yendo la planificación de procesos, el almacenamiento en disco y la gestión de la memoria. Estas consideraciones se abordan a todo lo largo del libro.

En un sistema de tiempo compartido, el sistema operativo debe asegurar un tiempo de respuesta razonable, lo que en ocasiones se hace a través de un mecanismo de **intercambio**, donde los procesos se intercambian entrando y saliendo de la memoria al disco. Un método más habitual de conseguir este objetivo es la **memoria virtual**, una técnica que permite la ejecución de un proceso que no está completamente en memoria (Capítulo 9). La ventaja principal del esquema de memoria virtual es que permite a los usuarios ejecutar programas que sean más grandes que la **memoria física** real. Además, realiza la abstracción de la memoria principal, sustituyéndola desde el punto de vista lógico por una matriz uniforme de almacenamiento de gran tamaño, separando así la **memoria lógica**, como la ve el usuario, de la memoria física. Esta disposición libera a los programadores de preocuparse por las limitaciones de almacenamiento en memoria.

Los sistemas de tiempo compartido también tienen que proporcionar un sistema de archivos (Capítulos 10 y 11). El sistema de archivos reside en una colección de discos; por tanto, deberán proporcionarse también mecanismos de gestión de discos (Capítulo 12). Los sistemas de tiempo compartido también suministran un mecanismo para proteger los recursos frente a usos inapropiados (Capítulo 14). Para asegurar una ejecución ordenada, el sistema debe proporcionar mecanismos para la comunicación y sincronización de trabajos (Capítulo 6) y debe asegurar que los trabajos no darán lugar a interbloqueos que pudieran hacer que quedaran en espera permanentemente (Capítulo 7).

1.5 Operaciones del sistema operativo

Como se ha mencionado anteriormente, los sistemas operativos modernos están **controlados mediante interrupciones**. Si no hay ningún proceso que ejecutar, ningún dispositivo de E/S al que dar servicio y ningún usuario al que responder, un sistema operativo debe permanecer inactivo, esperando a que algo ocurra. Los sucesos casi siempre se indican mediante la ocurrencia de una interrupción o una excepción. Una **excepción** es una interrupción generada por software, debida a un error (por ejemplo, una división por cero o un acceso a memoria no válido) o a una solicitud específica de un programa de usuario de que se realice un servicio del sistema operativo. La característica de un sistema operativo de estar controlado mediante interrupciones define la estructura general de dicho sistema. Para cada tipo de interrupción, diferentes segmentos de código del sistema operativo determinan qué acción hay que llevar a cabo. Se utilizará una rutina de servicio a la interrupción que es responsable de tratarla.

Dado que el sistema operativo y los usuarios comparten los recursos hardware y software del sistema informático, necesitamos asegurar que un error que se produzca en un programa de usuario sólo genere problemas en el programa que se estuviera ejecutando. Con la compartición, muchos procesos podrían verse afectados negativamente por un fallo en otro programa. Por ejemplo, si un proceso entra en un bucle infinito, este bucle podría impedir el correcto funcionamiento de muchos otros procesos. En un sistema de multiprogramación se pueden producir errores más sutiles, pudiendo ocurrir que un programa erróneo modifique otro programa, los datos de otro programa o incluso al propio sistema operativo.

Sin protección frente a este tipo de errores, o la computadora sólo ejecuta un proceso cada vez o todas las salidas deben considerarse sospechosas. Un sistema operativo diseñado apropiadamente debe asegurar que un programa incorrecto (o malicioso) no pueda dar lugar a que otros programas se ejecuten incorrectamente.

1.5.1 Operación en modo dual

Para asegurar la correcta ejecución del sistema operativo, tenemos que poder distinguir entre la ejecución del código del sistema operativo y del código definido por el usuario. El método que usan la mayoría de los sistemas informáticos consiste en proporcionar soporte hardware que nos permita diferenciar entre varios modos de ejecución.

Como mínimo, necesitamos dos **modos** diferentes de operación: **modo usuario** y **modo *kernel*** (también denominado **modo de supervisor**, **modo del sistema** o **modo privilegiado**). Un bit, denominado **bit de modo**, se añade al hardware de la computadora para indicar el modo actual: *kernel* (0) o usuario (1). Con el bit de modo podemos diferenciar entre una tarea que se ejecute en nombre del sistema operativo y otra que se ejecute en nombre del usuario. Cuando el sistema informático está ejecutando una aplicación de usuario, el sistema se encuentra en modo de usuario. Sin embargo, cuando una aplicación de usuario solicita un servicio del sistema operativo (a través de una llamada al sistema), debe pasar del modo de usuario al modo *kernel* para satisfacer la solicitud. Esto se muestra en la Figura 1.8. Como veremos, esta mejora en la arquitectura resulta útil también para muchos otros aspectos del sistema operativo.

Cuando se arranca el sistema, el hardware se inicia en el modo *kernel*. El sistema operativo se carga y se inician las aplicaciones de usuario en el modo usuario. Cuando se produce una excepción o interrupción, el hardware conmuta del modo de usuario al modo *kernel* (es decir, cambia el estado del bit de modo a 0). En consecuencia, cuando el sistema operativo obtiene el control de la computadora, estará en el modo *kernel*. El sistema siempre cambia al modo de usuario (poniendo el bit de modo a 1) antes de pasar el control a un programa de usuario.

El modo dual de operación nos proporciona los medios para proteger el sistema operativo de los usuarios que puedan causar errores, y también para proteger a los usuarios de los errores de otros usuarios. Esta protección se consigue designando algunas de las instrucciones de máquina que pueden causar daño como **instrucciones privilegiadas**. El hardware hace que las instrucciones privilegiadas sólo se ejecuten en el modo *kernel*. Si se hace un intento de ejecutar una instrucción privilegiada en modo de usuario, el hardware no ejecuta la instrucción sino que la trata como ilegal y envía una excepción al sistema operativo.

La instrucción para conmutar al modo usuario es un ejemplo de instrucción privilegiada. Entre otros ejemplos se incluyen el control de E/S, la gestión del temporizador y la gestión de interrupciones. A medida que avancemos a lo largo del texto, veremos que hay muchas instrucciones privilegiadas adicionales.

Ahora podemos ver el ciclo de vida de la ejecución de una instrucción en un sistema informático. El control se encuentra inicialmente en manos del sistema operativo, donde las instrucciones se ejecutan en el modo *kernel*. Cuando se da el control a una aplicación de usuario, se pasa a modo usuario. Finalmente, el control se devuelve al sistema operativo a través de una interrupción, una excepción o una llamada al sistema.

Las llamadas al sistema proporcionan los medios para que un programa de usuario pida al sistema operativo que realice tareas reservadas del sistema operativo en nombre del programa del usuario. Una llamada al sistema se invoca de diversas maneras, dependiendo de la funcionalidad proporcionada por el procesador subyacente. En todas sus formas, se trata de un método usado por un proceso para solicitar la actuación del sistema operativo. Normalmente, una llamada al sistema toma la forma de una excepción que efectúa una transferencia a una posición específica en el vector de interrupción. Esta excepción puede ser ejecutada mediante una instrucción genérica `trap`, aunque algunos sistemas (como la familia MIPS R2000) tienen una instrucción `syscall` específica.

Cuando se ejecuta una llamada al sistema, el hardware la trata como una interrupción software. El control pasa a través del vector de interrupción a una rutina de servicio del sistema opera-

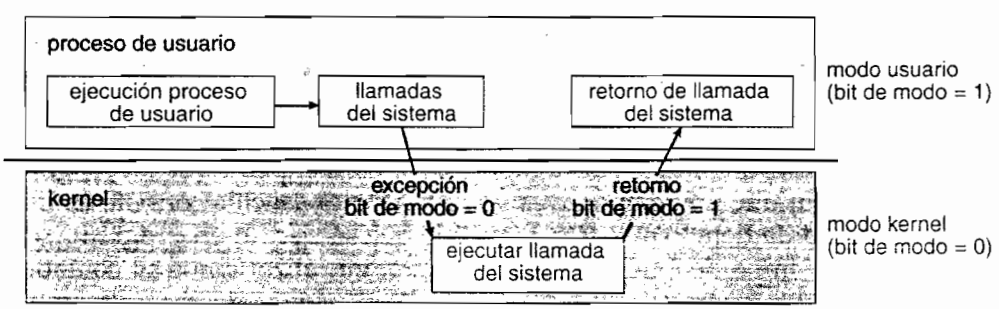

Figura 1.8 Transición del modo usuario al modo *kernel*.

tivo, y el bit de modo se establece en el modo *kernel*. La rutina de servicio de la llamada al sistema es una parte del sistema operativo. El *kernel* examina la instrucción que interrumpe para determinar qué llamada al sistema se ha producido; un parámetro indica qué tipo de servicio está requiriendo el programa del usuario. Puede pasarse la información adicional necesaria para la solicitud mediante registros, mediante la pila o mediante la memoria (pasando en los registros una serie de punteros a posiciones de memoria). El *kernel* verifica que los parámetros son correctos y legales, ejecuta la solicitud y devuelve el control a la instrucción siguiente a la de la llamada de servicio. En la Sección 2.3 se describen de forma más completa las llamadas al sistema.

La falta de un modo dual soportado por hardware puede dar lugar a serios defectos en un sistema operativo. Por ejemplo, MS-DOS fue escrito para la arquitectura 8088 de Intel, que no dispone de bit de modo y, por tanto, tampoco del modo dual. Un programa de usuario que se ejecute erróneamente puede corromper el sistema operativo sobreescribiendo sus archivos; y múltiples programas podrían escribir en un dispositivo al mismo tiempo, con resultados posiblemente desastrosos. Las versiones recientes de la CPU de Intel, como por ejemplo Pentium, sí que proporcionan operación en modo dual. De acuerdo con esto, la mayoría de los sistemas operativos actuales, como Microsoft Windows 2000, Windows XP, Linux y Solaris para los sistemas x86, se aprovechan de esta característica y proporcionan una protección mucho mayor al sistema operativo.

Una vez que se dispone de la protección hardware, el hardware detecta los errores de violación de los modos. Normalmente, el sistema operativo se encarga de tratar estos errores. Si un programa de usuario falla de alguna forma, como por ejemplo haciendo un intento de ejecutar una instrucción ilegal o de acceder a una zona se memoria que no esté en el espacio de memoria del usuario, entonces el hardware envía una excepción al sistema operativo. La excepción transfiere el control al sistema operativo a través del vector de interrupción, igual que cuando se produce una interrupción. Cuando se produce un error de programa, el sistema operativo debe terminar el programa anormalmente. Esta situación se trata con el mismo código software que se usa en una terminación anormal solicitada por el usuario. El sistema proporciona un mensaje de error apropiado y puede volcarse la memoria del programa. Normalmente, el volcado de memoria se escribe en un archivo con el fin de que el usuario o programador puedan examinarlo y quizá corregir y reiniciar el programa.

1.5.2 Temporizador

Debemos asegurar que el sistema operativo mantenga el control sobre la CPU. Por ejemplo, debemos impedir que un programa de usuario entre en un bucle infinito o que no llame a los servicios del sistema y nunca devuelva el control al sistema operativo. Para alcanzar este objetivo, podemos usar un **temporizador**. Puede configurarse un temporizador para interrumpir a la computadora después de un período especificado. El período puede ser fijo (por ejemplo, 1/60 segundos) o variable (por ejemplo, entre 1 milisegundo y 1 segundo). Generalmente, se implementa un **temporizador variable** mediante un reloj de frecuencia fija y un contador. El sistema operativo configura el contador. Cada vez que el reloj avanza, el contador se decrementa. Cuando el contador alcanza el valor 0, se produce una interrupción. Por ejemplo, un contador de 10 bits con un reloj de 1 milisegundo permite interrupciones a intervalos de entre 1 milisegundo y 1.024 milisegundos, en pasos de 1 milisegundo.

Antes de devolver el control al usuario, el sistema operativo se asegura de que el temporizador esté configurado para realizar interrupciones. Cuando el temporizador interrumpe, el control se transfiere automáticamente al sistema operativo, que puede tratar la interrupción como un error fatal o puede conceder más tiempo al programa. Evidentemente, las instrucciones que modifican el contenido del temporizador son instrucciones privilegiadas.

Por tanto, podemos usar el temporizador para impedir que un programa de usuario se esté ejecutando durante un tiempo excesivo. Una técnica sencilla consiste en inicializar un contador con la cantidad de tiempo que esté permitido que se ejecute un programa. Para un programa con un límite de tiempo de 7 minutos, por ejemplo, inicializaríamos su contador con el valor 420. Cada segundo, el temporizador interrumpe y el contador se decrementa en una unidad. Mientras que el valor del contador sea positivo, el control se devuelve al programa de usuario. Cuando el valor

del contador pasa a ser negativo, el sistema operativo termina el programa, por haber sido excedido el límite de tiempo asignado.

1.6 Gestión de procesos

Un programa no hace nada a menos que una CPU ejecute sus instrucciones. Un programa en ejecución, como ya hemos dicho, es un proceso. Un programa de usuario de tiempo compartido, como por ejemplo un compilador, es un proceso. Un procesador de textos que ejecute un usuario individual en un PC es un proceso. Una tarea del sistema, como enviar datos de salida a una impresora, también puede ser un proceso (o al menos, una parte de un proceso). Por el momento, vamos a considerar que un proceso es un trabajo o un programa en tiempo compartido, aunque más adelante veremos que el concepto es más general. Como veremos en el Capítulo 3, es posible proporcionar llamadas al sistema que permitan a los procesos crear subprocesos que se ejecuten de forma concurrente.

Un proceso necesita para llevar a cabo su tarea ciertos recursos, entre los que incluyen tiempo de CPU, memoria, archivos y dispositivos de E/S. Estos recursos se proporcionan al proceso en el momento de crearlo o se le asignan mientras se está ejecutando. Además de los diversos recursos físicos y lógicos que un proceso obtiene en el momento de su creación, pueden pasársele diversos datos de inicialización (entradas). Por ejemplo, considere un proceso cuya función sea la de mostrar el estado de un archivo en la pantalla de un terminal. A ese proceso le proporcionaríamos como entrada el nombre del archivo y el proceso ejecutaría las apropiadas instrucciones y llamadas al sistema para obtener y mostrar en el terminal la información deseada. Cuando el proceso termina, el sistema operativo reclama todos los recursos reutilizables.

Hagamos hincapié en que un programa por sí solo no es un proceso; un programa es una entidad *pasiva*, tal como los contenidos de un archivo almacenado en disco, mientras que un proceso es una entidad *activa*. Un proceso de una sola hebra tiene un **contador de programa** que especifica la siguiente instrucción que hay que ejecutar, (las hebras se verán en el Capítulo 4). La ejecución de un proceso así debe ser secuencial: la CPU ejecuta una instrucción del proceso después de otra, hasta completarlo. Además, en cualquier instante, se estará ejecutando como mucho una instrucción en nombre del proceso. Por tanto, aunque pueda haber dos procesos asociados con el mismo programa, se considerarían no obstante como dos secuencias de ejecución separadas. Un proceso multihebra tiene múltiples contadores de programa, apuntado cada uno de ellos a la siguiente instrucción que haya que ejecutar para una hebra determinada.

Un proceso es una unidad de trabajo en un sistema. Cada sistema consta de una colección de procesos, siendo algunos de ellos procesos del sistema operativo (aquéllos que ejecutan código del sistema) y el resto procesos de usuario (aquéllos que ejecutan código del usuario). Todos estos procesos pueden, potencialmente, ejecutarse de forma concurrente, por ejemplo multiplexando la CPU cuando sólo se disponga de una.

El sistema operativo es responsable de las siguientes actividades en lo que se refiere a la gestión de procesos:

- Crear y borrar los procesos de usuario y del sistema.
- Suspender y reanudar los procesos.
- Proporcionar mecanismos para la sincronización de procesos.
- Proporcionar mecanismos para la comunicación entre procesos.
- Proporcionar mecanismos para el tratamiento de los interbloqueos.

En los Capítulos 3 a 6 se estudian las técnicas de gestión de procesos.

1.7 Gestión de memoria

Como se ha visto en la Sección 1.2.2, la memoria principal es fundamental en la operación de un sistema informático moderno. La memoria principal es una matriz de palabras o bytes cuyo tama-

ño se encuentra en el rango de cientos de miles a miles de millones de posiciones distintas. Cada palabra o byte tiene su propia dirección. La memoria principal es un repositorio de datos rápidamente accesibles, compartida por la CPU y los dispositivos de E/S. El procesador central lee las instrucciones de la memoria principal durante el ciclo de extracción de instrucciones y lee y escribe datos en la memoria principal durante el ciclo de extracción de datos (en una arquitectura Von Neumann). La memoria principal es, generalmente, el único dispositivo de almacenamiento de gran tamaño al que la CPU puede dirigirse y acceder directamente. Por ejemplo, para que la CPU procese datos de un disco, dichos datos deben transferirse en primer lugar a la memoria principal mediante llamadas de E/S generadas por la CPU. Del mismo modo, las instrucciones deben estar en memoria para que la CPU las ejecute.

Para que un programa pueda ejecutarse, debe estar asignado a direcciones absolutas y cargado en memoria. Mientras el programa se está ejecutando, accede a las instrucciones y a los datos de la memoria generando dichas direcciones absolutas. Finalmente, el programa termina, su espacio de memoria se declara disponible y el siguiente programa puede ser cargado y ejecutado.

Para mejorar tanto la utilización de la CPU como la velocidad de respuesta de la computadora frente a los usuarios, las computadoras de propósito general pueden mantener varios programas en memoria, lo que crea la necesidad de mecanismos de gestión de la memoria. Se utilizan muchos esquemas diferentes de gestión de la memoria. Estos esquemas utilizan enfoques distintos y la efectividad de cualquier algoritmo dado depende de la situación. En la selección de un esquema de gestión de memoria para un sistema específico, debemos tener en cuenta muchos factores, especialmente relativos al diseño *hardware* del sistema. Cada algoritmo requiere su propio soporte hardware.

El sistema operativo es responsable de las siguientes actividades en lo que se refiere a la gestión de memoria:

- Controlar qué partes de la memoria están actualmente en uso y por parte de quién.
- Decidir qué datos y procesos (o partes de procesos) añadir o extraer de la memoria.
- Asignar y liberar la asignación de espacio de memoria según sea necesario.

En los Capítulos 8 y 9 se estudian las técnicas de gestión de la memoria.

1.8 Gestión de almacenamiento

Para que el sistema informático sea cómodo para los usuarios, el sistema operativo proporciona una vista lógica y uniforme del sistema de almacenamiento de la información. El sistema operativo abstrae las propiedades físicas de los dispositivos de almacenamiento y define una unidad de almacenamiento lógico, el **archivo**. El sistema operativo asigna los archivos a los soportes físicos y accede a dichos archivos a través de los dispositivos de almacenamiento.

1.8.1 Gestión del sistema de archivos

La gestión de archivos es uno de los componentes más visibles de un sistema operativo. Las computadoras pueden almacenar la información en diferentes tipos de medios físicos. Los discos magnéticos, discos ópticos y cintas magnéticas son los más habituales. Cada uno de estos medios tiene sus propias características y organización física. Cada medio se controla mediante un dispositivo, tal como una unidad de disco o una unidad de cinta, que también tiene sus propias características distintivas. Estas propiedades incluyen la velocidad de acceso, la capacidad, la velocidad de transferencia de datos y el método de acceso (secuencial o aleatorio).

Un archivo es una colección de información relacionada definida por su creador. Comúnmente, los archivos representan programas (tanto en formato fuente como objeto) y datos. Los archivos de datos pueden ser numéricos, alfabéticos, alfanuméricos o binarios. Los archivos pueden tener un formato libre (como, por ejemplo, los archivos de texto) o un formato rígido, como por ejemplo una serie de campos fijos. Evidentemente, el concepto de archivo es extremadamente general.

El sistema operativo implementa el abstracto concepto de archivo gestionando los medios de almacenamiento masivos, como las cintas y discos, y los dispositivos que los controlan. Asimismo, los archivos normalmente se organizan en directorios para hacer más fácil su uso. Por último, cuando varios usuarios tienen acceso a los archivos, puede ser deseable controlar quién y en qué forma (por ejemplo, lectura, escritura o modificación) accede a los archivos.

El sistema operativo es responsable de las siguientes actividades en lo que se refiere a la gestión de archivos:

- Creación y borrado de archivos.
- Creación y borrado de directorios para organizar los archivos.
- Soporte de primitivas para manipular archivos y directorios.
- Asignación de archivos a los dispositivos de almacenamiento secundario.
- Copia de seguridad de los archivos en medios de almacenamiento estables (no volátiles).

Las técnicas de gestión de archivos se tratan en los Capítulos 10 y 11.

1.8.2 Gestión del almacenamiento masivo

Como ya hemos visto, dado que la memoria principal es demasiado pequeña para acomodar todos los datos y programas, y puesto que los datos que guarda se pierden al desconectar la alimentación, el sistema informático debe proporcionar un almacenamiento secundario como respaldo de la memoria principal. La mayoría de los sistemas informáticos modernos usan discos como principal medio de almacenamiento en línea, tanto para los programas como para los datos. La mayor parte de los programas, incluyendo compiladores, ensambladores o procesadores de texto, se almacenan en un disco hasta que se cargan en memoria, y luego usan el disco como origen y destino de su procesamiento. Por tanto, la apropiada gestión del almacenamiento en disco tiene una importancia crucial en un sistema informático. El sistema operativo es responsable de las siguientes actividades en lo que se refiere a la gestión de disco:

- Gestión del espacio libre.
- Asignación del espacio de almacenamiento.
- Planificación del disco.

Dado que el almacenamiento secundario se usa con frecuencia, debe emplearse de forma eficiente. La velocidad final de operación de una computadora puede depender de las velocidades del subsistema de disco y de los algoritmos que manipulan dicho subsistema.

Sin embargo, hay muchos usos para otros sistemas de almacenamiento más lentos y más baratos (y que, en ocasiones, proporcionan una mayor capacidad) que el almacenamiento secundario. La realización de copias de seguridad de los datos del disco, el almacenamiento de datos raramente utilizados y el almacenamiento definitivo a largo plazo son algunos ejemplos.

Las unidades de cinta magnética y sus cintas y las unidades de CD y DVD y sus discos son dispositivos típicos de **almacenamiento terciario**. Los soportes físicos (cintas y discos ópticos) varían entre los formatos **WORM** (write-once, read-many-times; escritura una vez, lectura muchas veces) y **RW** (read-write, lectura-escritura).

El almacenamiento terciario no es crucial para el rendimiento del sistema, aunque también es necesario gestionarlo. Algunos sistemas operativos realizan esta tarea, mientras que otros dejan el control del almacenamiento terciario a los programas de aplicación. Algunas de las funciones que dichos sistemas operativos pueden proporcionar son el montaje y desmontaje de medios en los dispositivos, la asignación y liberación de dispositivos para su uso exclusivo por los procesos, y la migración de datos del almacenamiento secundario al terciario.

Las técnicas para la gestión del almacenamiento secundario y terciario se estudian en el Capítulo 12.

1.8.3 Almacenamiento en caché

El **almacenamiento en caché** es una técnica importante en los sistemas informáticos. Normalmente, la información se mantiene en algún sistema de almacenamiento, como por ejemplo la memoria principal. Cuando se usa, esa información se copia de forma temporal en un sistema de almacenamiento más rápido, la caché. Cuando necesitamos una información particular, primero comprobamos si está en la caché. Si lo está, usamos directamente dicha información de la caché; en caso contrario, utilizamos la información original, colocando una copia en la caché bajo la suposición de que pronto la necesitaremos nuevamente.

Además, los registros programables internos, como los registros de índice, proporcionan una caché de alta velocidad para la memoria principal. El programador (o compilador) implementa los algoritmos de asignación de recursos y de reemplazamiento de registros para decidir qué información mantener en los registros y cuál en la memoria principal. También hay disponibles cachés que se implementan totalmente mediante hardware. Por ejemplo, la mayoría de los sistemas disponen de una caché de instrucciones para almacenar las siguientes instrucciones en espera de ser ejecutadas. Sin esta caché, la CPU tendría que esperar varios ciclos mientras las instrucciones son extraídas de la memoria principal. Por razones similares, la mayoría de los sistemas disponen de una o más cachés de datos de alta velocidad en la jerarquía de memoria. En este libro no vamos a ocuparnos de estás cachés implementadas totalmente mediante hardware, ya que quedan fuera del control del sistema operativo.

Dado que las cachés tienen un tamaño limitado, la **gestión de la caché** es un problema de diseño importante. La selección cuidadosa del tamaño de la caché y de una adecuada política de reemplazamiento puede dar como un resultado un incremento enorme del rendimiento. Consulte la Figura 1.9 para ver una comparativa de las prestaciones de almacenamiento en las estaciones de trabajo grandes y pequeños servidores; dicha comparativa ilustra perfectamente la necesidad de usar el almacenamiento en caché. En el Capítulo 9 se exponen varios algoritmos de reemplazamiento para cachés controladas por software.

La memoria principal puede verse como una caché rápida para el almacenamiento secundario, ya que los datos en almacenamiento secundario deben copiarse en la memoria principal para poder ser utilizados, y los datos deben encontrarse en la memoria principal antes de ser pasados al almacenamiento secundario cuando llega el momento de guardarlos. Los datos del sistema de archivos, que residen permanentemente en el almacenamiento secundario, pueden aparecer en varios niveles de la jerarquía de almacenamiento. En el nivel superior, el sistema operativo puede mantener una caché de datos del sistema de archivos en la memoria principal. También pueden utilizarse discos RAM (también conocidos como **discos de estado sólido**) para almacenamiento de alta velocidad, accediendo a dichos discos a través de la interfaz del sistema de archivos. La mayor parte del almacenamiento secundario se hace en discos magnéticos. Los datos almacenados en disco magnético, a su vez, se copian a menudo en cintas magnéticas o discos extraíbles con el fin de protegerlos frente a pérdidas de datos en caso de que un disco duro falle. Algunos sistemas realizan automáticamente un archivado definitivo de los datos correspondientes a los archivos antiguos, transfiriéndolos desde el almacenamiento secundario a un almacenamiento terciario, como por ejemplo un servidor de cintas, con el fin de reducir costes de almacenamiento (véase el Capítulo 12).

El movimiento de información entre niveles de una jerarquía de almacenamiento puede ser explícito o implícito, dependiendo del diseño hardware y del software del sistema operativo que controle dicha funcionalidad. Por ejemplo, la transferencia de datos de la caché a la CPU y los registros es, normalmente, una función hardware en la que no interviene el sistema operativo. Por el contrario, la transferencia de datos de disco a memoria normalmente es controlada por el sistema operativo.

En una estructura de almacenamiento jerárquica, los mismos datos pueden aparecer en diferentes niveles del sistema de almacenamiento. Por ejemplo, suponga que un entero A que hay que incrementar en 1 se encuentra almacenado en el archivo B, el cual reside en un disco magnético. La operación de incremento ejecuta primero una operación de E/S para copiar el bloque de disco en el que reside A a la memoria principal. A continuación se copia A en la caché y en un registro

Nivel	1	2	3	4
Nombre	registros	caché	memoria principal	almacenamiento en disco
Tamaño típico	< 1 KB	> 16 MB	> 16 GB	> 100 GB
Tecnología de implementación	memoria custom con múltiples puertos, CMOS	SRAM CMOS on-chip u off-chip	CMOS DRAM	disco magnético
Tiempo de acceso (ns)	0,25 – 0,5	0,5 – 25	80 – 250	5.000.000
Ancho de banda (MB/s)	20.000 – 100.000	5000 – 10.000	1000 – 5000	20 – 150
Gestionado por	compilador	hardware	sistema operativo	sistema operativo
Copiado en	caché	memoria principal	disco	CD o cinta

Figura 1.9. Prestaciones de los distintos niveles de almacenamiento.

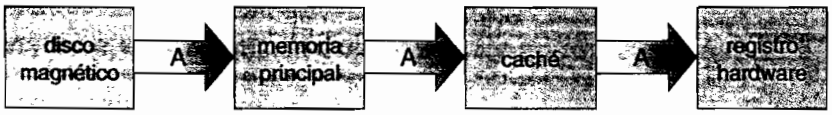

Figura 1.10 Migración de un entero A del disco a un registro.

interno. Por tanto, la copia de A aparece en varios lugares: en el disco magnético, en la memoria-principal, en la caché y en un registro interno (véase la Figura 1.10). Una vez que se hace el incremento en el registro interno, el valor de A es distinto en los diversos sistemas de almacenamiento. El valor de A sólo será el mismo después de que su nuevo valor se escriba desde el registro interno al disco magnético.

En un entorno informático donde sólo se ejecuta un proceso cada vez, este proceso no plantea ninguna dificultad, ya que un acceso al entero A siempre se realizará a la copia situada en el nivel más alto de la jerarquía. Sin embargo, en un entorno multitarea, en el que la CPU conmuta entre varios procesos, hay que tener un extremo cuidado para asegurar que, si varios procesos desean acceder a A, cada uno de ellos obtenga el valor más recientemente actualizado de A.

La situación se hace más complicada en un entorno multiprocesador donde, además de mantener registros internos, cada una de las CPU también contiene una caché local. En un entorno de este tipo, una copia de A puede encontrarse simultáneamente en varias cachés. Dado que las diversas CPU puede ejecutar instrucciones concurrentemente, debemos asegurarnos de que una actualización del valor de A en una caché se refleje inmediatamente en las restantes cachés en las que reside A. Esta situación se denomina **coherencia de caché** y, normalmente, se trata de un problema de hardware, que se gestiona por debajo del nivel del sistema operativo.

En un entorno distribuido, la situación se hace incluso más compleja. En este tipo de entorno, varias copias (o réplicas) del mismo archivo pueden estar en diferentes computadoras distribuidas geográficamente. Dado que se puede acceder de forma concurrente a dichas réplicas y actualizarlas, algunos sistemas distribuidos garantizan que, cuando una réplica se actualiza en un sitio, todas las demás réplicas se actualizan lo más rápidamente posible. Existen varias formas de proporcionar esta garantía, como se verá en el Capítulo 17.

1.8.4 Sistemas de E/S

Uno de los propósitos de un sistema operativo es ocultar al usuario las peculiaridades de los dispositivos hardware específicos. Por ejemplo, en UNIX, las peculiaridades de los dispositivos de E/S se ocultan a la mayor parte del propio sistema operativo mediante el **subsistema** de E/S. El subsistema de E/S consta de varios componentes:

- Un componente de gestión de memoria que incluye almacenamiento en búfer, gestión de caché y gestión de colas.
- Una interfaz general para controladores de dispositivo.

- Controladores para dispositivos hardware específicos.

Sólo el controlador del dispositivo conoce las peculiaridades del dispositivo específico al que está asignado.

En la Sección 1.2.3 se expone cómo se usan las rutinas de tratamiento de interrupciones y los controladores de dispositivo en la construcción de subsistemas de E/S eficientes. En el Capítulo 13 se aborda el modo en que el subsistema de E/S interactúa con los otros componentes del sistema, gestiona los dispositivos, transfiere datos y detecta que las operaciones de E/S han concluido.

1.9 Protección y seguridad

Si un sistema informático tiene múltiples usuarios y permite la ejecución concurrente de múltiples procesos, entonces el acceso a los datos debe regularse. Para dicho propósito, se emplean mecanismos que aseguren que sólo puedan utilizar los recursos (archivos, segmentos de memoria, CPU y otros) aquellos procesos que hayan obtenido la apropiada autorización del sistema operativo. Por ejemplo, el hardware de direccionamiento de memoria asegura que un proceso sólo se pueda ejecutar dentro de su propio espacio de memoria; el temporizador asegura que ningún proceso pueda obtener el control de la CPU sin después ceder el control; los usuarios no pueden acceder a los registros de control, por lo que la integridad de los diversos dispositivos periféricos está protegida, etc.

Por tanto, **protección** es cualquier mecanismo que controle el acceso de procesos y usuarios a los recursos definidos por un sistema informático. Este mecanismo debe proporcionar los medios para la especificación de los controles que hay que imponer y para la aplicación de dichos controles.

Los mecanismos de protección pueden mejorar la fiabilidad, permitiendo detectar errores latentes en las interfaces entre los subsistemas componentes. La detección temprana de los errores de interfaz a menudo puede evitar la contaminación de un subsistema que funciona perfectamente por parte de otro subsistema que funcione mal. Un recurso desprotegido no puede defenderse contra el uso (o mal uso) de un usuario no autorizado o incompetente. Un sistema orientado a la protección proporciona un medio para distinguir entre un uso autorizado y no autorizado, como se explica en el Capítulo 14.

Un sistema puede tener la protección adecuada pero estar expuesto a fallos y permitir accesos inapropiados. Considere un usuario al que le han robado su información de autenticación (los medios de identificarse ante el sistema); sus datos podrían ser copiados o borrados, incluso aunque esté funcionando la protección de archivos y de memoria. Es responsabilidad de los mecanismos de **seguridad** defender al sistema frente a ataques internos y externos. Tales ataques abarcan un enorme rango, en el que se incluyen los virus y gusanos, los ataques de denegación de servicio (que usan todos los recursos del sistema y mantienen a los usuarios legítimos fuera del sistema), el robo de identidad y el robo de servicio (el uso no autorizado de un sistema). La prevención de algunos de estos ataques se considera una función del sistema operativo en algunos sistemas, mientras que en otros se deja a la política de prevención o a algún software adicional. Debido a la creciente alarma en lo que se refiere a incidentes de seguridad, las características de seguridad del sistema operativo constituyen un área de investigación e implementación en rápido crecimiento. Los temas relativos a la seguridad se estudian en el Capítulo 15.

La protección y la seguridad requieren que el sistema pueda distinguir a todos sus usuarios. La mayoría de los sistemas operativos mantienen una lista con los nombres de usuario y sus **identificadores de usuario** (ID) asociados. En la jerga de Windows NT, esto se denomina ID **de seguridad (SID, security ID).** Estos ID numéricos son unívocos, uno por usuario. Cuando un usuario inicia una sesión en el sistema, la fase de autenticación determina el ID correspondiente a dicho usuario. Ese ID de usuario estará asociado con todos los procesos y hebras del usuario. Cuando un ID necesita poder ser leído por los usuarios, se utiliza la lista de nombres de usuario para traducir el ID al nombre correspondiente.

En algunas circunstancias, es deseable diferenciar entre conjuntos de usuarios en lugar de entre usuarios individuales. Por ejemplo, el propietario de un archivo en un sistema UNIX puede ejecu-

tar todas las operaciones sobre dicho archivo, mientras que a un conjunto seleccionado de usuarios podría permitírsele sólo leer el archivo. Para conseguir esto, es necesario definir un nombre de grupo y especificar los usuarios que pertenezcan a dicho grupo. La funcionalidad de grupo se puede implementar manteniendo en el sistema una lista de nombres de grupo e **identificadores de grupo**. Un usuario puede pertenecer a uno o más grupos, dependiendo de las decisiones de diseño del sistema operativo. Los ID de grupo del usuario también se incluyen en todos los procesos y hebras asociados.

Durante el uso normal de un sistema, el ID de usuario y el ID de grupo de un usuario son suficientes. Sin embargo, en ocasiones, un usuario necesita **escalar sus privilegios** para obtener permisos adicionales para una actividad. Por ejemplo, el usuario puede necesitar acceder a un dispositivo que está restringido. Los sistemas operativos proporcionan varios métodos para el escalado de privilegios. Por ejemplo, en UNIX, el atributo `setuid` en un programa hace que dicho programa se ejecute con el ID de usuario del propietario del archivo, en lugar de con el ID del usuario actual. El proceso se ejecuta con este **UID efectivo** hasta que se desactivan los privilegios adicionales o se termina el proceso. Veamos un ejemplo de cómo se hace esto en Solaris 10: el usuario pbg podría tener un ID de usuario igual a 101 y un ID de grupo igual a 14, los cuales se asignarían mediante `/etc/passwd:pbg:x:101:14::/export/home/pbg:/usr/bin/bash`.

1.10 Sistemas distribuidos

Un sistema distribuido es una colección de computadoras físicamente separadas y posiblemente heterogéneas que están conectadas en red para proporcionar a los usuarios acceso a los diversos recursos que el sistema mantiene. Acceder a un recurso compartido incrementa la velocidad de cálculo, la funcionalidad, la disponibilidad de los datos y la fiabilidad. Algunos sistemas operativos generalizan el acceso a red como una forma de acceso a archivo, manteniendo los detalles de la conexión de red en el controlador de dispositivo de la interfaz de red. Otros sistemas operativos invocan específicamente una serie de funciones de red. Generalmente, los sistemas contienen una mezcla de los dos modos, como por ejemplo FTP y NFS. Los protocolos que forman un sistema distribuido pueden afectar enormemente a la popularidad y utilidad de dicho sistema.

En términos sencillos, una **red** es una vía de comunicación entre dos o más sistemas. La funcionalidad de los sistemas distribuidos depende de la red. Las redes varían según el protocolo que usen, las distancias entre los nodos y el medio de transporte. TCP/IP es el protocolo de red más común, aunque el uso de ATM y otros protocolos está bastante extendido. Asimismo, los protocolos soportados varían de unos sistemas operativos a otros. La mayoría de los sistemas operativos soportan TCP/IP, incluidos los sistemas operativos Windows y UNIX. Algunos sistemas soportan protocolos propietarios para ajustarse a sus necesidades. En un sistema operativo, un protocolo de red simplemente necesita un dispositivo de interfaz (por ejemplo, un adaptador de red), con un controlador de dispositivo que lo gestione y un software para tratar los datos. Estos conceptos se explican a lo largo del libro.

Las redes se caracterizan en función de las distancias entre sus nodos. Una **red de área local** (LAN) conecta una serie de computadoras que se encuentran en una misma habitación, planta o edificio. Normalmente, una **red de área extendida** (WAN) conecta varios edificios, ciudades o países; una multinacional puede disponer de una red WAN para conectar sus oficinas en todo el mundo. Estas redes pueden ejecutar uno o varios protocolos y la continua aparición de nuevas tecnologías está dando lugar a nuevas formas de redes. Por ejemplo, una **red de área metropolitana** (MAN, metropolitan-area network) puede conectar diversos edificios de una ciudad. Los dispositivos BlueTooth y 802.11 utilizan tecnología inalámbrica para comunicarse a distancias de unos pocos metros, creando en esencia lo que se denomina una **red de área pequeña**, como la que puede haber en cualquier hogar.

Los soportes físicos o medios que se utilizan para implementar las redes son igualmente variados. Entre ellos se incluyen el cobre, la fibra óptica y las transmisiones inalámbricas entre satélites, antenas de microondas y aparatos de radio. Cuando se conectan los dispositivos informáticos a teléfonos móviles, se puede crear una red. También se puede emplear incluso comunicación por infrarrojos de corto alcance para establecer una red. A nivel rudimentario, siempre que las com-