento lógico de los registros.

El sistema de servicios de entrada/salida proporciona acceso independiente y dependiente de los dispositivos por medio de llamadas al sistema operativo.

La entrada/salida del VMS es normalmente asíncrona.

■ **Control de entrada/salida**

Las estructuras de datos se utilizan para describir el estado del sistema de entrada/salida. Estos bloques especiales de control se utilizan para describir cada:

- Adaptador del bus.
- Controlador.
- Periférico.
- Petición de entrada/salida.

Existen seis tipos diferentes de bloques de control en el sistema VMS:

- **Bloque de datos del dispositivo** *(Device Data Block-DDB).* Contiene información común de todos los dispositivos del mismo tipo conectados a un mismo controlador.
- **Bloque de control del dispositivo** *(Unit Control Block-UCB).* Contiene las características y el estado de un único dispositivo, así como el contexto del controlador al que está conectado.
- **Bloque de peticiones de entrada/salida** *(I/O Request Packet-IRP).* Contiene la información que describe una petición de entrada/salida.
- **Bloque de petición del canal** *(Channel Request Block-CRB).* Contiene información del estado del controlador, del dispositivo que se encuentra transmitiendo datos y de aquellos que se encuentran en espera.
- **Bloque de servicio de interrupciones** *(Interrupt Dispatch Block-IDB).* Indica el estado actual de un controlador y se utiliza para determinar qué dispositivo utilizó una determinada interrupción.
- **Bloque de control de adaptador de controladores** *(Adaptor Control Block-ACB).* Contiene información de las características y el estado de los adaptadores de entrada/salida.

El contenido de estos bloques de control en su conjunto indican el estado del hardware en el sistema de entrada/salida en cada momento.

Los elementos del sistema de entrada/salida del VMS son:

- **Cola de servicios de entrada/salida** *(Queue I/O-QIO).* Es un procedimiento para atender las peticiones del usuario referidas a operaciones de entrada/salida.
- **Controladores** *(Device Drivers).* Controlan las operaciones de cada dispositivo periférico.
- **Rutinas de finalización de operaciones de entrada/salida.** Se encargan de completar todas las operaciones de entrada/salida.
- **Gestión de registros del VMS** *(Record Management Services-RMS).* Se utiliza para el control de entradas y salidas a dispositivos de almacenamiento masivo. Facilita el proceso de registros lógicos y maneja automáticamente el bloqueo y desbloqueo de registros en beneficio del usuario. Puede crear archivos de tres tipos: secuenciales, directos e indexados.

16.4.4. Comunicación y sincronización entre procesos

El sistema operativo VMS permite que la comunicación entre procesos se realice sincronizando la ejecución, enviando mensajes y compartiendo datos comunes con los procesos relacionados. Las técnicas utilizadas son:

- **Indicadores de eventos comunes** *(Flags).* Es el método más sencillo y se basa en la utilización de un bit de estado que puede ser fijado o eliminado indicando la ocurrencia de un evento.

- **Buzones** *(Mailboxes)*. Son registros que se utilizan para la comunicación entre procesos para la sincronización en la utilización de sus recursos comunes.
- **Almacenamiento compartido** *(Shared Storage)*. Es el más generalizado en la comunicación entre procesos y consiste en utilizar los protocolos de los propios dispositivos para la sincronización entre procesos.
- **Archivos compartidos** *(Shared Files)*. Los procesos pueden compartir archivos en disco en operaciones de lectura y escritura dependiendo de su organización (fundamentalmente existen restricciones en la secuencial).

16.5. EL INTERPRETE DE COMANDOS

Se conoce con el nombre del **DCL** *(DIGITAL Command Language)* y es un intérprete de comandos interactivo, fácil de aprender y usar y extremadamente flexible.

Permite al usuario identificarse ante el sistema, manipular archivos, desarrollar y verificar programas, y obtener información del sistema, permitiéndole, además, ampliar o redefinir su repertorio de comandos, así como escribir procedimientos de comandos fácilmente.

El DCL incluye:

- Un conjunto de comandos considerado como repertorio básico.
- Un conjunto de caracteres de control que proporcionan funciones especiales.
- Un conjunto de operadores especiales y comandos que pueden usarse para automatizar secuencias de comandos y ampliar el repertorio de éstos.
- Una ayuda muy potente en línea que, interactivamente, ofrece información de forma sencilla y rápida del uso del repertorio básico de comandos.

La línea de comandos contiene normalmente un verbo seguido por uno o más parámetros que identifican el objeto de la operación (por ejemplo un archivo) o que indica cómo debe ser realizada la operación.

Cuando se introduce un comando incompleto, el DCL pregunta uno a uno el resto de parámetros hasta completar la información necesaria para ejecutarlo. Por ejemplo, si se utiliza el comando COPY para hacer una copia de un archivo y no se dan los nombres correspondientes, tendremos:

```
$ COPY
$_FROM: archivo-origen
$_TO: archivo-destino
```

equivalente a la orden completa:

```
$ COPY archivo-origen archivo-destino
```

El símbolo que utiliza el DCL para indicar al usuario que está listo para recibir una orden es el $, con lo que una pantalla de comienzo de una sesión será como indica la Figura 16.5.

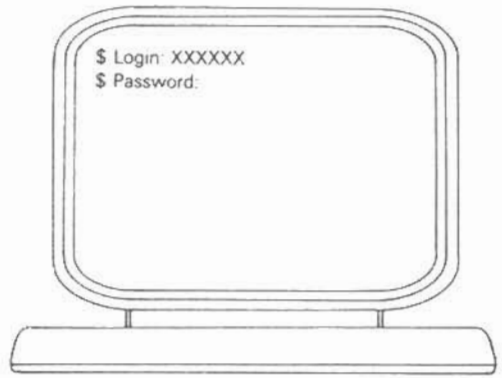


Figura 16.5. Pantalla de inicio de sesión.

Se pueden crear **procedimientos de comandos** consistentes en un archivo compuesto por comandos completos incluyendo el símbolo $ al comienzo. El usuario puede pedir al intérprete de comandos que ejecute dichos procedimientos de tal forma que se irán ejecutando según el orden lógico indicado en el mismo. Para ello se escribe el nombre correspondiente precedido del símbolo @.

```
$ @ archivo-procedimiento
```

También permite ejecutar trabajos batch desde el terminal, ejecutándose bajo el control del operador del sistema dejando libre el terminal del usuario para continuar con el trabajo interactivo.

DCL cuenta también con instrucciones y estructuras de programación de alto nivel que permiten definir y controlar el entorno en una determinada aplicación.

16.6. FACILIDADES DE AYUDA Y DESARROLLO DE PROGRAMAS

El VAX/VMS proporciona un entorno completo de desarrollo de programas para el usuario, cuyas herramientas se ponen a su servicio por medio del intérprete de comandos DCL.

Las principales herramientas son:

▪ Editores de texto

Se utilizan para crear, modificar y mantener archivos. Existen tres editores:

- **EDT**

  Es un editor interactivo de pantalla, que utiliza el VAX/VMS por defecto y permite al usuario entrar y manipular texto y programas, muy potente, flexible y fácil de manejar para un usuario inexperto. Presenta capacidades y facilidades que deleitan a los usuarios avanzados.

- **DBMS**

Es un sistema de gestión de bases de datos basado en CODASYL, a la que se puede acceder interactivamente y desde los programas.

- **FMS**

Proporciona capacidad para la gestión de informes para lenguajes de programación y Datatrieve. Soporta informes en video (VT100, VT200, etcétera) que son definidos interactivamente y almacenados en una librería de informes FMS.

- **BLISS-32**

Es un lenguaje de desarrollo de lenguajes de alto nivel o, dicho de otro modo, su finalidad es la construcción de compiladores de lenguajes de alto nivel, de procesadores de tiempo real, de utilidades y software para el sistema operativo.

CUESTIONES

1. El sistema operativo VMS, ¿en qué computadoras se encuentra funcionando en la actualidad?
2. ¿Cuáles fueron los motivos de la sustitución de las computadoras PDP-11 por la familia VAX-11?
3. Realizar un esquema general de la estructura de una minicomputadora VAX-11.
4. ¿Para qué tipo de aplicaciones resulta interesante que la computadora que la soporta disponga de alguna ayuda a la unidad aritmético lógica como puede ser el acelerador de operaciones en coma flotante del VAX?
5. Realizar un pequeño esquema de los distintos modelos de la familia VAX.
6. Dibujar esquemáticamente la estructura del sistema operativo VMS.
7. ¿Qué operaciones o trabajos pueden hacerse desde el sistema operativo VMS en modo usuario?
8. ¿Qué sistemas o procesos tienen en el sistema operativo VMS una mayor prioridad de ejecución?
9. En la planificación del procesador del sistema operativo VMS, ¿cómo se gestionan las colas de procesos de igual prioridad?
10. ¿Cuál es el proceso de gestión de memoria utilizado por el sistema operativo VMS?
11. Realizar un esquema de las fases en que se desarrolla el sistema de entrada/salida del sistema operativo VMS.
12. ¿Cuáles son los elementos del sistema de entrada/salida del sistema operativo VMS?
13. ¿Cuáles son las técnicas utilizadas en VMS para la comunicación y sincronización entre procesos?
14. ¿Para qué se utilizan los buzones (Mailboxes) en este sistema operativo?
15. ¿Qué realiza en VMS el intérprete de comandos BCL?
16. Enumerar y comentar las facilidades de ayuda y desarrollo de programas existentes en el sistema operativo VMS.

CAPITULO 17

Sistema Operativo OS/400

17.1. INTRODUCCION

La computadora IBM Application System/400 (AS/400) tiene la tecnología más avanzada entre las computadoras de tipo medio que existen actualmente. Apareció en el mercado a finales de los 80 y representa una notable mejoría en la facilidad de uso e integración del sistema, es, por decirlo de algún modo, el sustituto de la línea de los sistemas 36 y 38. Se relaciona con el sistema 38 por su funcionalidad e incorpora el interfaz de usuario del Sistema 36. El AS/400 respeta, además, el esquema de conectividad System Application Architecture (SAA) de IBM entre grandes computadoras, computadoras de tipo medio y computadoras personales (Figura 17.1).

Figura 17.1. Familia de computadoras AS/400. (Cortesía de IBM.)

El sistema AS/400 tiene una nueva organización en su arquitectura que ha sido diseñada pensando en la conectividad (Figura 17.2).

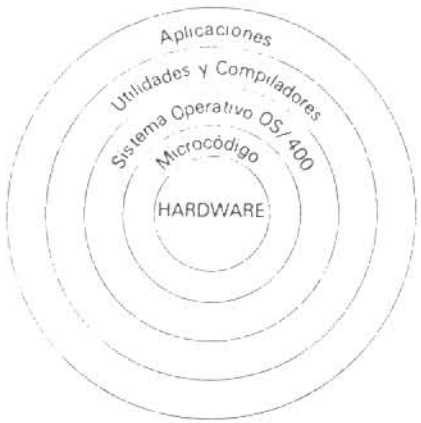


Figura 17.2. Estructura del sistema AS/400.

Está organizado en unidades estructurales y lógicas. A estas unidades se las conoce como objetos, librerías, archivos, miembros y carpetas. Virtualmente, todo el sistema es reconocido como un objeto, como las librerías, los programas, las bases de datos, etc. Las librerías son las estructuras principales de organización del sistema AS/400 (Figura 17.3).

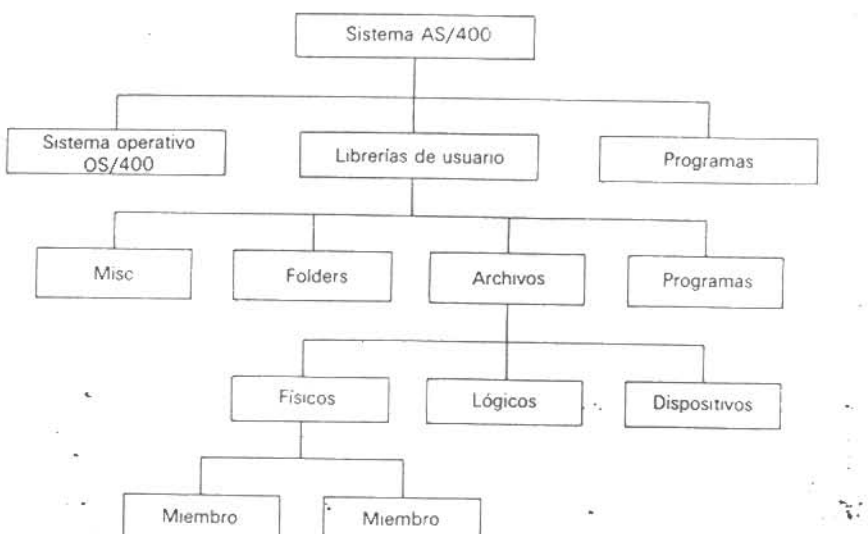


Figura 17.3. Organización del sistema AS/400.

Uno de los principales aspectos del sistema AS/400 es que trata casi todas las entidades de datos como objetos, incluyendo las descripciones de los dispositivos para la utilización de terminales o impresoras, bases de datos, programas y pantallas. El sistema almacena las descripciones de todos los objetos y cuando un usuario nombra un objeto, el sistema lo reconoce y lo busca, obteniendo una completa información de dicho objeto. De ahí que el sistema pueda utilizar un objeto predefinido con la mínima intervención del usuario.

17.2. ESTRUCTURA DEL SISTEMA OPERATIVO OS/400

El sistema operativo nativo de la computadora AS/400 es el Operating System/400 (OS/400) y su principal característica es la sencillez de operación que ofrece al usuario a partir de menús simplificados y ayudas de todo tipo. La figura 17.4 muestra un esquema de la estructura del OS/400.

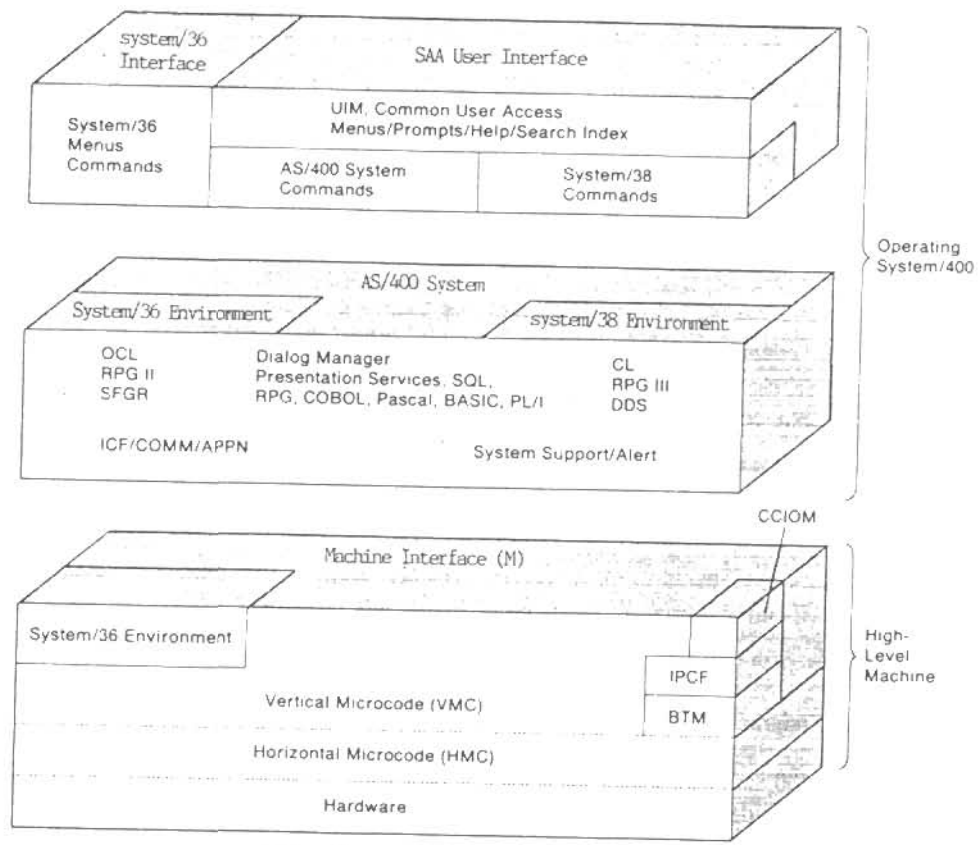


Figura 17.4. Estructura del OS/400. (Cortesía de IBM.)

17.3. CONCEPTOS BASICOS

■ Gestión de trabajos

En el sistema operativo OS/400 los usuarios pueden coordinar el flujo de trabajos para asegurar el máximo aprovechamiento del sistema, que se puede acomodar a las necesidades de los programadores, usuarios, trabajos prioritarios, trabajos batch, etc. El propio sistema se puede ajustar automáticamente para realizar las tareas actuales o, por el contrario, se puede eliminar esta característica y realizarlas el propio usuario. El gestor de trabajos manipula los objetos utilizando el lenguaje de control *(Control Language-CL).*

■ Operaciones automáticas

El AS/400 puede realizar varias operaciones en modo desatendido, de manera que una función o un trabajo pueden ser ejecutados automáticamente sin la intervención del usuario.

■ Interfaz de usuario

El lenguaje de control y la presentación de las pantallas del AS/400 se componen de frases en inglés. Por tanto, un usuario puede conectarse al sistema y realizar cualquier tarea, utilizando los menús correspondientes a cada nivel, y con la ayuda correspondiente a cada uno de ellos (Figura 17.5).

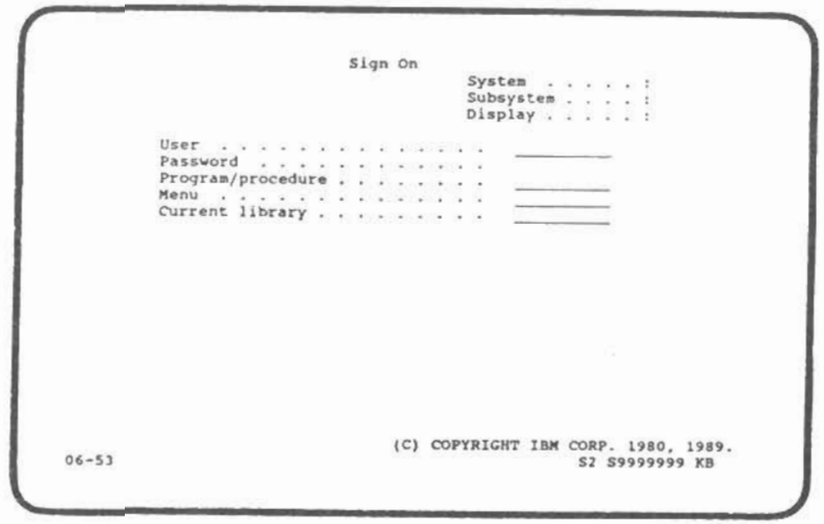
■ Librerías

Las librerías se utilizan para el almacenamiento de datos y contienen todo lo que pueda ser manejable por el sistema. Se asocian en los siguientes grupos de objetos:

- **Librerías del sistema.** Contienen objetos del sistema definidos por IBM como el sistema operativo, compiladores, utilidades y descripciones de otras librerías.
- **Librerías de usuario.** Contienen objetos definidos por el usuario como bases de datos y programas de aplicación.
- **Lista de librerías.** Es una lista de librerías para realizar la búsqueda de un determinado objeto.
- **Archivos.** Es una librería que contiene código fuente o datos.
- **Miembros.** Son datos de un tipo determinado almacenados en un archivo.

■ Emulación y migración de programas.

El sistema AS/400, como ya hemos dicho, es descendiente de los sistemas 36 y 38. Este sistema tiene la capacidad de emular el entorno de los sistemas 36 y 38 completamente. Por ello, los programas compilados en el sistema 38 podrán ser directamente ejecutados en el AS/400, por el contrario, los que pertenezcan al sistema 36 tienen que ser recompilados. Virtualmente, todas las funciones y opciones de los sistemas 36 y 38 son aplicables al AS/400. De todas formas, un código escrito en AS/400 obtendrá mejores resultados que un código que haya sido transportado de otro sistema.



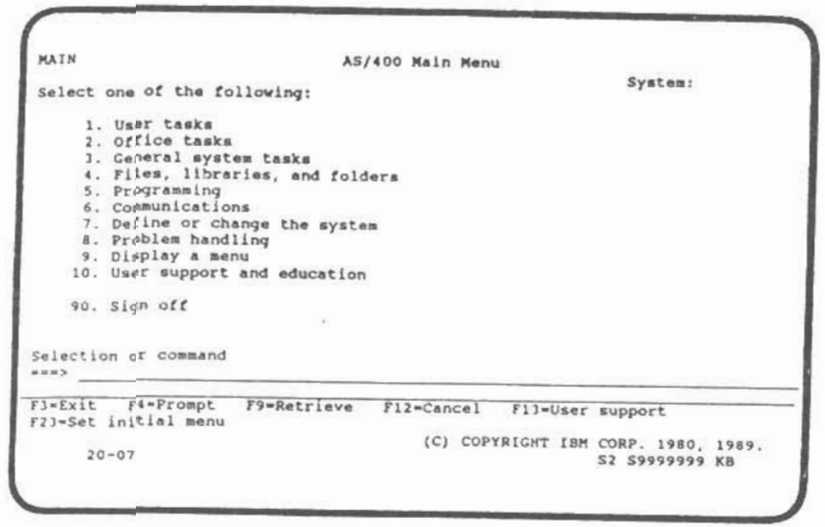


Figura 17.5. Pantalla de inicio y menú principal.

■ **Bases de datos**

La gestión de bases de datos en el sistema AS/400 se hereda directamente del sistema operativo. Esto significa que la mayoría de las llamadas a las bases de datos y funciones se realizan a nivel del sistema operativo.

■ **El lenguaje de control** *(Control Language-CL)*

El lenguaje de control del OS/400 es potente y fácil de aprender. El CL gestiona el sistema operativo mediante comandos en inglés.

■ **Seguridad**

La seguridad del AS/400 es una de las más restringidas del mercado. Está basado en el nivel que tenga el objeto y es necesario que la gestione el administrador del sistema.

17.4. PROGRAMAS DE APLICACION INTEGRADOS

A continuación se muestran los programas integrados que posee el sistema AS/400 para mayor aprovechamiento del potencial del sistema operativo OS/400:

■ **Herramientas para el desarollo de aplicaciones** *(Application Development Tools-ADT)*

El ADT es una herramienta vital para el sistema AS/400. Se trata de una colección de programas integrados al servicio del programador. Se compone de los siguientes:

- **Utilidades de entrada al sistema** *(System Entry Utility-SEU)*. Es el principal método para la edición de código fuente para programas de aplicaciones desarrollados en diferentes lenguajes. Incluye un diccionario para la correción de sintaxis.
- **Diseño de pantallas** *(Screen Design Aid-SDA)*. Facilita el diseño de pantallas y de menús, siendo compiladas automáticamente.
- **Gestor para el desarrollo de programas** *(Programmer Development Manager-PDM)*. Es una aplicación que permite trabajar con librerías y objetos, pudiendo exportar un posible programa a otra librería para que otro usuario pueda utilizarlo.
- **Gráficos** *(Business Graphics Utility-BGU)*. Se trata de una herramienta para el diseño de gráficos.
- **Utilidad de archivos de datos** *(Data File Utility-DFU)*. Facilita la edición y visualización de bases de datos creando programas para las entradas y salidas de usuario.
- **Consulta** *(Query-QRY)*. Al igual que el DFU, permite la creación de programas para el usuario, principalmente para extraer informes mostrando los resultados de varios cálculos, también se puede utilizar para actualizar bases de datos.

■ **Office Visión**

Es el procesador de textos y gestor de tareas del AS/400. Diseñado para la ayuda en el desarrollo de las tareas diarias de oficina. Se puede utilizar para enviar y recibir documentos, consultar números de teléfonos, como agenda, planificar la ejecución de trabajos automáticamente, imprimir informes y realizar diferentes tareas desde un terminal. Para ello cuenta con los siguientes elementos:

- **Calendario.** Las funciones del calendario incluyen citas, horarios y recordatorios y puede ejecutar cualquier programa en un momento determinado fijado previamente.
- **Editor de textos.** El AS/400 tiene la posibilidad de utilizar distintos editores:
  - AS/400 Office.
  - WordPerfect.
  - Display Write.
- **Correo y mensajes.** Consiste en la gestión y envio de mensajes y documentos de un usuario a otro. El correo se puede enviar o recibir mediante una copia impresa o utilizando el sistema de correo electrónico.
- **Documentos.** Se pueden crear documentos con tipos de letras especiales, párrafos o textos condicionales y campos de bases de datos.
- **Gestor de computadoras personales.** Para el control de computadoras personales conectadas.

■ **Lenguajes de alto nivel**

El OS/400 está diseñado para soportar distintos compiladores, montadores y depuradores de lenguajes de alto nivel estándar, de ahí que un programa escrito en una computadora IBM-370, AS/400 o PS/2 podrá ser ejecutado en cualquiera de los otros dos. Los lenguajes de programación utilizados en el AS/400 son los siguientes:

- **RPG/400.** Lenguaje primario de los sistemas 36 y 38, y ahora del AS/400. RPG significa Programa Generador de Informes *(Report Program Generator)* y fue diseñado para aplicaciones de gestión con alto indice de manejo de archivos. Es un lenguaje de difícil lectura y mantenibilidad pero, por el contrario, es escueto y potente.
- **COBOL/400.** La versión de COBOL *(COmmon Business Oriented Language)* que ofrece el AS/400 es muy parecida a las de las computadoras personales. Ya que muchas de las aplicaciones instaladas en computadoras grandes están escritas en este lenguaje, pueden ser transferidas al AS/400 sin ningún problema. El COBOL es un lenguaje de programación fácil de leer y muy difundido, con una gran capacidad en el manejo de archivos y facilidad de diseño de programas. Por el contrario, consume más tiempo que el RPG en el procesamiento de los trabajos.
- **SQL/400.** SQL significa Lenguaje de Consulta Estructurado *(Structured Query Language)*. No es un lenguaje muy difundido y no se ha implementado mucho en grandes computadoras, pero posee una gran capacidad en el manejo de datos, realizando con-

sultas en las bases de datos e incluso puede transformar el código en otro lenguaje (como por ejemplo RPG).

- **C/400.** El lenguaje C es una de las mejores alternativas para el desarrollo de software. Es un lenguaje potente, completo y fácil de leer. Posee entre sus mejores virtudes la estructuración de sus programas por tener una amplia gama de estructuras y permite la utilización de tipos de variables de usuario.
- **Otros.** Existe un conjunto de lenguajes que se encuentran presentes pero que en la actualidad no son muy utilizados en estos equipos. Entre ellos podemos citar PL/1, BASIC y Pascal.

■ Comunicaciones

El sistema AS/400 soporta varios tipos de comunicaciones, desde entradas de trabajos remotas *(Remote Job Entry-RJE)* para aplicaciones del sistema 370, hasta transferencias de y hasta computadoras personales.

- **Soporte.** Las comunicaciones forman parte del concepto básico del sistema. Utilizando el soporte para computadoras personales, el procesamiento conjunto forma un enlace transparente de comunicaciones avanzadas programa a programa *(Advanced Program to Program Communications-APPC)* con el AS/400. Las comunicaciones múltiples del sistema forman un entorno de proceso sofisticado muy fácil de utilizar, que puede ser conectado con cualquier plataforma IBM sin la intervención del usuario.

La Figura 17.6 muestra un ejemplo de conectividad del sistema.

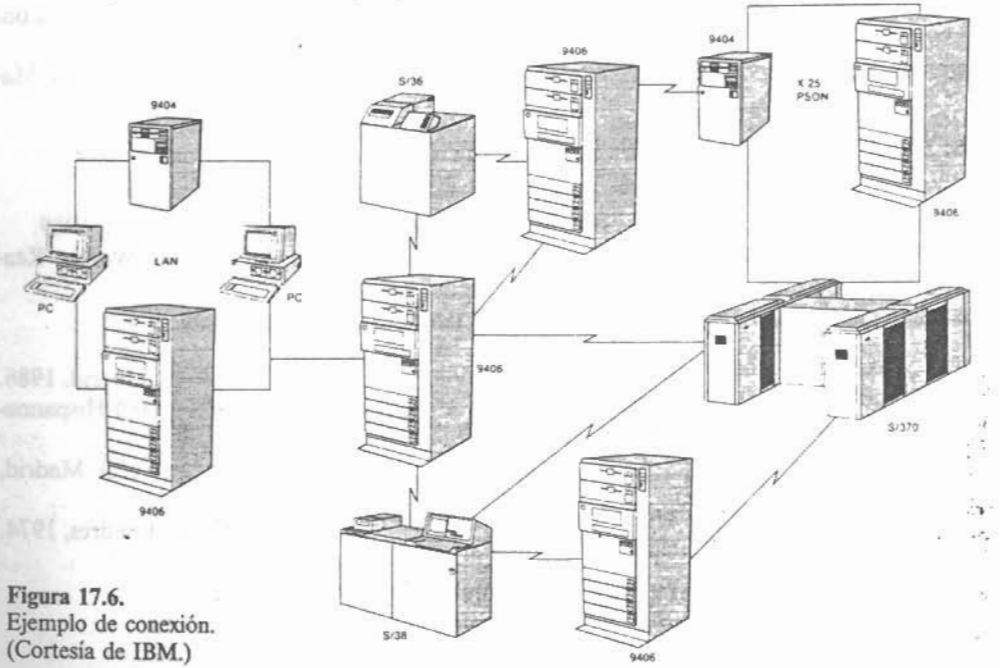


Figura 17.6.
Ejemplo de conexión.
(Cortesía de IBM.)

CUESTIONES

1. ¿Cuáles son las características fundamentales que posee el sistema AS/400?
2. Realizar un esquema de la estructura del AS/400.
3. ¿Con qué nombres se conoce a las unidades estructurales y lógicas sobre las que está organizado el AS/400?
4. Dibujar esquemáticamente la estructura del sistema operativo OS/400.
5. Enumerar las distintas librerías que utiliza para el almacenamiento el sistema OS/400.
6. ¿Es compatible en cuanto a intercambio de software de aplicación el sistema AS/400 con algún otro?
7. Enumerar algunas de las herramientas para el desarrollo de aplicaciones que posee este sistema.
8. ¿Qué posibilidades de utilización de procesadores de texto tiene el AS/400?
9. ¿Qué lenguajes de alto nivel son los más típicos en el AS/400?
10. ¿Qué significados tienen las siglas RPG, COBOL y SQL?
11. El sistema AS/400, ¿es conectable a otros sistemas?
12. ¿Para qué se utiliza el lenguaje de control CL en el sistema operativo OS/400?