Microsoft, aduciendo en esencia que Microsoft incluía demasiada funcionalidad en su sistema operativo, impidiendo a los vendedores de aplicaciones competir. Por ejemplo, un explorador web era una parte esencial del sistema operativo. Como resultado, Microsoft fue declarado culpable de usar su monopolio en los sistemas operativos para limitar la competencia.

1.2 Organización de una computadora

Antes de poder entender cómo funciona una computadora, necesitamos unos conocimientos generales sobre su estructura. En esta sección, veremos varias partes de esa estructura para completar nuestros conocimientos. La sección se ocupa principalmente de la organización de una computadora, así que puede hojearla o saltársela si ya está familiarizado con estos conceptos.

1.2.1 Funcionamiento de una computadora

Una computadora moderna de propósito general consta de una o más CPU y de una serie de controladoras de dispositivo conectadas a través de un bus común que proporciona acceso a la memoria compartida (Figura 1.2). Cada controladora de dispositivo se encarga de un tipo específico de dispositivo, por ejemplo, unidades de disco, dispositivos de audio y pantallas de vídeo. La CPU y las controladoras de dispositivos pueden funcionar de forma concurrente, compitiendo por los ciclos de memoria. Para asegurar el acceso de forma ordenada a la memoria compartida, se proporciona una controladora de memoria cuya función es sincronizar el acceso a la misma.

Para que una computadora comience a funcionar, por ejemplo cuando se enciende o se reinicia, es necesario que tenga un programa de inicio que ejecutar. Este programa de inicio, o **programa de arranque**, suele ser simple. Normalmente, se almacena en la memoria ROM (read only memory, memoria de sólo lectura) o en una memoria EEPROM (electrically erasable programmable read-only memory, memoria de sólo lectura programable y eléctricamente borrable), y se conoce con el término general **firmware**, dentro del hardware de la computadora. Se inicializan todos los aspectos del sistema, desde los registros de la CPU hasta las controladoras de dispositivos y el contenido de la memoria. El programa de arranque debe saber cómo cargar el sistema operativo e iniciar la ejecución de dicho sistema. Para conseguir este objetivo, el programa de arranque debe localizar y cargar en memoria el *kernel* (núcleo) del sistema operativo. Después, el sistema operativo comienza ejecutando el primer proceso, como por ejemplo "init", y espera a que se produzca algún suceso.

La ocurrencia de un suceso normalmente se indica mediante una **interrupción** bien hardware o bien software. El hardware puede activar una interrupción en cualquier instante enviando una señal a la CPU, normalmente a través del bus del sistema. El software puede activar una interrupción ejecutando una operación especial denominada **llamada del sistema** (o también **llamada de monitor**).

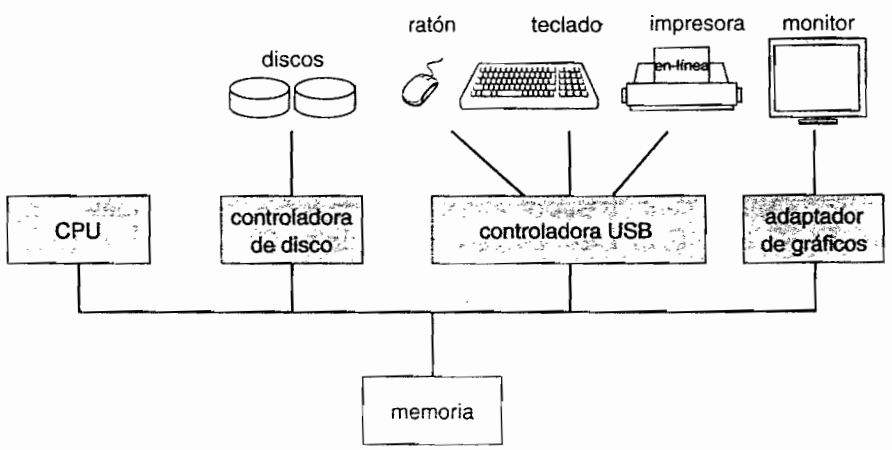

Figura 1.2 Una computadora moderna.

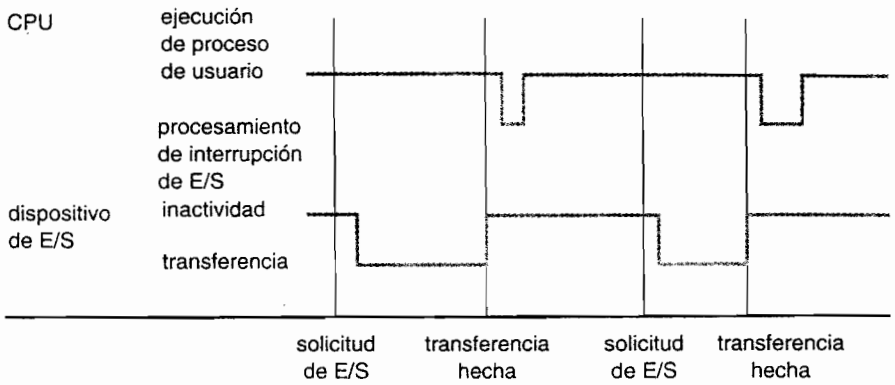

Figura 1.3 Diagrama de tiempos de una interrupción para un proceso de salida.

Cuando se interrumpe a la CPU, deja lo que está haciendo e inmediatamente transfiere la ejecución a una posición fijada (establecida). Normalmente, dicha posición contiene la dirección de inicio de donde se encuentra la rutina de servicio a la interrupción. La rutina de servicio a la interrupción se ejecuta y, cuando ha terminado, la cpu reanuda la operación que estuviera haciendo. En la Figura 1.3 se muestra un diagrama de tiempos de esta operación.

Las interrupciones son una parte importante de la arquitectura de una computadora. Cada diseño de computadora tiene su propio mecanismo de interrupciones, aunque hay algunas funciones comunes. La interrupción debe transferir el control a la rutina de servicio apropiada a la interrupción.

El método más simple para tratar esta transferencia consiste en invocar una rutina genérica para examinar la información de la interrupción; esa rutina genérica, a su vez, debe llamar a la rutina específica de tratamiento de la interrupción. Sin embargo, las interrupciones deben tratarse rápidamente y este método es algo lento. En consecuencia, como sólo es posible un número predefinido de interrupciones, puede utilizarse otro sistema, consistente en disponer una tabla de punteros a las rutinas de interrupción, con el fin de proporcionar la velocidad necesaria. De este modo, se llama a la rutina de interrupción de forma indirecta a través de la tabla, sin necesidad de una rutina intermedia. Generalmente, la tabla de punteros se almacena en la zona inferior de la memoria (las primeras 100 posiciones). Estas posiciones almacenan las direcciones de las rutinas de servicio a la interrupción para los distintos dispositivos. Esta matriz, o **vector de interrupciones**, de direcciones se indexa mediante un número de dispositivo unívoco que se proporciona con la solicitud de interrupción, para obtener la dirección de la rutina de servicio a la interrupción para el dispositivo correspondiente. Sistemas operativos tan diferentes como Windows y UNIX manejan las interrupciones de este modo.

La arquitectura de servicio de las interrupciones también debe almacenar la dirección de la instrucción interrumpida. Muchos diseños antiguos simplemente almacenaban la dirección de interrupción en una posición fija o en una posición indexada mediante el número de dispositivo. Las arquitecturas más recientes almacenan la dirección de retorno en la pila del sistema. Si la rutina de interrupción necesita modificar el estado del procesador, por ejemplo modificando valores del registro, debe guardar explícitamente el estado actual y luego restaurar dicho estado antes de volver. Después de atender a la interrupción, la dirección de retorno guardada se carga en el contador de programa y el cálculo interrumpido se reanuda como si la interrupción no se hubiera producido.

1.2.2 Estructura de almacenamiento

Los programas de la computadora deben hallarse en la memoria principal (también llamada memoria **RAM**, random-access memory, memoria de acceso aleatorio) para ser ejecutados. La memoria principal es el único área de almacenamiento de gran tamaño (millones o miles de millones de bytes) a la que el procesador puede acceder directamente. Habitualmente, se implementa con una tecnología de semiconductores denominada **DRAM (dynamic random-access memory,**

memoria dinámica de acceso aleatorio), que forma una matriz de palabras de memoria. Cada palabra tiene su propia dirección. La interacción se consigue a través de una secuencia de carga (`load`) o almacenamiento (`store`) de instrucciones en direcciones específicas de memoria. La instrucción `load` mueve una palabra desde la memoria principal a un registro interno de la CPU, mientras que la instrucción `store` mueve el contenido de un registro a la memoria principal. Aparte de las cargas y almacenamientos explícitos, la CPU carga automáticamente instrucciones desde la memoria principal para su ejecución.

Un ciclo típico instrucción-ejecución, cuando se ejecuta en un sistema con una arquitectura de **von Neumann**, primero extrae una instrucción de memoria y almacena dicha instrucción en el **registro de instrucciones**. A continuación, la instrucción se decodifica y puede dar lugar a que se extraigan operandos de la memoria y se almacenen en algún registro interno. Después de ejecutar la instrucción con los necesarios operandos, el resultado se almacena de nuevo en memoria. Observe que la unidad de memoria sólo ve un flujo de direcciones de memoria; no sabe cómo se han generado (mediante el contador de instrucciones, indexación, indirección, direcciones literales o algún otro medio) o qué son (instrucciones o datos). De acuerdo con esto, podemos ignorar *cómo* genera un programa una dirección de memoria. Sólo nos interesaremos por la secuencia de direcciones de memoria generada por el programa en ejecución.

Idealmente, es deseable que los programas y los datos residan en la memoria principal de forma permanente. Usualmente, esta situación no es posible por las dos razones siguientes:

1. Normalmente, la memoria principal es demasiado pequeña como para almacenar todos los programas y datos necesarios de forma permanente.
2. La memoria principal es un dispositivo de almacenamiento *volátil* que pierde su contenido cuando se quita la alimentación.

Por tanto, la mayor parte de los sistemas informáticos proporcionan **almacenamiento secundario** como una extensión de la memoria principal. El requerimiento fundamental de este almacenamiento secundario es que se tienen que poder almacenar grandes cantidades de datos de forma permanente.

El dispositivo de almacenamiento secundario más común es el **disco magnético**, que proporciona un sistema de almacenamiento tanto para programas como para datos. La mayoría de los programas (exploradores web, compiladores, procesadores de texto, hojas de cálculo, etc.) se almacenan en un disco hasta que se cargan en memoria. Muchos programas utilizan el disco como origen y destino de la información que están procesando. Por tanto, la apropiada administración del almacenamiento en disco es de importancia crucial en un sistema informático, como veremos en el Capítulo 12.

Sin embargo, en un sentido amplio, la estructura de almacenamiento que hemos descrito, que consta de registros, memoria principal y discos magnéticos, sólo es uno de los muchos posibles sistemas de almacenamiento. Otros sistemas incluyen la memoria caché, los CD-ROM, cintas magnéticas, etc. Cada sistema de almacenamiento proporciona las funciones básicas para guardar datos y mantener dichos datos hasta que sean recuperados en un instante posterior. Las principales diferencias entre los distintos sistemas de almacenamiento están relacionadas con la velocidad, el coste, el tamaño y la volatilidad.

La amplia variedad de sistemas de almacenamiento en un sistema informático puede organizarse en una jerarquía (Figura 1.4) según la velocidad y el coste. Los niveles superiores son caros, pero rápidos. A medida que se desciende por la jerarquía, el coste por bit generalmente disminuye, mientras que el tiempo de acceso habitualmente aumenta. Este compromiso es razonable; si un sistema de almacenamiento determinado fuera a la vez más rápido y barato que otro (siendo el resto de las propiedades las mismas), entonces no habría razón para emplear la memoria más cara y más lenta. De hecho, muchos de los primeros dispositivos de almacenamiento, incluyendo las cintas de papel y las memorias de núcleo, han quedado relegadas a los museos ahora que las cintas magnéticas y las **memorias semiconductoras** son más rápidas y baratas. Los cuatro niveles de memoria superiores de la Figura 1.4 se pueden construir con memorias semiconductoras.

Además de diferenciarse en la velocidad y en el coste, los distintos sistemas de almacenamiento pueden ser volátiles o no volátiles. Como hemos mencionado anteriormente, el **almacena-**

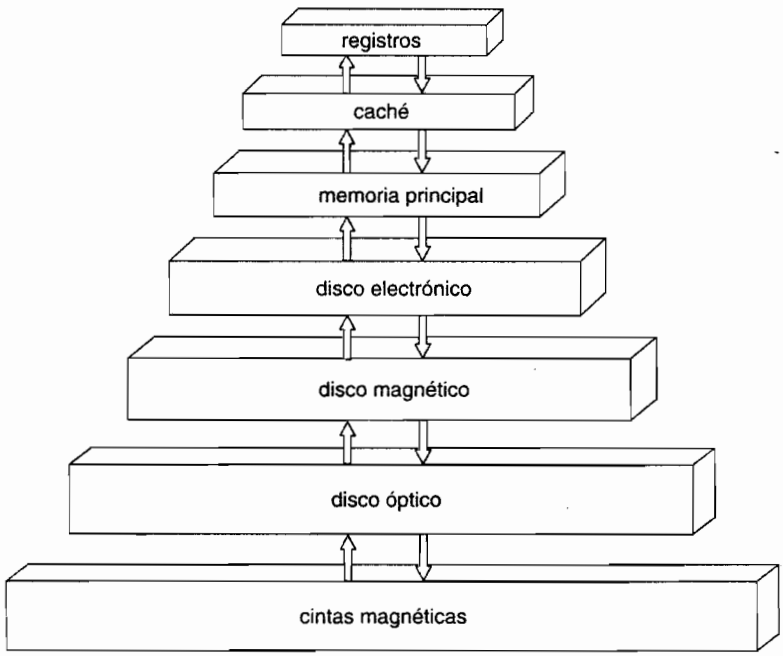

Figura 1.4 Jerarquía de dispositivos de almacenamiento.

miento volátil pierde su contenido cuando se retira la alimentación del dispositivo. En ausencia de baterías caras y sistemas de alimentación de reserva, los datos deben escribirse en **almacenamiento no volátiles** para su salvaguarda. En la jerarquía mostrada en la Figura 1.4, los sistemas de almacenamiento que se encuentran por encima de los discos electrónicos son volátiles y los que se encuentran por debajo son no volátiles. Durante la operación normal, el disco electrónico almacena datos en una matriz DRAM grande, que es volátil. Pero muchos dispositivos de disco electrónico contienen un disco duro magnético oculto y una batería de reserva. Si la alimentación externa se interrumpe, la controladora del disco electrónico copia los datos de la RAM en el disco magnético. Cuando se restaura la alimentación, los datos se cargan de nuevo en la RAM. Otra forma de disco electrónico es la memoria flash, la cual es muy popular en las cámaras, los PDA **(personal digital assistant)** y en los robots; asimismo, está aumentando su uso como dispositivo de almacenamiento extraíble en computadoras de propósito general. La memoria flash es más lenta que la DRAM, pero no necesita estar alimentada para mantener su contenido. Otra forma de almacenamiento no volátil es la **NVRAM**, que es una DRAM con batería de reserva. Esta memoria puede ser tan rápida como una DRAM, aunque sólo mantiene su carácter no volátil durante un tiempo limitado.

El diseño de un sistema de memoria completo debe equilibrar todos los factores mencionados hasta aquí: sólo debe utilizarse una memoria cara cuando sea necesario y emplear memorias más baratas y no volátiles cuando sea posible. Se pueden instalar memorias caché para mejorar el rendimiento cuando entre dos componentes existe un tiempo de acceso largo o disparidad en la velocidad de transferencia.

1.2.3 Estructura de E/S

Los de almacenamiento son sólo uno de los muchos tipos de dispositivos de E/S que hay en un sistema informático. Gran parte del código del sistema operativo se dedica a gestionar la entrada y la salida, debido a su importancia para la fiabilidad y rendimiento del sistema y debido también a la variada naturaleza de los dispositivos. Vamos a realizar, por tanto, un repaso introductorio del tema de la E/S.

Una computadora de propósito general consta de una o más CPU y de múltiples controladoras de dispositivo que se conectan a través de un bus común. Cada controladora de dispositivo se encarga de un tipo específico de dispositivo. Dependiendo de la controladora, puede haber más

de un dispositivo conectado. Por ejemplo, siete o más dispositivos pueden estar conectados a la controladora **SCSI** (**small computer-system interface**, interfaz para sistemas informáticos de pequeño tamaño). Una controladora de dispositivo mantiene algunos búferes locales y un conjunto de registros de propósito especial. La controladora del dispositivo es responsable de transferir los datos entre los dispositivos periféricos que controla y su búfer local. Normalmente, los sistemas operativos tienen un **controlador** (*driver*) **de dispositivo** para cada controladora (*controller*) de dispositivo. Este software controlador del dispositivo es capaz de entenderse con la controladora hardware y presenta al resto del sistema operativo una interfaz uniforme mediante la cual comunicarse con el dispositivo.

Al iniciar una operación de E/S, el controlador del dispositivo carga los registros apropiados de la controladora hardware. Ésta, a su vez, examina el contenido de estos registros para determinar qué acción realizar (como, por ejemplo, "leer un carácter del teclado"). La controladora inicia entonces la transferencia de datos desde el dispositivo a su búfer local. Una vez completada la transferencia de datos, la controladora hardware informa al controlador de dispositivo, a través de una interrupción, de que ha terminado la operación. El controlador devuelve entonces el control al sistema operativo, devolviendo posiblemente los datos, o un puntero a los datos, si la operación ha sido una lectura. Para otras operaciones, el controlador del dispositivo devuelve información de estado.

Esta forma de E/S controlada por interrupción resulta adecuada para transferir cantidades pequeñas de datos, pero representa un desperdicio de capacidad de proceso cuando se usa para movimientos masivos de datos, como en la E/S de disco. Para resolver este problema, se usa el acceso directo a memoria (**DMA, direct memory access**). Después de configurar búferes, punteros y contadores para el dispositivo de E/S, la controladora hardware transfiere un bloque entero de datos entre su propio búfer y la memoria, sin que intervenga la CPU. Sólo se genera una interrupción por cada bloque, para decir al controlador software del dispositivo que la operación se ha completado, en lugar de la interrupción por byte generada en los dispositivos de baja velocidad. Mientras la controladora hardware realiza estas operaciones, la CPU está disponible para llevar a cabo otros trabajos.

Algunos sistemas de gama alta emplean una arquitectura basada en conmutador, en lugar de en bus. En estos sistemas, los diversos componentes pueden comunicarse con otros componentes de forma concurrente, en lugar de competir por los ciclos de un bus compartido. En este caso,

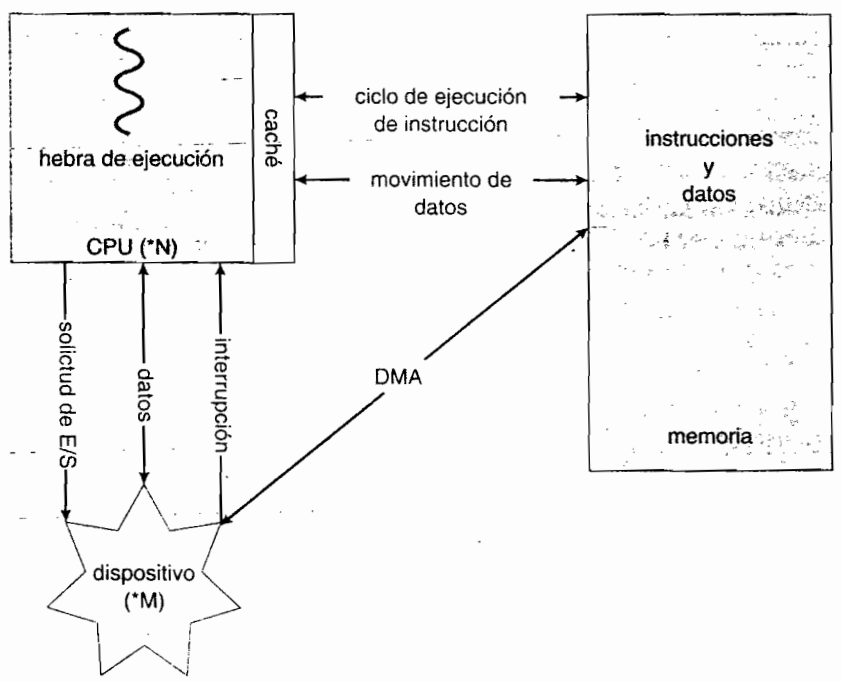

Figura 1.5 Funcionamiento de un sistema informático moderno.

el acceso directo a memoria es incluso más eficaz. La Figura 1.5 muestra la interacción de los distintos componentes de un sistema informático.

1.3 Arquitectura de un sistema informático

En la Sección 1.2 hemos presentado la estructura general de un sistema informático típico. Un sistema informático se puede organizar de varias maneras diferentes, las cuales podemos clasificar de acuerdo con el número de procesadores de propósito general utilizados.

1.3.1 Sistemas de un solo procesador

La mayor parte de los sistemas sólo usan un procesador. No obstante, la variedad de sistemas de un único procesador puede ser realmente sorprendente, dado que van desde los PDA hasta los sistemas *mainframe*. En un sistema de un único procesador, hay una CPU principal capaz de ejecutar un conjunto de instrucciones de propósito general, incluyendo instrucciones de los procesos de usuario. Casi todos los sistemas disponen también de otros procesadores de propósito especial. Pueden venir en forma de procesadores específicos de un dispositivo, como por ejemplo un disco, un teclado o una controladora gráfica; o, en *mainframes*, pueden tener la forma de procesadores de propósito general, como procesadores de E/S que transfieren rápidamente datos entre los componentes del sistema.

Todos estos procesadores de propósito especial ejecutan un conjunto limitado de instrucciones y no ejecutan procesos de usuario. En ocasiones, el sistema operativo los gestiona, en el sentido de que les envía información sobre su siguiente tarea y monitoriza su estado. Por ejemplo, un microprocesador incluido en una controladora de disco recibe una secuencia de solicitudes procedentes de la CPU principal e implementa su propia cola de disco y su algoritmo de programación de tareas. Este método libera a la CPU principal del trabajo adicional de planificar las tareas de disco. Los PC contienen un microprocesador en el teclado para convertir las pulsaciones de tecla en códigos que se envían a la CPU. En otros sistemas o circunstancias, los procesadores de propósito especial son componentes de bajo nivel integrados en el hardware. El sistema operativo no puede comunicarse con estos procesadores, sino que éstos hacen su trabajo de forma autónoma. La presencia de microprocesadores de propósito especial resulta bastante común y no convierte a un sistema de un solo procesador en un sistema multiprocesador. Si sólo hay una CPU de propósito general, entonces el sistema es de un solo procesador.

1.3.2 Sistemas multiprocesador

Aunque los sistemas de un solo procesador son los más comunes, la importancia de los **sistemas multiprocesador** (también conocidos como **sistemas paralelos** o **sistemas fuertemente acoplados**) está siendo cada vez mayor. Tales sistemas disponen de dos o más procesadores que se comunican entre sí, compartiendo el bus de la computadora y, en ocasiones, el reloj, la memoria y los dispositivos periféricos.

Los sistemas multiprocesador presentan tres ventajas fundamentales:

1. **Mayor rendimiento**. Al aumentar el número de procesadores, es de esperar que se realice más trabajo en menos tiempo. Sin embargo, la mejora en velocidad con N procesadores no es N, sino que es menor que N. Cuando múltiples procesadores cooperan en una tarea, cierta carga de trabajo se emplea en conseguir que todas las partes funcionen correctamente. Esta carga de trabajo, más la contienda por los recursos compartidos, reducen la ganancia esperada por añadir procesadores adicionales. De forma similar, N programadores trabajando simultáneamente no producen N veces la cantidad de trabajo que produciría un solo programador .
2. **Economía de escala**. Los sistemas multiprocesador pueden resultar más baratos que su equivalente con múltiples sistemas de un solo procesador, ya que pueden compartir perifé-

ricos, almacenamiento masivo y fuentes de alimentación. Si varios programas operan sobre el mismo conjunto de datos, es más barato almacenar dichos datos en un disco y que todos los procesadores los compartan, que tener muchas computadoras con discos locales y muchas copias de los datos.

3. **Mayor fiabilidad.** Si las funciones se pueden distribuir de forma apropiada entre varios procesadores, entonces el fallo de un procesador no hará que el sistema deje de funcionar, sino que sólo se ralentizará. Si tenemos diez procesadores y uno falla, entonces cada uno de los nueve restantes procesadores puede asumir una parte del trabajo del procesador que ha fallado. Por tanto, el sistema completo trabajará un 10% más despacio, en lugar de dejar de funcionar.

En muchas aplicaciones, resulta crucial conseguir la máxima fiabilidad del sistema informático. La capacidad de continuar proporcionando servicio proporcionalmente al nivel de hardware superviviente se denomina **degradación suave**. Algunos sistemas van más allá de la degradación suave y se denominan sistemas **tolerantes a fallos**, dado que pueden sufrir un fallo en cualquier componente y continuar operando. Observe que la tolerancia a fallos requiere un mecanismo que permita detectar, diagnosticar y, posiblemente, corregir el fallo. El sistema HP NonStop (antes sistema Tandem) duplica el hardware y el software para asegurar un funcionamiento continuado a pesar de los fallos. El sistema consta de múltiples parejas de CPU que trabajan sincronizadamente. Ambos procesadores de la pareja ejecutan cada instrucción y comparan los resultados. Si los resultados son diferentes, quiere decir que una de las CPU de la pareja falla y ambas dejan de funcionar. El proceso que estaba en ejecución se transfiere a otra pareja de CPU y la instrucción fallida se reinicia. Esta solución es cara, dado que implica hardware especial y una considerable duplicación del hardware.

Los sistemas multiprocesador actualmente utilizados son de dos tipos. Algunos sistemas usan el **multiprocesamiento asimétrico**, en el que cada procesador se asigna a una tarea específica. Un procesador maestro controla el sistema y el resto de los procesadores esperan que el maestro les dé instrucciones o tienen asignadas tareas predefinidas. Este esquema define una relación maestro-esclavo. El procesador maestro planifica el trabajo de los procesadores esclavos y se lo asigna.

Los sistemas más comunes utilizan el **multiprocesamiento simétrico (SMP)**, en el que cada procesador realiza todas las tareas correspondientes al sistema operativo. En un sistema SMP, todos los procesadores son iguales; no existe una relación maestro-esclavo entre los procesadores. La Figura 1.6 ilustra una arquitectura SMP típica. Un ejemplo de sistema SMP es Solaris, una versión comercial de UNIX diseñada por Sun Microsystems. Un sistema Sun se puede configurar empleando docenas de procesadores que ejecuten Solaris. La ventaja de estos modelos es que se pueden ejecutar simultáneamente muchos procesos (se pueden ejecutar N procesos si se tienen N CPU) sin que se produzca un deterioro significativo del rendimiento. Sin embargo, hay que controlar cuidadosamente la E/S para asegurar que los datos lleguen al procesador adecuado. También, dado que las CPU están separadas, una puede estar en un período de inactividad y otra puede estar sobrecargada, dando lugar a ineficiencias. Estas situaciones se pueden evitar si los procesadores comparten ciertas estructuras de datos. Un sistema multiprocesador de este tipo permitirá que los procesos y los recursos (como la memoria) sean compartidos dinámicamente entre los distintos procesadores, lo que permite disminuir la varianza entre la carga de trabajo de los procesadores. Un sistema así se debe diseñar con sumo cuidado, como veremos en el Capítulo 6. Prácticamente todos los sistemas operativos modernos, incluyendo Windows, Windows XP, Mac OS X y Linux, proporcionan soporte para SMP.

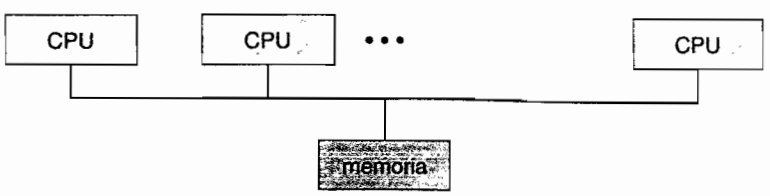

Figura 1.6 Arquitectura de multiprocesamiento simétrico.

La diferencia entre multiprocesamiento simétrico y asimétrico puede deberse tanto al hardware como al software. Puede que haya un hardware especial que diferencie los múltiples procesadores o se puede escribir el software para que haya sólo un maestro y múltiples esclavos. Por ejemplo, el sistema operativo SunOS Versión 4 de Sun proporciona multiprocesamiento asimétrico, mientras que Solaris (Versión 5) es simétrico utilizando el mismo hardware.

Una tendencia actual en el diseño de las CPU es incluir múltiples **núcleos** de cálculo en un mismo chip. En esencia, se trata de chips multiprocesador. Los chips de dos vías se están convirtiendo en la corriente dominante, mientras que los chips de *N* vías están empezando a ser habituales en los sistemas de gama alta. Dejando aparte las consideraciones sobre la arquitectura, como la caché, la memoria y la contienda de bus, estas CPU con múltiples núcleos son vistas por el sistema operativo simplemente como *N* procesadores estándar.

Por último, los **servidores blade** son un reciente desarrollo, en el que se colocan múltiples procesadores, tarjetas de E/S y tarjetas de red en un mismo chasis. La diferencia entre estos sistemas y los sistemas multiprocesador tradicionales es que cada tarjeta de procesador blade arranca independientemente y ejecuta su propio sistema operativo. Algunas tarjetas de servidor blade también son multiprocesador, lo que difumina la línea divisoria entre los distintos tipos de computadoras. En esencia, dichos servidores constan de múltiples sistemas multiprocesador independientes.

1.3.3 Sistemas en cluster

Otro tipo de sistema con múltiples CPU es el **sistema en *cluster***. Como los sistemas multiprocesador, los sistemas en cluster utilizan múltiples CPU para llevar a cabo el trabajo. Los sistemas en cluster se diferencian de los sistemas de multiprocesamiento en que están formados por dos o más sistemas individuales acoplados. La definición del término *en cluster* no es concreta; muchos paquetes comerciales argumentan acerca de qué es un sistema en *cluster* y por qué una forma es mejor que otra. La definición generalmente aceptada es que las computadoras en cluster comparten el almacenamiento y se conectan entre sí a través de una red de área local (LAN, **local area network**), como se describe en la Sección 1.10, o mediante una conexión más rápida como InfiniBand.

Normalmente, la conexión en cluster se usa para proporcionar un servicio con **alta disponibilidad**; es decir, un servicio que funcionará incluso si uno o más sistemas del cluster fallan. Generalmente, la alta disponibilidad se obtiene añadiendo un nivel de redundancia al sistema. Sobre los nodos del *cluster* se ejecuta una capa de software de gestión del *cluster*. Cada nodo puede monitorizar a uno o más de los restantes (en una LAN). Si la máquina monitorizada falla, la máquina que la estaba monitorizando puede tomar la propiedad de su almacenamiento y reiniciar las aplicaciones que se estuvieran ejecutando en la máquina que ha fallado. Los usuarios y clientes de las aplicaciones sólo ven una breve interrupción del servicio.

El cluster se puede estructurar simétrica o asimétricamente. En un **cluster asimétrico**, una máquina está en **modo de espera en caliente**, mientras que la otra está ejecutando las aplicaciones. La máquina *host* en modo de espera en caliente no hace nada más que monitorizar al servidor activo. Si dicho servidor falla, el *host* que está en espera pasa a ser el servidor activo. En el **modo simétrico**, dos o más *hosts* ejecutan aplicaciones y se monitorizan entre sí. Este modo es obviamente más eficiente, ya que usa todo el hardware disponible. Aunque, desde luego, requiere que haya disponible más de una aplicación para ejecutar.

Otras formas de cluster incluyen el *cluster* en paralelo y los *clusters* conectados a una red de área extensa (WAN, **wide area network**), como se describe en la Sección 1.10. Los *clusters* en paralelo permiten que múltiples *hosts* accedan a unos mismos datos, disponibles en el almacenamiento compartido. Dado que la mayoría de los sistemas operativos no soportan el acceso simultáneo a datos por parte de múltiples *hosts*, normalmente los clusters en paralelo requieren emplear versiones especiales de software y versiones especiales de las aplicaciones. Por ejemplo, Oracle Parallel Server es una versión de la base de datos de Oracle que se ha diseñado para ejecutarse en un cluster paralelo. Cada una de las máquinas ejecuta Oracle y hay una capa de software que controla el acceso al disco compartido. Cada máquina tiene acceso total a todos los datos de la base

de datos. Para proporcionar este acceso compartido a los datos, el sistema tiene que proporcionar mecanismos de control de acceso y de bloqueo, para asegurar que no se produzcan conflictos entres las operaciones. Esta función, conocida habitualmente como **DLM (distributed lock manager**, gestor de bloqueos distribuido) está incluida en algunas tecnologías de cluster.

La tecnología de implementación de clusters está cambiando rápidamente. Algunos productos de cluster permiten interconectar docenas de sistemas en un mismo cluster, así como conectar en cluster una serie de nodos que estén separados por muchos kilómetros. Muchas de estas mejoras han sido posibles gracias a las redes de área de almacenamiento o redes **SAN (storage-area network)**, que se describen en la Sección 12.3.3 y que permiten conectar muchos sistemas a una misma batería de almacenamiento. Si las aplicaciones y sus datos se almacenan en la red SAN, entonces el software de gestión del cluster puede asignar la aplicación para que se ejecute en cualquier *host* que esté conectado a la SAN. Si el *host* falla, entonces cualquier otro *host* puede sustituirle. En un cluster de base de datos, docenas de *hosts* pueden compartir la misma base de datos, incrementando enormemente el rendimiento y la fiabilidad.

1.4 Estructura de un sistema operativo

Ahora que hemos visto la información básica sobre la arquitectura y organización de un sistema informático, estamos preparados para hablar sobre los sistemas operativos. Un sistema operativo proporciona el entorno en el cual se ejecutan los programas. Internamente, los sistemas operativos varían enormemente en su composición, ya que se organizan a lo largo de muchas líneas diferentes. Sin embargo, tienen muchas características comunes, que consideraremos en esta sección.

Uno de los aspectos más importantes de los sistemas operativos es la capacidad para multiprogramar. En general, un solo usuario no puede mantener la CPU o los dispositivos de E/S ocupados continuamente. La **multiprogramación** incrementa el uso de la CPU organizando los trabajos (código y datos) de modo que la CPU siempre tenga uno que ejecutar.

La idea es la siguiente: el sistema operativo mantiene en memoria simultáneamente varios trabajos (Figura 1.7). Este conjunto de trabajos puede ser un subconjunto de los trabajos guardados en la cola de trabajos, la cual contiene todos los trabajos que entran en el sistema, dado que el número de trabajos que puede mantenerse simultáneamente en memoria es normalmente menor que el [illegible] que puede mantener la cola de trabajos. El sistema operativo toma y comienza