dos lecturas de acceso directo. La Figura 10.5 muestra una situación similar, implementada mediante los archivos de índice y archivos relativos de VMS.

## 10.3 Estructura de directorios

Hasta este punto, hemos estado analizando "un sistema de archivos". En realidad, los sistemas pueden tener cero o más sistemas de archivos y los sistemas de archivos pueden ser de tipos variados. Por ejemplo, un sistema Solaris típico puede tener unos cuantos sistemas de archivos UFS, un sistema de archivos VFS y algunos sistemas de archivos NFS. En el Capítulo 11 se analizan los detalles de la implementación de los sistemas de archivos.

Los sistemas de archivos de las computadoras pueden, por tanto, tener una gran complejidad. Algunos sistemas almacenan millones de archivos en terabytes de disco. Para gestionar todos estos datos, necesitamos organizarlos de alguna manera y esta organización implica el uso de directorios. En esta sección, vamos a analizar el tema de la estructura de directorios. Pero primero, es necesario explicar algunas características básicas de la estructura de almacenamiento.

### 10.3.1 Estructura de almacenamiento

Un disco (o cualquier dispositivo de almacenamiento que sea lo suficientemente grande) puede utilizarse completamente para un sistema de archivos. Sin embargo, en ciertas ocasiones es deseable colocar múltiples sistemas de archivos en un mismo disco o utilizar partes de un disco para un sistema de archivos y otras partes para otras cosas, como por ejemplo para espacio de intercambio o como espacio de disco sin formato (**raw**). Estas partes se conocen con diversos nombres, como **particiones**, **franjas** o (en el mundo IBM) **minidiscos**. Podemos crear un sistema de archivos en cada una de estas partes del disco. Como veremos en el siguiente capítulo, las partes también pueden combinarse para formar estructuras de mayor tamaño, conocidas con el nombre de **volúmenes**, y también pueden crearse sistemas de archivos en dichos volúmenes. Pero por el momento, en aras de la claridad, utilizaremos el término volumen simplemente para referirnos a un espacio de almacenamiento que alberga un sistema de archivos. Cada volumen puede considerarse como si fuera un disco virtual. Los volúmenes pueden también almacenar múltiples sistemas operativos, permitiendo que un sistema se inicie en cualquiera de ellos y ejecute dicho sistema operativo.

Cada volumen que contenga un sistema de archivos debe también contener información acerca de los archivos almacenados en el sistema. Esta información se almacena como entrada en un **directorio de dispositivo** o **tabla de contenidos del volumen**. El directorio del dispositivo (más comúnmente conocido simplemente como **directorio**) almacena información de todos los archivos de dicho volumen, como por ejemplo el nombre, la ubicación, el tamaño y el tipo. La Figura 10.6 muestra una organización típica de un sistema de archivo.

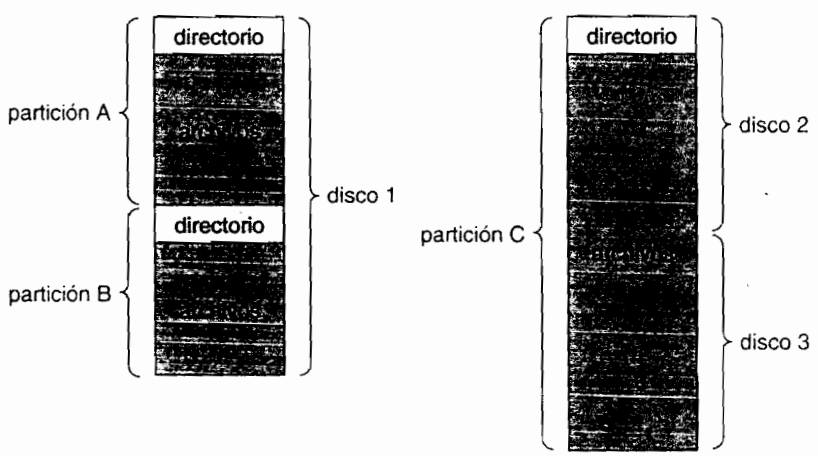


Figura 10.6 Una organización típica de un sistema de archivos.

### 10.3.2 Introducción a los directorios

El directorio puede considerarse como una tabla de símbolos que traduce los nombres de archivo a sus correspondientes entradas de directorio. Si adoptamos esta visión, es fácil comprender que el propio directorio puede organizarse de muchas formas. Queremos poder insertar entradas, borrar entradas, buscar una entrada conociendo su nombre y enumerar todas las entradas del directorio. En esta sección, vamos a examinar diversos esquemas para definir la estructura lógica del sistema de directorios.

Cuando consideremos una estructura de directorio concreta, tendremos que tener presentes las operaciones que habrá que realizar con el directorio:

- **Búsqueda de un archivo**. Tenemos que poder explorar la estructura de directorio para encontrar la entrada correspondiente a un archivo concreto. Puesto que los archivos tienen nombres simbólicos y los nombres similares pueden indicar que existe una relación entre los archivos, también queremos poder encontrar todos los archivos cuyos nombres se correspondan con un patrón concreto.
- **Crear un archivo**. Es necesario poder crear nuevos archivos y añadirlos al directorio.
- **Borrar un archivo**. Cuando un archivo ya no es necesario, queremos poder eliminarlo del directorio.
- **Listar un directorio**. Tenemos que poder enumerar los archivos contenidos en un directorio y el contenido de la entrada de directorio correspondiente a cada uno de los archivos de la lista.
- **Renombrar un archivo**. Puesto que el nombre de un archivo representa su contenido para los usuarios, debemos poder cambiar el nombre cuando el contenido o el uso del archivo varíen. Renombrar un archivo puede también significar que se modifique su posición dentro de la estructura de directorio.
- **Recorrer el sistema de archivos**. Puede que queramos acceder a todos los directorios y a todos los archivos contenidos dentro de una estructura de directorios. Para conseguir una mayor fiabilidad, resulta conveniente guardar el contenido y la estructura de todo el sistema de archivos a intervalos regulares. A menudo, esto se suele hacer copiando todos los archivos en una cinta magnética. Esta técnica proporciona una copia de seguridad para el caso de que se produzca un fallo del sistema. Además, si un archivo ya ha dejado de utilizarse, el archivo puede copiarse en cinta y puede liberarse el espacio de disco ocupado por ese archivo, con el fin de que lo reutilicen otros archivos.

En las siguientes secciones, vamos a describir los esquemas más comunes que se usan para definir la estructura lógica de un directorio.

### 10.3.3 Directorio de un único nivel

La estructura de directorio más simple es el directorio de un único nivel. Todos los archivos están contenidos en el mismo directorio, que resulta fácil de mantener y de comprender (Figura 10.7).

Sin embargo, un directorio de un único nivel tiene limitaciones significativas cuando el número de archivos se incrementa o cuando el sistema tiene más de un usuario. Puesto que todos los archivos se encuentran en el mismo directorio, deberán tener nombres distintivos. Si dos usuarios llaman a su archivo de datos *test*, se violará la regla de unicidad de los nombres. Por ejemplo, en una clase de programación, 23 estudiantes denominaron al programa correspondiente a la segunda de las prácticas realizadas *prog2*, mientras que otros 11 lo denominaron *assign2*. Aunque los nombres de archivo se seleccionan, por regla general, de modo que reflejen el contenido del archivo, a menudo están limitados en longitud, lo que complica la tarea de hacer que esos nombres de archivos sean unívocos. El sistema operativo MS-DOS sólo permite utilizar nombres de archivo de once caracteres; UNIX, por el contrario, permite hasta 255 caracteres.

Incluso aunque sólo haya un usuario en un directorio de un único nivel, puede que ese usuario tenga dificultades para recordar los nombres de todos los archivos a medida que se incremen-

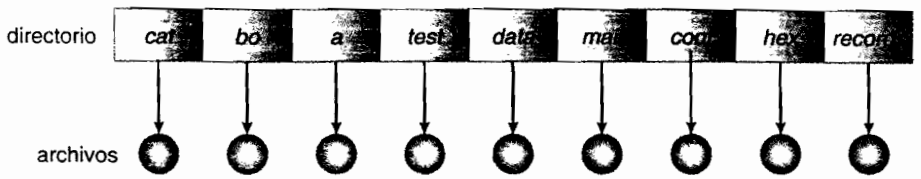


**Figura 10.7** Directorio de un único nivel.

ta el número de archivos. No resulta extraño que un usuario tenga cientos de archivos en su sistema informático y un número igual de archivos adicionales en otros sistemas. Controlar todos esos archivos es una tarea extremadamente compleja.

### 10.3.4 Directorio en dos niveles

Como hemos visto, los directorios de un único nivel conducen a menudo a que exista confusión entre los nombres de archivo de los distintos usuarios. La solución estándar consiste en crear un directorio *separado* para cada usuario.

En la estructura de directorio de dos niveles, cada usuario tiene su propio **directorio de archivos de usuario** (UFD, user file directory). Cada uno de los UFD tiene una estructura similar, pero sólo incluye los archivos de un único usuario. Cuando un trabajo de un usuario se inicia o cuando un usuario se conecta al sistema, se explora el **directorio maestro de archivos** (MFD, master file directory) del sistema. El MFD está indexado por el nombre de usuario o por el número de cuenta y cada una de sus entradas apunta al UFD de dicho usuario (Figura 10.8).

Cuando un usuario hace referencia a un archivo concreto, sólo se explora su propio UFD. Por tanto, cada uno de los usuarios puede tener archivos con el mismo nombre, siempre y cuando los nombres de archivo dentro de cada UFD sean unívocos. Para crear un archivo para un usuario, el sistema operativo sólo explora el UFD de ese usuario para comprobar si ya existe otro archivo con el mismo nombre. Para borrar un archivo, el sistema operativo confina su búsqueda al UFD local y no puede, por tanto, borrar accidentalmente un archivo de otro usuario que tenga el mismo nombre.

Los propios directorios de usuario deben poder crearse y borrarse según sea necesario. Para ello se ejecuta un programa especial del sistema, con el nombre de usuario apropiado y la correspondiente información de cuenta. El programa crea un nuevo UFD y añade una entrada para él en el MFD. La ejecución de este programa puede estar restringida a los administradores del sistema. La asignación de espacio de disco para las direcciones de usuario puede gestionarse mediante las técnicas explicadas en el Capítulo 11 para los propios archivos.

Aunque la estructura de directorio en dos niveles resuelve el problema de la colisión de nombres, sigue teniendo ciertas desventajas. Esta estructura aísla efectivamente a un usuario y otro y ese aislamiento es una ventaja cuando los usuarios son completamente independientes, pero puede llegar a ser una desventaja cuando los usuarios *quieren* cooperar en una cierta tarea y poder acceder a los archivos del otro. Algunos sistemas simplemente no permiten que un usuario acceda a los archivos locales de otro usuario.

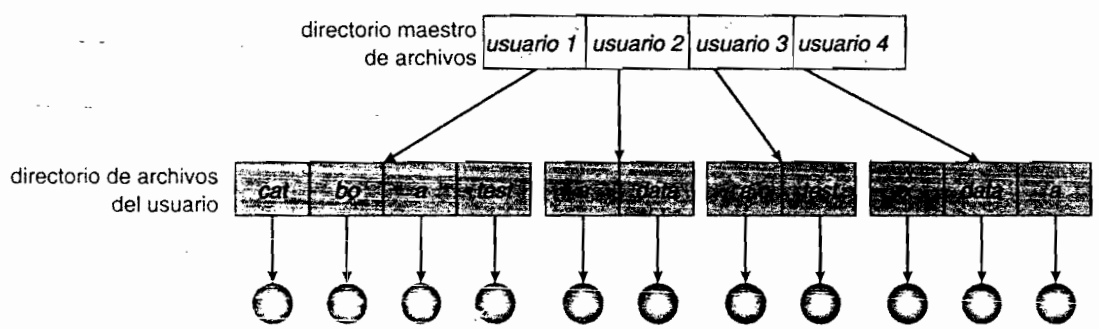


**Figura 10.8** Estructura de directorios en dos niveles.

Si queremos permitir ese tipo de acceso, cada usuario debe tener la posibilidad de especificar un archivo que pertenezca al directorio de otro usuario. Para denominar a los archivos de manera unívoca en una estructura de directorio de dos niveles, debemos proporcionar tanto el nombre de usuario como el nombre del archivo. Un directorio en dos niveles puede verse como una estructura de árbol, o de árbol invertido, de altura 2. La raíz del árbol es el MFD y sus descendientes directos son los UFD. Los descendientes de los UFD son los propios archivos, que actúan como hojas del árbol. Al especificar un nombre de usuario y un nombre de archivo, estamos definiendo una ruta dentro del árbol que va desde la raíz (el MFD) hasta una hoja (el archivo especificado). Por tanto, un nombre de usuario y un nombre de archivo definen un *nombre de ruta*. Cada archivo del sistema tiene un nombre de ruta y para designar a un archivo de manera unívoca, el usuario debe conocer el nombre de ruta del archivo deseado.

Por ejemplo, si el usuario A quiere acceder a su propio archivo denominado *test*, simplemente puede referirse a él como *test*. Sin embargo, para acceder al archivo denominado *test* del usuario B (cuyo nombre de entrada de directorio es *userb*), puede que tenga que utilizar la notación */userb/test*. Cada sistema tiene su propia sintaxis para nombrar los archivos contenidos en los directorios que no pertenecen al propio usuario.

Se necesita una sintaxis adicional para especificar el volumen de un archivo. Por ejemplo, en MS-DOS un volumen se especifica mediante una letra seguida de un carácter de dos puntos. Por tanto, una especificación de archivo podría tener el siguiente aspecto *C:/userb/test*. Algunos sistemas van todavía más allá y separan las partes de la especificación correspondientes al volumen, al nombre del directorio y al nombre del archivo. Por ejemplo, en VMS, el archivo *login.com* puede especificarse como *u:[sst.jdeck]login.com;1*, donde *u* es el nombre del volumen, *sst* es el nombre del directorio, *jdeck* es el nombre del subdirectorio y 1 es el número de versión. Otros sistemas simplemente tratan el nombre del volumen como parte del nombre de directorio. El primer nombre que se proporcione será el del volumen y el resto se referirá al directorio y al archivo. Por ejemplo, */u/pbg/test* podría especificar el volumen *u*, el directorio *pbg* y el archivo *test*.

Un caso especial de esta situación es el que se refiere a los archivos del sistema. Los programas proporcionados como parte del sistema (cargadores, ensambladores, compiladores, rutinas de utilidad, bibliotecas, etc.) están generalmente definidos como archivos. Cuando se proporcionan los comandos apropiados al sistema operativo, el cargador carga estos archivos y los ejecuta. Muchos intérpretes de comandos simplemente tratan dicho comando como el nombre de un archivo que hay que cargar y ejecutar. Tal como hemos definido el sistema de directorios, este nombre de archivos se buscaría en el UFD actual. Una solución consiste en copiar los archivos del sistema dentro de cada UFD, pero copiar todos los archivos del sistema implicaría desperdiciar una cantidad enorme de espacio (si los archivos del sistema requieren 5 MB, soportar a 12 usuarios requeriría $5 \times 12 = 60$ MB simplemente para las copias de los archivos del sistema).

La solución estándar consiste en complicar el procedimiento de búsqueda ligeramente. Se define un directorio de usuario especial para contener los archivos del sistema (por ejemplo, el usuario 0). Cada vez que se proporciona un nombre de archivo para cargarlo, el sistema operativo analiza primero el UFD local; si encuentra allí el archivo, lo utiliza, pero si no lo encuentra, el sistema explora automáticamente el directorio de usuario especial que contiene los archivos del sistema. La secuencia de directorios que se exploran cada vez que se proporciona un nombre de archivos se denomina **ruta de búsqueda**. La ruta de búsqueda puede emplearse para contener un número ilimitado de directorios que haya que explorar cada vez que se proporcione un nombre de comando. Este método es el más utilizado en UNIX y MS-DOS. También pueden diseñarse los sistemas de modo que cada usuario tenga su propia ruta de búsqueda.

### 10.3.5 Directorios con estructura de árbol

Una vez que hemos visto cómo puede contemplarse un directorio en dos niveles como un árbol de dos niveles, la generalización natural consiste en ampliar la estructura de directorios para que sea un árbol de altura arbitraria (Figura 10.9). Esta generalización permite a los usuarios crear sus propios subdirectorios y organizar sus archivos correspondientemente. Los árboles son la estructura de directorio más común. Cada árbol tiene un directorio raíz y todos los archivos del sistema tienen un nombre de ruta distintivo.

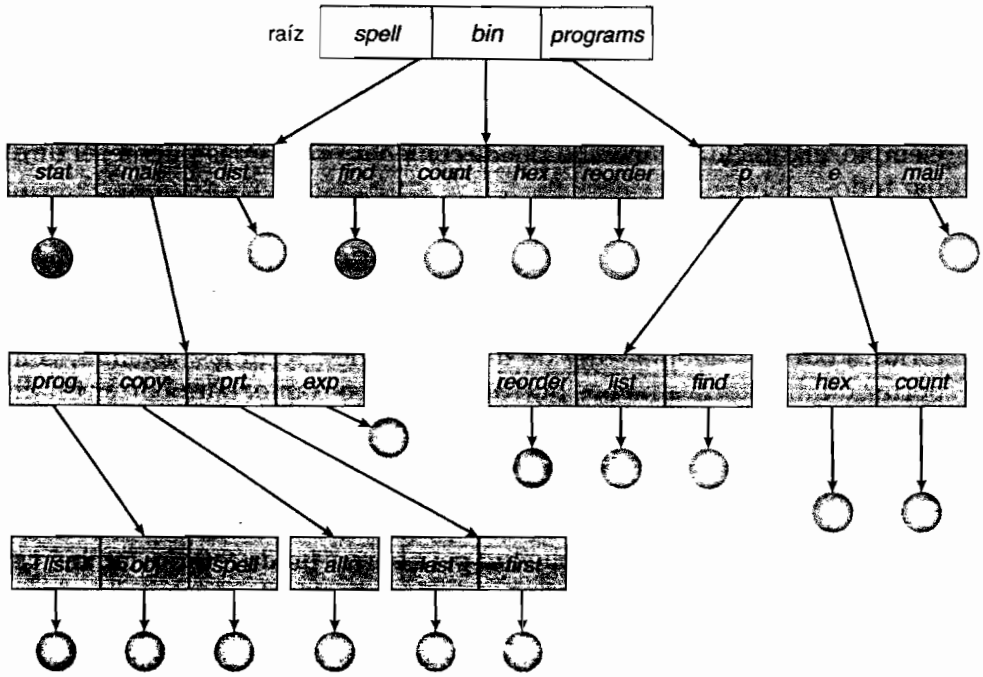


**Figura 10.9** Estructura de directorio en forma de árbol.

Cada directorio (o subdirectorio) contiene un conjunto de archivos o subdirectorios. Un directorio es simplemente otro archivo, pero que se trata de forma especial. Todos los directorios tienen el mismo formato interno. Un bit de cada entrada de directorio define si esa entrada es un archivo (0) o un subdirectorio (1). Se utilizan llamadas al sistema especiales para crear y borrar los directorios.

En el uso normal, cada proceso tiene un directorio actual. Ese **directorio actual** debe contener la mayor parte de los archivos que actualmente interesen al proceso. Cuando se hace referencia a un archivo, se explora el directorio actual. Si se necesita un archivo que no se encuentre en el directorio actual, entonces el usuario deberá normalmente especificar un nombre de ruta o cambiar el directorio actual, para situarse en el directorio donde fue almacenado ese archivo. Para cambiar de directorio, se proporciona una llamada al sistema que toma como parámetro un nombre de directorio y lo utiliza para redefinir el directorio actual. Así, el usuario puede cambiar su directorio actual cada vez que lo desee. Entre una llamada al sistema `change directory` (cambiar directorio) y la siguiente, todas las llamadas al sistema `open` analizarán el directorio actual en busca del archivo especificado. Observe que la ruta de búsqueda puede o no contener una entrada especial que signifique "el directorio actual".

El directorio actual inicial de la shell de inicio de sesión de un usuario se designa en el momento de comenzar el trabajo del usuario o en el momento en que el usuario inicia la sesión. El sistema operativo analiza el archivo de cuentas (o alguna otra ubicación predefinida) con el fin de localizar la entrada correspondiente a este usuario (para propósitos de contabilización). En el archivo de cuentas habrá almacenado un puntero (o el nombre) al directorio inicial del usuario. Este puntero se copia en una variable local de ese usuario que especifica el directorio actual inicial del mismo. A partir de esa shell, pueden arrancarse otros procesos. El directorio actual de cada subproceso será, usualmente, el directorio actual que su proceso padre tenía en el momento de crear el subproceso.

Los nombres de ruta pueden ser de dos tipos: *absolutos* y *relativos*. Un **nombre de ruta absoluto** comienza en la raíz y sigue una ruta descendente hasta el archivo especificado, indicando los nombres de los directorios que componen la ruta. Un **nombre de ruta relativo** define una ruta a partir del directorio actual. Por ejemplo, en el sistema de archivos con estructura de árbol de la Figura 10.9, si el directorio actual es *root/spell/mail*, entonces el nombre de ruta relativo *prt/first* hará referencia al mismo archivo que el nombre de ruta absoluto *root/spell/mail/prt/first*.

Permitir a un usuario definir sus propios subdirectorios hace que el usuario pueda imponer una estructura a sus archivos. Esta estructura puede dar como resultado que se utilicen directorios separados para los archivos asociados con diferentes temas (por ejemplo, el subdirectorio que nosotros creamos para almacenar el texto de este libro) o con diferentes tipos de información (por ejemplo, el directorio *programs* puede contener programas puente; el directorio *bin* puede almacenar todos los binarios, etc.).

Una decisión interesante de política dentro de un directorio con estructura de árbol es la que se refiere a qué hacer si se borra un directorio. Si el directorio está vacío, podemos simplemente borrar la entrada correspondiente dentro del directorio que lo contuviera. Sin embargo, suponga que el directorio que hay que borrar no está vacío, sino que contiene varios archivos o subdirectorios. Podemos adoptar una de dos soluciones. Algunos sistemas, como MS-DOS, no permiten borrar un directorio a menos que esté vacío; por tanto, para borrar un directorio, el usuario debe primero borrar todos los archivos contenidos en ese directorio. Si existen subdirectorios, este procedimiento debe aplicarse recursivamente a los mismos, para que también puedan ser borrados. Esta técnica puede dar como resultado que haya que realizar una gran cantidad de trabajo. Otra técnica alternativa, como la adoptada por el comando `rm` de UNIX, consiste en proporcionar una opción: cuando se hace una solicitud para borrar un directorio, también se borran todos los archivos y subdirectorios de dicho directorio. Las dos técnicas son fáciles de implementar y la decisión entre una y otra es cuestión de la política que se adopte. La última de las dos políticas mencionadas es más cómoda, pero también es más peligrosa, porque puede eliminarse una estructura de directorios completa con un único comando. Si se ejecutara por error ese comando, puede que fuera necesario restaurar un gran número de archivos y directorios (suponiendo que exista una copia de seguridad).

Con un sistema de directorios con estructura de árbol, podemos permitir que los usuarios accedan a los archivos de otros usuarios, además de acceder a los suyos propios. Por ejemplo, el usuario B puede acceder a un archivo del usuario A especificando su nombre de ruta; el usuario B puede especificar un nombre de ruta relativo o absoluto. Alternativamente, el usuario B puede cambiar el subdirectorio actual para situarse en el directorio del usuario A y acceder al archivo simplemente proporcionando su nombre.

Una ruta a un archivo en un directorio con estructura de árbol puede ser más larga que las rutas típicas de los directorios en dos niveles. Para permitir a los usuarios acceder a los programas sin tener que recordar esos largos nombres de ruta, el sistema operativo Macintosh automatizaba la búsqueda de los programas ejecutables. Este sistema mantiene un archivo, denominado *Desktop File*, que contiene los nombres y ubicaciones de todos los programas ejecutables que ha encontrado. Cuando se añade un nuevo disco duro o disquete al sistema, o cuando se accede a la red, el sistema operativo recorre la estructura de directorios en busca de programas ejecutables que pueda haber en el dispositivo y almacena la información pertinente. Este mecanismo soporta la funcionalidad de ejecución mediante doble clic que hemos descrito anteriormente. Un doble clic sobre un archivo hace que se lea su atributo de creador y que se explore el archivo *Desktop File* en busca de una correspondencia. Si se encuentra una correspondencia, se inicia el programa ejecutable apropiado, utilizando como entrada el archivo sobre el que se ha hecho clic. La familia Microsoft Windows de sistemas operativos (95, 98, NT, 2000, XP) mantiene una estructura de directorio en dos niveles ampliada, asignando letras de unidad a los dispositivos y a los volúmenes (Sección 10.4).

### 10.3.6 Directorios en un grafo acíclico

Considere dos programadores que estén trabajando en un proyecto conjunto. Los archivos asociados con dicho proyecto pueden almacenarse en un subdirectorio, separándolos de otros proyectos y archivos de los dos programadores. Pero, como ambos programadores son igualmente responsables del proyecto, ambos quieren que el subdirectorio se encuentre dentro de su propio directorio. El subdirectorio común debe, por tanto, ser *compartido*. Cada directorio o archivo compartido existirá en el sistema de archivos en dos (o más) lugares simultáneamente.

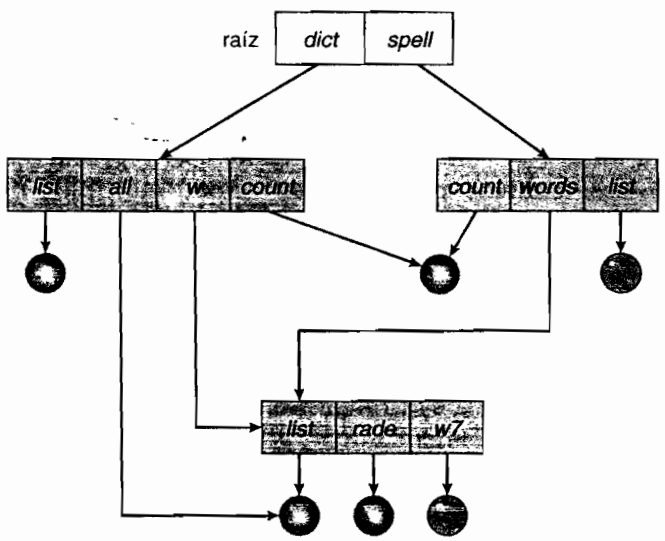


**Figura 10.10** Estructura de directorio en grafo acíclico.

Una estructura en árbol prohibe la compartición de archivos o directorios. Por el contrario, un grafo acíclico (es decir, un grafo sin ningún ciclo) permite que los directorios compartan subdirectorios de archivos (Figura 10.10). El *mismo* archivo o subdirectorio puede estar en dos directorios diferentes. El grafo acíclico es una generalización natural del esquema de directorio con estructura de árbol.

Es importante observar que un archivo (o directorio) compartido no es lo mismo que dos copias del archivo. Con dos copias cada programador puede ver su propia copia en lugar del original, pero si un programador modifica el archivo, esos cambios no se reflejarán en la copia del otro. Con un archivo compartido, sólo existe *un* archivo real, por lo que cualquier cambio realizado por una persona será inmediatamente visible para la otra. La compartición es particularmente importante para los subdirectorios; cada nuevo archivo creado por una persona aparecerá automáticamente en todos los subdirectorios compartidos,

Cuando las personas trabajan en grupo, todos los archivos que quieran compartir pueden colocarse dentro de un directorio. El UFD de cada miembro del grupo contendrá como subdirectorio este directorio de archivos compartidos. Incluso en el caso de un único usuario, la organización de archivos del usuario puede requerir que algún archivo sea colocado en diferentes subdirectorios. Por ejemplo, un programa escrito para un proyecto concreto podría tener que almacenarse tanto en el directorio correspondiente a todos los programas como en el directorio correspondiente a ese trayecto.

Los archivos y subdirectorios compartidos pueden implementarse de diversas formas. Una manera bastante común, ejemplificada por muchos de los sistemas UNIX, consiste en crear una nueva entrada de directorio denominada enlace (*link*). Un **enlace** es, en la práctica, un puntero a otro archivo o subdirectorio. Por ejemplo, un enlace puede implementarse como un nombre de ruta absoluto o relativo. Cuando se realiza una referencia a un archivo, exploramos el directorio y, si la entrada del directorio está marcada como enlace, entonces se incluye el nombre del archivo real dentro de la información del enlace. Para **resolver** el enlace, utilizamos el nombre de ruta con el fin de localizar el archivo real. Los enlaces pueden identificarse fácilmente por el formato que tienen en la entrada de directorio (o bien porque tengan un tipo especial en aquellos sistemas que soporten los tipos) y son, en la práctica, punteros indirectos nominados. El sistema operativo ignora estos enlaces a la hora de recorrer los árboles de directorio, con el fin de preservar la estructura acíclica del sistema.

Otra técnica común para la implementación de archivos compartidos consiste simplemente en duplicar toda la información acerca de esos archivos en todos los directorios que compartan esos archivos. Así, ambas entradas serán idénticas. Un enlace es, claramente, diferente de la entrada original de directorio, por lo que ambas entradas no son iguales; sin embargo, las entradas de

directorio duplicadas hacen que el original y la copia sean indistinguibles. El principal problema con las entradas de directorio duplicadas es el de mantener la coherencia cuando se modifica un archivo.

Una estructura de directorio en forma de grafo acíclico es más flexible que una estructura simple en árbol, pero también es más compleja. Es necesario prestar cuidadosa atención a determinados problemas. Los archivos podrán tener ahora múltiples nombres de ruta absoluta. En consecuencia, puede haber nombres de archivo distintos que hagan referencia a un mismo archivo. La situación es similar al problema de los alias en los lenguajes de programación. Si estamos intentando recorrer el sistema de archivos completo (para encontrar un archivo, para acumular estadísticas sobre todos los archivos o para copiar todos los archivos en un dispositivo de copia de seguridad) este problema cobra una gran importancia, ya que no conviene recorrer las estructuras compartidas más de una vez.

Otro problema es el que se refiere al borrado. ¿Cuándo puede desasignarse y reutilizarse el espacio asignado a un archivo compartido? Una posibilidad consiste en eliminar el archivo cuando un usuario cualquiera lo borre, pero esta acción puede dejar punteros colgantes al archivo que ha dejado de existir. El problema puede complicarse si los punteros de archivo restantes contienen direcciones reales de disco y ese espacio se reutiliza subsiguientemente para otros archivos; en ese caso, esos punteros colgantes pueden apuntar a un lugar cualquiera de esos nuevos archivos.

En un sistema en el que la compartición se implemente mediante enlaces simbólicos, la situación es algo más fácil de manejar. El borrado de un enlace no tiene por qué afectar al archivo original, ya que sólo se elimina el enlace. Si lo que se elimina es la propia entrada del archivo, se desasignará el espacio del archivo, dejando que los enlaces cuelguen. Podemos buscar estos enlaces y eliminarlos también, pero a menos que se mantenga con cada archivo una lista de los enlaces asociados esta búsqueda puede ser bastante costosa. Alternativamente, podemos dejar esos enlaces hasta que se produzca un intento de utilizarlos, en cuyo momento podemos determinar que el archivo del nombre indicado por el enlace no existe y que no podemos resolver el nombre del enlace; el acceso se tratará exactamente igual que se haría con cualquier otro nombre legal de archivo (en este caso, el diseñador del sistema debe considerar cuidadosamente qué hacer cuando se borra un archivo y se crea otro archivo del mismo nombre, antes de que se utilice un enlace simbólico al archivo original). En el caso de UNIX, cuando se borra un archivo se dejan los enlaces simbólicos y es responsabilidad del usuario darse cuenta de que el archivo original ya no existe o ha sido sustituido. Microsoft Windows (todas las versiones) utiliza la misma técnica.

Otra técnica de borrado consiste en preservar el archivo hasta que se borren todas las referencias al mismo. Para implementar esta técnica, debemos disponer de algún mecanismo para determinar que se ha borrado la última referencia al archivo. Podríamos mantener una lista de todas las referencias al archivo (entradas de directorio o enlaces simbólicos). Cuando se establece un enlace o una copia de la entrada de directorio, se añade una nueva entrada a la lista de referencias al archivo. Cuando se borra un enlace o una entrada de directorio, eliminamos la correspondiente entrada de la lista. El archivo se borrará cuando se vacíe su lista de referencias al archivo.

El problema con esa técnica es el tamaño variable, y potencialmente grande, de la lista de referencias al archivo. Sin embargo, no es necesario que mantengamos la lista completa, sino que bastaría con mantener sólo un recuerdo del *número* de referencia. Añadir una nueva entrada de directorio o un nuevo enlace hará que se incremente el contador de referencias, mientras que borrar un enlace o una entrada hará que el contador se decremente. Cuando el contador sea 0, podrá borrarse el archivo, ya que no habrá más referencias a él. El sistema operativo UNIX utiliza esta técnica para los enlaces no simbólicos (o enlaces duros), manteniendo un contador de referencias dentro del bloque de información del archivo (o *inodo*; véase el Apéndice A.7.2). Prohibiendo en la práctica que existan múltiples referencias a los directorios, podemos mantener una estructura de grafo acíclico.

Para evitar problemas como los que se acaban de describir, algunos sistemas no permiten que existan enlaces o directorios compartidos. Por ejemplo, en MS-DOS, la estructura de directorio es una estructura de árbol en lugar de un grafo acíclico.

### 10.3.7 Directorio en forma de grafo general

Uno de los problemas más graves que afectan a la utilización de una estructura en forma de grafo acíclico consiste en garantizar que no exista ningún ciclo. Si comenzamos con un directorio en dos niveles y permitimos a los usuarios crear subdirectorios, obtendremos un directorio con estructura de árbol. Es fácil darse cuenta de que la simple adición de nuevos archivos y subdirectorios a una estructura en árbol existente preserva la naturaleza y la estructura de árbol de ese directorio. Sin embargo, si añadimos enlaces a un directorio existente con estructura de árbol, la estructura en árbol se destruye, dando como resultado una estructura en grafo simple (Figura 10.11).

La principal ventaja de un grafo acíclico es la relativa simplicidad de los algoritmos requeridos para recorrer el grafo y para determinar cuándo no existen ya más referencias a un archivo. Necesitamos poder evitar el tener que recorrer las secciones compartidas de un grafo cíclico dos veces, principalmente por razones de rendimiento. Si acabamos de explorar un subdirectorio compartido de gran tamaño en busca de un archivo concreto sin haber llegado a encontrarlo, necesitamos evitar tener que volver a explorar dicho subdirectorio otra vez, ya que esa segunda búsqueda sería una pérdida de tiempo.

Si permitimos que existan ciclos en el directorio, necesitaremos, de la misma manera evitar tener que buscar cualquier componente dos veces, por razones tanto de corrección como de rendimiento. Un algoritmo mal diseñado podría provocar un bucle infinito que estuviera explorando el ciclo continuamente, sin terminar nunca. Una solución consiste en limitar arbitrariamente el número de directorio a los que se accederá durante una búsqueda.

Un problema similar existe cuando tratamos de determinar si un archivo puede ser borrado. Con las estructuras de directorio en grafo acíclico, un valor de 0 en el contador de referencia significará que ya no hay más referencias al archivo o directorio, y que ese archivo puede ser borrado. Sin embargo, si existen ciclos, el contador de referencias puede no ser 0 incluso aunque ya no sea posible hacer referencia a un directorio de archivo. Esta anomalía resulta de la posibilidad de auto-referencia (o de un ciclo) en la estructura de directorio. En este caso, generalmente necesitamos utilizar un esquema de recolección de memoria para determinar cuándo se ha borrado la última referencia y que, en consecuencia, puede reasignarse el espacio de disco. La recolección en memoria implica recorrer todo el sistema de archivos, marcando todos aquellos elementos a los que se pueda acceder. Después, una segunda pasada recopila todo aquello que no esté marcado, incluyéndolo en una lista de espacio libre (puede utilizarse un procedimiento similar de marcación para garantizar que todo recorrido o búsqueda visitará cada elemento del sistema de archivos una vez y sólo una vez). Sin embargo, la recolección de memoria para un sistema de archivos basado en disco consume muchísimo tiempo y raramente se la suele utilizar.

La recolección de memoria sólo es necesaria debido a la posible existencia de ciclos dentro del grafo. Por tanto, es mucho más sencillo trabajar con estructuras de grafo acíclico. La principal difi-

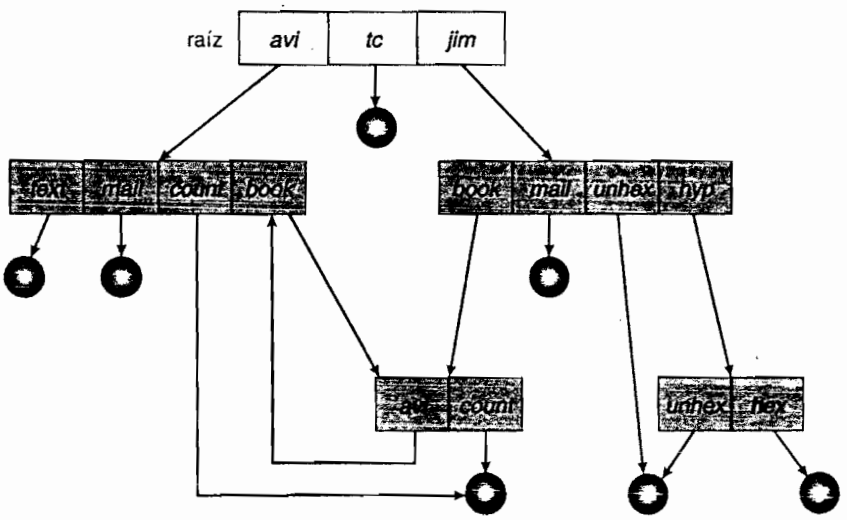


**Figura 10.11** Directorio con forma de grafo general.

cultad es la de evitar que aparezcan ciclos a medida que se añaden nuevos enlaces a la estructura. ¿Cómo podemos saber si un nuevo enlace va a completar el ciclo? Existen algoritmos para detectar la existencia de ciclos en los grafos; sin embargo, estos algoritmos son muy costosos desde el punto de vista computacional, especialmente cuando el grafo se encuentra almacenado en disco. Un algoritmo más simple en el caso especial de directorios y enlaces consiste en ignorar los enlaces durante el recorrido de los directorios. De este modo, se evitan los ciclos sin necesidad de efectuar ningún procesamiento adicional.

## 10.4 Montaje de sistemas de archivos

De la misma forma que un archivo debe *abrirse* antes de utilizarlo, un sistema de archivos debe *montarse* para poder estar disponible para los procesos del sistema. Más específicamente, la estructura de directorios puede estar formada por múltiples volúmenes, que puede montarse para hacer que estén disponibles dentro del espacio de nombres del sistema de archivos.

El proceso de montaje es bastante simple. Al sistema operativo se le proporciona el nombre de dispositivo y el **punto de montaje** que es la ubicación dentro de la estructura de archivos a la que hay que conectar el sistema de archivos que se está montando. Normalmente, el punto de montaje será un directorio vacío. Por ejemplo, en un sistema UNIX, un sistema de archivos que contenga los directorios principales de un usuario puede montarse como */home*; después, para acceder a la estructura de directorios contenida en ese sistema de archivos, podemos anteponer a los nombres de directorio el nombre */home*, como por ejemplo en */home/jane*. Si montáramos ese sistema de archivos bajo el directorio */users* obtendríamos el nombre de ruta */users/jane*, que podríamos utilizar para acceder al mismo directorio.

A continuación, el sistema operativo verifica que el dispositivo contiene un sistema de archivos válido. Para ello, pide al controlador del dispositivo que lea el directorio de dispositivo y verifique que ese directorio tiene el formato esperado. Finalmente, el sistema operativo registra en su estructura de directorios que hay un sistema de archivo montado en el punto de montaje especificado. Este esquema permite al sistema operativo recorrer su estructura de directorios, pasando de un sistema de archivos a otro según sea necesario.

Para ilustrar el montaje de archivos, considere el sistema de archivos mostrado en la Figura 10.12, donde los triángulos representan subárboles de directorios en los que estamos interesados. La Figura 10.12(a) muestra un sistema de archivos existente, mientras que la Figura 10.12(b) muestra un volumen no montado que reside en */device/dsk*. En este punto, sólo puede accederse a los archivos del sistema de archivos existente. La Figura 10.13 muestra los efectos de mostrar en

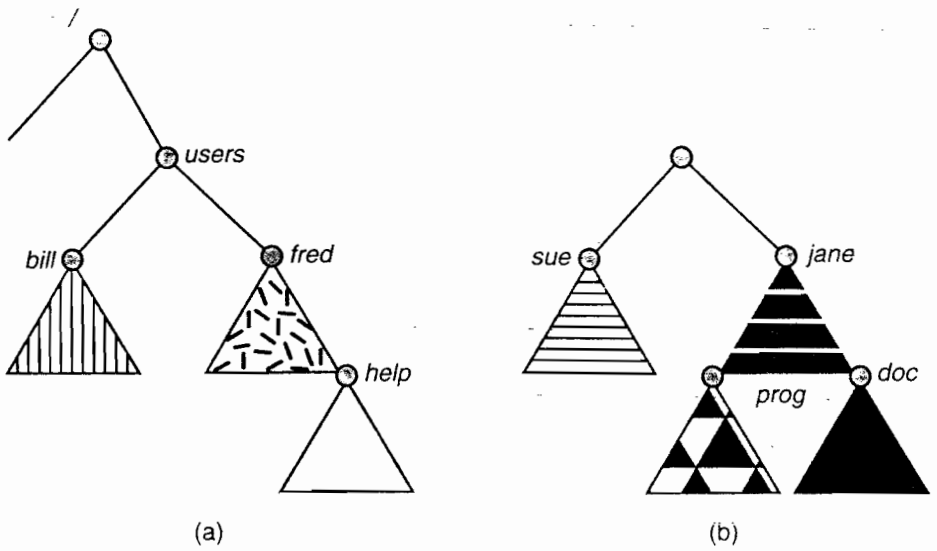


**Figura 10.12** Sistema de archivos. (a) Sistema existente. (b) Volumen no montado.

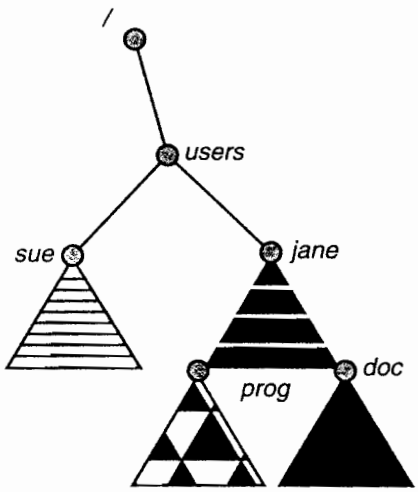


**Figura 10.13** Punto de montaje.

*/users* el volumen que reside en */device/dsk*. Si se desmonta el volumen, el sistema de archivos vuelve a la situación mostrada en la Figura 10.12.

Los sistemas imponen una cierta semántica para clarificar la funcionalidad. Por ejemplo, un determinado sistema podría prohibir que se realizara un montaje en un directorio que contuviera archivos, o bien podría hacer que el sistema de archivos montado estuviera disponible en dichos directorios y ocultar los archivos existentes en dicho directorio hasta que el sistema de archivos se desmontara, en cuyo momento se prohibe el uso de ese sistema de archivos y se permite el acceso a los archivos originales contenidos en el directorio. Otro ejemplo, un determinado sistema podría permitir que se montara repetidamente el mismo sistema de archivos, en diferentes puntos de montaje; o, por el contrario, podría permitir un único montaje por cada sistema de archivos.

Consideremos el ejemplo del sistema operativo Macintosh. Cuando el sistema encuentra un disco por primera vez (los discos duros se localizan en el momento de iniciar el sistema, mientras que los disquetes se detectan cuando se insertan en la unidad), el sistema operativo Macintosh busca un sistema de archivos en el dispositivo. Si encuentra uno, monta automáticamente el sistema de archivos en el nivel raíz, añadiendo a la pantalla un icono de carpeta cuya etiqueta coincide con el nombre del sistema de archivos (tal como esté almacenado en el directorio del dispositivo). El usuario podrá entonces hacer clic sobre el icono y mostrar el sistema de archivos recién montado.

La familia Microsoft Windows de sistemas operativos (95, 98, NT, 2000, XP) mantiene una estructura ampliada de directorio en dos niveles, en la que a los dispositivos en los volúmenes se les asignan letras de unidad. Los volúmenes tienen una estructura de directorio en forma de grafo general asociada con la letra de la unidad. La ruta para un archivo específico toma la forma *letra-unidad:\ruta\al\archivo*. Las versiones más recientes de Windows permiten montar un sistema de archivos en cualquier punto del árbol de directorios, al igual que en UNIX. Los sistemas operativos Windows descubren automáticamente todos los dispositivos y montan todos los sistemas de archivo localizados en el momento de iniciar el sistema. En algunos sistemas, como UNIX, los comandos de montaje son explícitos. Un determinado archivo de configuración del sistema contiene una lista de los dispositivos y puntos de montaje para realizar el montaje automático en el momento de iniciar el sistema, pero pueden ejecutarse otras operaciones de montaje manualmente.

Los temas relativos al montaje de sistema de archivos se analizan con más profundidad en la Sección 11.2.2 y en el Apéndice A.7.5.

## 10.5 Compartición de archivos

En las secciones anteriores, hemos explorado los motivos que existen para compartir archivos y algunas de las dificultades que surgen al permitir a los usuarios compartir archivos. Dicha com-