El sistema de ficheros

José Ramón Herrero Zaragoza
Teodor Jové Lagunas
Enric Morancho Llena

PID_00169385

UOC
Universitat Oberta de Catalunya

Índice

Introducción

Este módulo didáctico se centra en el estudio del **sistema de ficheros** (**SF**). Con este objetivo realizamos los pasos siguientes:

1) En primer lugar, estudiaremos el concepto de **fichero** como dispositivo lógico.

2) En segundo lugar, analizaremos el **sistema de ficheros** como gestor de objetos.

3) Finalmente, veremos el caso concreto del **sistema de ficheros en UNIX, en Mac OS y en Windows**.

Los tres aspectos se estudian siempre desde la perspectiva de la asignatura, que es el punto de vista del usuario del sistema operativo.

Objetivos

Los materiales didácticos de este módulo incluyen las herramientas necesarias para alcanzar los objetivos siguientes:

1. Conocer las funciones del sistema de ficheros.

2. Conocer el concepto de fichero y saber qué características pueden tener los ficheros en función de sus propiedades y de sus tipologías.

3. Entender la necesidad de facilitar el uso por parte de los usuarios. Por ello se utilizan funciones de traducción para acercar los objetos del sistema a las costumbres de los usuarios.

4. Aprender los distintos tipos de espacios de nombres y sus ventajas e inconvenientes.

5. Familiarizarse con las operaciones que ofrecen los sistemas operativos para gestionar los espacios de nombres.

6. Entender los conceptos de seguridad y protección.

7. Ser conscientes de la necesidad de ofrecer herramientas de protección que permitan llevar a cabo distintas políticas de protección en función de las necesidades de los usuarios.

8. Aprender los distintos esquemas de protección y sus ventajas e inconvenientes.

1. Definición del sistema de ficheros

El **sistema de ficheros** (**SF**[1]) es el encargado de gestionar el conjunto de ficheros contenidos en un mismo dispositivo de almacenamiento. Esta gestión se concreta en los tres aspectos fundamentales que indicamos a continuación:

1) La definición de los dispositivos lógicos que configuran los ficheros.

2) El espacio de nombres, que nos servirá para poder localizar y dirigir los ficheros.

3) El control de las acciones que se pueden efectuar sobre los ficheros.

(1)Utilizamos la sigla SF como abreviatura de *sistema de ficheros*.

Ved también

Podéis ver los dispositivos lógicos en el subapartado 4.2 del módulo didáctico 4.

Para ampliar esta definición, diremos que el SF es el encargado de proporcionar un espacio de nombres y un control de acceso a todos los dispositivos. Desde este punto de vista, se puede hablar tanto de ficheros como de dispositivos o, desde una óptica todavía más general, de objetos gestionados por el sistema[2] y visibles por los usuarios.

(2)Objetos gestionados por el sistema, como ficheros, dispositivos, procesos, espacios lógicos, etc.

Es necesario darse cuenta de que cuando dividimos un disco a nivel lógico y lo formateamos para poder guardar diferentes datos o sistemas operativos, lo que hacemos es crear sistemas de ficheros independientes para cada partición. No es el objetivo de este documento entrar a discutir los formatos y las estructuras internas de los distintos tipos de sistemas de ficheros. No obstante, como usuarios debemos ser conscientes de que existen distintos formatos y de que en el momento en el que dividimos un disco y queremos instalar uno o varios sistemas operativos, el administrador debe indicar el tipo de sistema de fichero que se quiere crear. Hay muchos tipos, que van evolucionando a lo largo de los años para proporcionar nuevas funcionalidades o permitir tamaños y sistemas de ficheros mayores. Como ejemplo, citaremos al menos uno de los utilizados en los sistemas operativos más habituales: ext4 en Linux, NTFS en Windows, o HFS+ en Mac Os.

2. El concepto de fichero

2.1. Definición de fichero

Un **fichero** es un dispositivo lógico formado por una agrupación lógica de información almacenada en un dispositivo físico, como un disco, un lápiz USB, o en la memoria, que puede ser manipulada como un todo. La información que incluye tiene en común un conjunto de propiedades que la caracterizan.

2.2. Las propiedades de los ficheros

Un fichero tiene un conjunto de características o propiedades relacionadas con los aspectos siguientes:

1) La **información** relativa al contenido del fichero y a su modificación: tamaño, fecha de creación, última fecha de acceso, última fecha de modificación, tipo de información, etc.

2) La **ubicación** de esta información dentro del dispositivo de almacenamiento. Este aspecto se refiere a la información necesaria para que el sistema pueda localizar el fichero dentro del dispositivo de almacenamiento. En la dinámica general de uso de los ficheros, los usuarios del sistema no necesitan conocer esta información.

3) La **accesibilidad** del fichero. Este aspecto hace referencia a la información relacionada con quién es el usuario propietario del fichero y con las operaciones que los diferentes usuarios pueden hacer: escribir, leer, ejecutar, etc. Todas estas informaciones están relacionadas con el concepto de protección.

Ved también

Podéis ver el concepto de protección en el apartado 4 de este módulo didáctico.

2.3. Los tipos de ficheros

El SO puede reconocer diferentes tipos de ficheros según la estructura de la información que contienen, su finalidad, las operaciones de acceso que permiten, etc. En este módulo didáctico nos centraremos sólo en estas tres primeras características, que están fuertemente interrelacionadas.

Así pues, en un SO podemos encontrar ficheros que contienen caracteres o información en formato binario, gráfico, etc. Podemos encontrar ficheros ejecutables, bibliotecas, fuentes, directorios, documentos, hojas de cálculo, bases de datos, etc. La información contenida en los ficheros siempre tiene una estructura y una finalidad. Estas dos características condicionan la manera de acceder a los ficheros y el tipo de funciones que se pueden llevar a cabo en éstos. En general, sobre los ficheros se pueden hacer el mismo conjunto de operaciones que se pueden llevar a cabo sobre cualquier dispositivo. No obstante, en función de la estructura y la finalidad de esta información, el sistema operativo puede ampliar o restringir el conjunto de operaciones posibles.

Ved también

Podéis ver las operaciones que pueden tener lugar sobre los dispositivos en el subapartado 5.2 del módulo didáctico 4.

Un caso de ampliación de operaciones es, por ejemplo, el que se produce con los **ficheros ejecutables**. Un fichero ejecutable, tal como hemos visto, contiene un tipo de información con una estructura muy determinada que permite al SO cargarla en la memoria como código de un proceso. Así pues, el sistema permite efectuar una acción especial sobre este fichero: la **ejecución**. Esta acción, tal como la acabamos de definir, no tiene sentido sobre ficheros que no sean ejecutables; sin embargo, algunos sistemas pueden redefinir la acción de ejecutar si asocian en la estructura de un fichero una aplicación que pueda interpretarla y/o manipularla.

Ejemplos de redefinición de la acción de ejecutar

A continuación presentamos un par de ejemplos en los que podéis ver la redefinición de la acción de ejecutar:

a) Un ejemplo es el caso en el que al hacer clic sobre el icono de un fichero en una interfaz gráfica, se inicia automáticamente la aplicación asociada capaz de entender la información que contiene el fichero.

b) Otro ejemplo es el caso de los ficheros que incluyen *shell-scripts*. En el sistema UNIX, cuando se lleva a cabo la acción de ejecutar un *shell-script*, el sistema lo detecta y ejecuta el *shell*[3] que es capaz de interpretar la información del fichero que configura los comandos.

(3)El *shell* que debe usarse para interpretar el fichero se indica en el sistema UNIX en la primera línea del fichero, siguiendo el patrón #! nombre_shell. Por ejemplo, #!/bin/bash.

Ved también

Podéis ver el *shell* en el subapartado 7.1.3 del módulo didáctico 4.

Otro caso muy diferente son los **ficheros directorios**, los que tienen una estructura que consiste en una lista de parejas de nombres y números. Su finalidad es dar nombre a los ficheros y permitir su localización. En este caso, el sistema sólo nos deja acceder a los directorios mediante un conjunto muy concreto de operaciones con el fin de proteger la gestión que realiza sobre los ficheros en general.

Ved también

Podéis ver los ficheros directorios en el apartado 3 de este módulo didáctico.

3. El espacio de nombres

Tal como hemos visto en la introducción, una de las **funciones propias del SF** es **proporcionar un espacio de nombres** que permita localizar y manipular los ficheros. Esta visión es extensible a todos los objetos que el sistema gestiona y que deben ser identificados en algún momento por parte de los usuarios. Así pues, de ahora en adelante, en este apartado utilizaremos de manera indistinta los términos *fichero*, *dispositivo* u *objeto*.

3.1. La función de traducción

En un sistema se gestionan multitud de objetos[4]. Para poder gestionarlos todos es necesario cumplir los requisitos siguientes:

(4)Algunos de los objetos que gestionan los sistemas son los ficheros, los dispositivos, los procesos y los espacios lógicos.

- Conocer los objetos que hay y sus características.
- Poder acceder a ellos para interactuar con ellos.
- Poder crearlos y destruirlos.

Para conseguir efectuar estas operaciones, es necesario que los objetos tengan nombre, con el fin de que el sistema y los usuarios los puedan identificar. Dentro del sistema, se identifica cada uno de los objetos mediante lo que podríamos denominar **nombres internos**, que consisten en direcciones de memoria, índices de tablas o números en general.

Además de los ficheros tenemos, por ejemplo, dispositivos con los que los usuarios debemos interactuar directamente. Entonces el sistema ha de proporcionar nombres próximos a nuestros hábitos. Nos es mucho más fácil recordar que el dispositivo con el nombre *impresora* es la impresora, que recordar un número de varias cifras.

Recordar

En el caso del sistema operativo UNIX, el nombre interno está compuesto por la unión de los números llamados *major* y *minor*. Afortunadamente, los usuarios no tenemos por qué saber cuáles son.

El SF es el encargado de proporcionar y gestionar un espacio de nombres que nos permita alcanzar este objetivo.

Una de las funciones del SF es **facilitar una función de traducción** entre unos nombres a los que estamos acostumbrados los humanos y los nombres internos del sistema o, lo que es lo mismo, los objetos tal como los conoce el sistema.

$$F\text{: nombre} \rightarrow \text{objeto.}$$

El conjunto de todos los nombres posibles configura lo que denominados **espacio de nombres**. La función de traducción debe ser unívoca: un nombre sólo puede hacer referencia a un, y sólo un, objeto.

El SO implementa la función de traducción de los nombres mediante estructuras de datos que llamamos **directorios**, los que, a su vez, son implementados mediante ficheros en los que el sistema almacena la estructura de datos que da lugar a la función de traducción. Esta estructura de datos consiste en una tabla que relaciona los nombres que contiene el directorio con los nombres internos del SO (podéis ver la figura 1): cada entrada de la tabla asocia un nombre[5] con la estructura de datos interna que representa en el objeto. Así, un directorio se puede ver como una agrupación de nombres de objetos. Dado que los SF pueden estar contenidos en dispositivos extraíbles, cada SF ha de contener toda la información que configura su espacio de nombres.

[5] El nombre puede ser el de un fichero, un directorio o, en general, de un dispositivo lógico accesible mediante el sistema de ficheros.

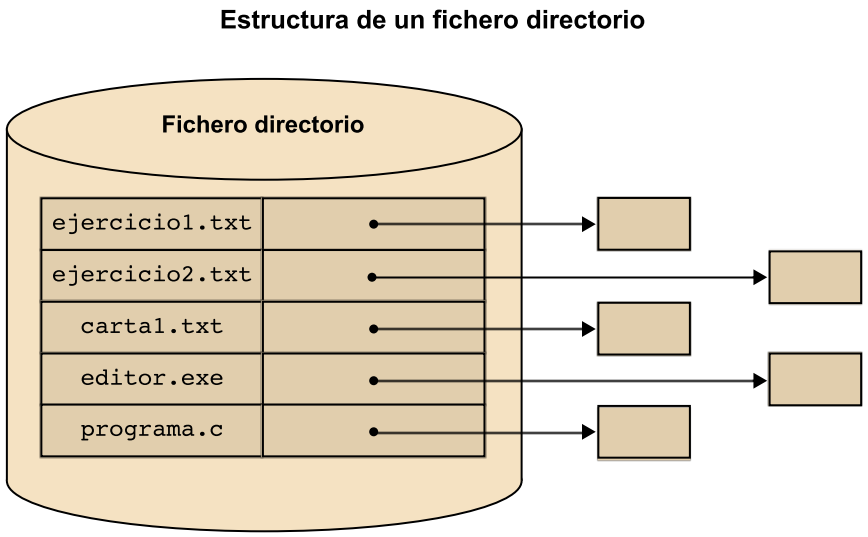

Figura 1

Un sistema operativo puede ofrecer una visión de un único espacio de nombres para todos los objetos y SF, o de un espacio de nombres separado e independiente de los otros para cada tipo de objeto o SF.

Espacio de nombres en UNIX, Mac Os y Windows

Los nombres posibles para los ficheros y directorios son similares en los tres sistemas. En cambio, en UNIX y Mac Os los ficheros y los dispositivos se llaman mediante un mismo espacio, que es gestionado por el SF. Es decir, distintos sistemas de ficheros se pueden montar[6] a partir de un punto determinado de un SF ya existente, de modo que el usuario tiene la visión de un único espacio de nombres, organizado a partir de un directorio que se llama raíz y se representa con el carácter '/'. De hecho, el sistema operativo Mac Os implementa un subsistema compatible con UNIX, con el que no encontraremos muchas diferencias con respecto a UNIX como usuarios del sistema de ficheros. El caso de Windows es distinto, ya que dispositivos y ficheros tienen espacios separados:

- Los dispositivos tienen unos nombres prefijados, como A:, B:, etc. para los discos, y CON:, LPT1:, etc. para el resto de dispositivos.
- En cambio, los ficheros se llaman con el nombre del volumen[7] que los contiene seguido del nombre del fichero, por ejemplo C:\AUTOEXEC.BAT.

[6] Veremos operaciones para montar y desmontar sistemas de ficheros en el subapartado 3.3.

[7] A menudo se llaman volúmenes (volumen A:, volumen C:, etc.).

3.2. La estructura de los espacios de nombres

Este subapartado se centra en los distintos tipos de espacios de nombres que puede tener el SF. Podemos clasificar los espacios de nombres en espacios lineales y jerárquicos, que, al mismo tiempo, pueden estar distribuidos en forma de árbol o en forma de grafo dirigido.

1) El espacio lineal

El espacio lineal es el espacio de nombres más sencillo: tiene una sola dimensión donde todos los nombres están al mismo nivel, contenidos en un único directorio. Dicho de otra manera, no se pueden hacer clasificaciones de los distintos objetos.

Espacios lineales en UNIX y en DOS

Encontramos ejemplos de espacios lineales en los nombres de los volúmenes en Windows (A:, B: ... Z:) o en los identificadores de los procesos o PID (*process identifier* o *process identification number*) que son números enteros. Este número es gestionado por el sistema operativo pero se puede conocer y utilizar fácilmente por los usuarios en sistemas UNIX para hacer tareas de gestión de los procesos que le pertenecen.

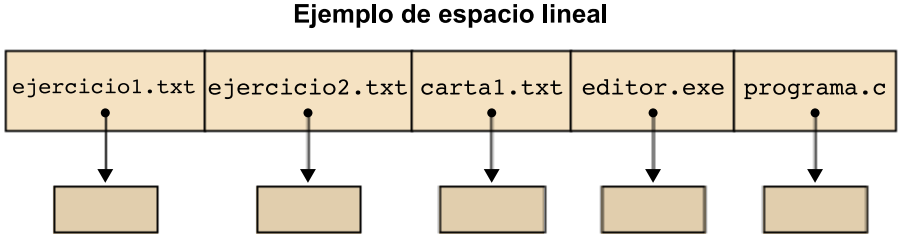

Figura 2

Los espacios lineales resultan buenos para referenciar objetos internos del sistema o para espacios con un número de objetos pequeños.

Así pues, en un sistema monousuario en el que haya pocos ficheros y todos tengan que ser visibles para un único usuario, pueden ser un buen medio de organización. Ahora bien, cuando el número de ficheros crece, los nombres que utilizamos deben ser cada vez más complejos para poder distinguir los

distintos ficheros. Cada vez es más difícil poder recordar qué nombre tenía un fichero concreto. También resulta inadecuado cuando tenemos distintos usuarios. Teniendo en cuenta estas situaciones, el objetivo de proporcionar unos nombres próximos a las costumbres y los hábitos de los humanos es imposible si usamos un espacio lineal de nombres. Por lo tanto, hace muchos años que no se utiliza esta organización lineal para el almacenamiento de los ficheros.

En resumen, con respecto a los **espacios lineales**, podemos decir que son:

- **Adecuados** para espacios con pocos objetos, ya que se trata de una estructura que permite tener una buena visión de conjunto y conseguir una localización rápida de cualquier objeto.

- **Inadecuados** cuando el número de objetos empieza a ser elevado, ya que el significado que para los humanos pueden tener los nombres puede llegar a quedar desvirtuado.

2) Espacio jerárquico

Como usuarios nos gustaría poder agrupar los ficheros por tipo, trabajos o cualquier criterio que nos ayude a organizar el SF. De esta manera sería mucho más fácil localizar los ficheros. El modo de conseguir este objetivo es dotar a los nombres del sistema de ficheros de una estructura jerárquica. Para **conseguir un espacio jerárquico**, tratamos los directorios como objetos con un nombre y, así, pueden formar parte de los objetos que agrupan otros directorios.

Con esta idea podemos organizar los espacios jerárquicos de las dos maneras siguientes:

a) Estructura de árbol. Es la estructura jerárquica más sencilla en la que podemos organizar los ficheros. Un espacio jerárquico en forma de árbol está formado por una jerarquía de directorios con un **directorio raíz** del que cuelgan otros directorios y/o ficheros, los que configuran las **hojas del árbol**.

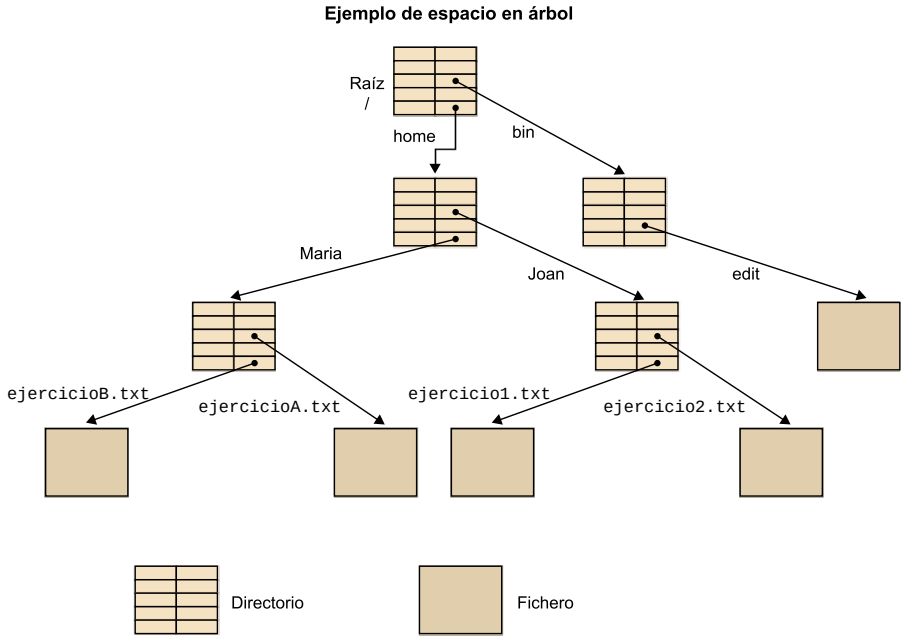

Figura 3

Llamamos **nombre absoluto** al nombre que está formado por la ruta que va desde la raíz, pasando por los diferentes subdirectorios, hasta llegar al fichero. Así pues, el nombre absoluto está compuesto por una secuencia de cadenas de caracteres, o componentes del nombre, contenidas en los subdirectorios que configuran la ruta de acceso al fichero. Los componentes se separan por un carácter delimitador[8].

[8]En UNIX y en Mac Os el carácter delimitador es "/". En Windows, en cambio, el carácter delimitador es "\".

Para facilitar la manipulación de los ficheros en las estructuras jerárquicas se define:

- El **directorio inicial**, que es el directorio desde donde un usuario puede crear su estructura de directorios y donde colocará los ficheros que cree.

- El **directorio de trabajo**, que es el directorio en el que se encuentran los ficheros con los que está trabajando en un instante concreto. Cuando un usuario empieza la sesión de trabajo, el directorio de trabajo es el directorio inicial.

Asociado al concepto de directorio de trabajo está el concepto de **nombres relativos**, que son los nombres formados por el recorrido que va desde el directorio de trabajo hasta el fichero. Distintos ficheros pueden tener el mismo nombre relativo, siempre que éste se obtenga a partir de directorios de trabajo diferentes[9].

[9]Con lo que los nombres absolutos no son iguales.

Como ya hemos dicho, una estructura en árbol soluciona los problemas organizativos que plantea la estructura lineal. No obstante, tiene los dos inconvenientes siguientes:

- No permite compartir de una manera sencilla ficheros entre diferentes usuarios.
- No permite moverse por el árbol de directorios en sentido ascendente, cambiando el directorio de trabajo.

b) Estructura de grafo dirigido. Para solucionar los problemas que plantea la estructura en árbol, definimos el espacio de nombres con una estructura de grafo dirigido como la que se representa en la figura 4.

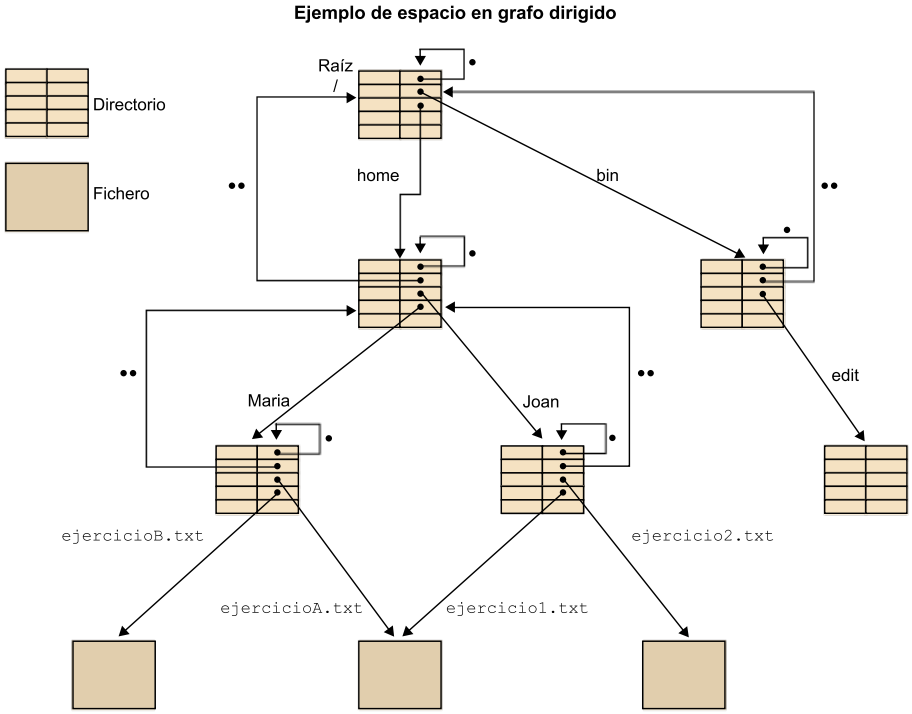

Figura 4

Con la estructura de la figura 4, un fichero puede ser dirigido al mismo tiempo con los nombres absolutos /home/joan/ejercicio1.txt y /home/maria/ejercicioA.txt. De esta manera puede ser compartido fácilmente por más de un usuario.

Ved también

En el subapartado 3.3.2.b introduciremos la operación que permite crear nombres, que nos servirá para este propósito.

En una estructura en grafo dirigido, se puede acceder a un mismo objeto mediante más de un nombre y, por lo tanto, también se puede acceder desde directorios de trabajo distintos.

Con esta estructura, el sistema genera de manera automática nombres adicionales para cada directorio del SF. Por ejemplo, en UNIX y en DOS, en cada directorio, figuran dos nombres especiales: el "." y el "..".

- El nombre "." hace referencia al directorio que lo contiene. Se utiliza para referenciar el directorio de trabajo sin necesidad de conocer el nombre absoluto.

- El nombre ".." es el nombre del directorio "padre" del directorio que lo contiene. Se utiliza para referenciar, mediante nombres relativos, objetos

que tienen como parte de su nombre subdirectorios que se encuentran por encima del directorio de trabajo.

En general, los directorios "." y ".." permiten referenciar objetos mediante nombres relativos sin tener que conocer el directorio de trabajo actual.

Ejemplo de dirección

En el caso de la figura 4, si el directorio de trabajo es /home/maria, se puede acceder al fichero con nombre absoluto /home/joan/ejercicio2.txt con el nombre relativo ../joan/ejercicio2.txt.

El principal inconveniente de los espacios con estructura de grafo dirigido es, como veremos inmediatamente, su gestión.

3.3. Las operaciones sobre el espacio de nombres

Una vez visto lo que es un espacio de nombres, pasamos a las operaciones que podemos realizar en él. El sistema ofrece tres tipos de operaciones: operaciones para manipular el SF como un todo, operaciones para gestionar los contenidos de los directorios y, finalmente, operaciones para manipular el directorio de trabajo. A continuación, trataremos cada uno de estos tipos de operaciones.

3.3.1. Operaciones de manipulación del SF

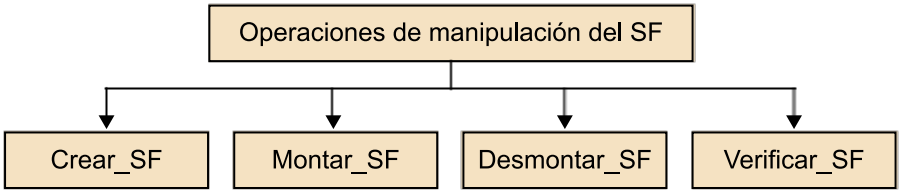

Figura 5

a) Operación *crear_SF*. Como ya hemos visto, un SF está incluido dentro de un dispositivo de almacenamiento y organiza la información de este dispositivo en ficheros. Para que eso sea posible, se debe dotar al dispositivo de un conjunto de estructuras de datos que permitan definir los ficheros, localizarlos mediante un espacio de nombres, crear nuevos, gestionar el espacio libre, etc. La operación *crear_SF* es la encargada de generar todas estas estructuras que se guardarán dentro del mismo dispositivo de almacenamiento. Este hecho permitirá, en caso de que se utilice un dispositivo extraíble, transportar el SF de un sistema a otro sin que haya pérdida de información.

b) Operación *montar_SF*. Algunos SO deben ser informados antes de acceder a un nuevo SF, tanto si es porque se acaba de introducir un volumen nuevo que contiene un SF en un dispositivo extraíble, como si es porque se ha creado un SF nuevo en un dispositivo ya existente, etc. El sistema necesita inicializar una serie de estructuras de datos internos para optimizar los accesos al SF, para reconocer de qué tipo de SF se trata, para restringir el tipo de acceso[10] sobre el SF o para dar, como en UNIX, una visión de un único espacio de nombres, etc. Todas estas acciones las lleva a cabo la operación *montar_SF*. Cabe señalar que esta operación no modifica el SF. Sólo cambia información del SO a la memoria con el fin de acceder de manera automática al punto (directorio) de montaje.

[10]Los tipos de acceso permitidos son de lectura y de escritura.

c) Operación *desmontar_SF*. De manera análoga al caso anterior, antes de retirar el volumen de un dispositivo extraíble en el que se está accediendo como SF, se debe efectuar la operación *desmontar_SF* con el fin de que se liberen las estructuras de datos que se han reservado en la operación de montaje y se actualice el contenido del SF. Es importante que se haga correctamente porque si, por ejemplo, extraemos un lápiz USB sin desmontarlo correctamente, podemos perder los cambios realizados. Eso sucede porque el sistema operativo puede anotar en la memoria los cambios que se deben realizar, retrasando la escritura en el disco para momentos posteriores, con el fin de aumentar el rendimiento, debido a que el acceso a la memoria es normalmente mucho más rápido que el acceso al disco. Si se extrae el dispositivo sin haberse materializado los cambios en el dispositivo, éstos se pierden. Lo mismo puede suceder si un ordenador se queda de repente sin alimentación eléctrica.

d) Operación *verificar_SF*. Finalmente, el sistema nos proporciona herramientas para verificar las estructuras de datos que configuran un SF y que están contenidas en el dispositivo mediante la operación *verificar_SF*. Normalmente estas herramientas no forman parte del núcleo del SO, sino que son aplicaciones y utilidades que se ejecutan sobre el SO.

Así pues, las operaciones de manipulación del SF pueden tener la forma siguiente:

- Estado = crear_SF (*nombre*, *tipos*, etc.).
- Estado = montar_SF (*dispositivo*, *tipo*, *modo L/E*, directorio, etc.).
- Estado = desmontar_SF (*dispositivo*).
- *Verificar_SF* no es una llamada al sistema, sino una utilidad.

3.3.2. Operaciones de manipulación de directorios

Las operaciones de manipulación de directorios nos permiten acceder a la tabla que implementa la función de traducción, que se encuentra almacenada dentro del fichero que configura el directorio.

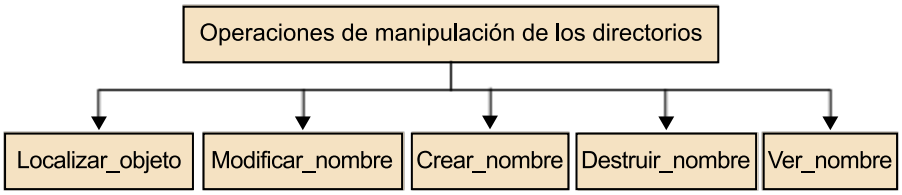

Figura 6

Ved también

Podéis ver la función de traducción en el subapartado 3.1 de este módulo didáctico.

a) Operación *localizar_objeto*. Consiste en, dado un nombre (relativo o absoluto), localizar el objeto al que hace referencia. Según el sistema, puede existir una llamada que efectúe directamente esta operación, o puede ser necesario realizar una búsqueda a partir de una llamada que devuelva una por una las entradas de directorio. La localización de un objeto mediante su nombre es un

procedimiento que forma parte de multitud de llamadas. En general, cualquier llamada al sistema que haga referencia a uno de los objetos contenidos en el SF debe utilizar este procedimiento.

b) Operación *modificar_nombre*. Consiste en localizar la entrada del directorio que contiene el nombre que se debe modificar y, una vez verificado que el nombre nuevo es único (no existe previamente), cambiarlo por el nuevo.

c) Operación *crear_nombre*. Es la encargada de crear un nombre nuevo para un objeto concreto. Esta acción se puede separar en diferentes operaciones en función de si el objeto ya existe y, por lo tanto, se pretende sencillamente crear un nombre nuevo, o si el objeto no existe y, por lo tanto, la creación del nombre se efectúa en el mismo instante de la creación del objeto:

- La **creación de nombres sobre objetos ya existentes**, que se puede ver como el establecimiento de un enlace entre el nombre y el objeto al que hace referencia. Desde esta óptica, la creación de un nombre nuevo consiste simplemente en añadir una línea de información a la tabla del directorio. Antes, sin embargo, se debe localizar el objeto al que se quiere dar un nombre nuevo y se debe comprobar que el nombre sea único. Este tipo de nombre se denomina **enlace físico** (*hard link*), en contraposición a los llamados *enlaces simbólicos*.
  Un **enlace simbólico** (*symbolic link* o *soft link*) es un enlace que no relaciona directamente un nombre con un objeto, sino que lo hace indirectamente mediante un enlace a otro fichero que contiene el nombre del objeto, como se ve en la figura 5 (que podéis ver en la página siguiente). Este tipo de nombres se utiliza fundamentalmente para dar un nombre a un objeto que se encuentra en un SF distinto de donde está el directorio que contiene el enlace simbólico. La existencia de los enlaces simbólicos provoca que la operación *localizar_objeto* deba saber de qué tipo es cada enlace, y actuar en consecuencia.

Nota

Un *hard link* sólo puede apuntar a un objeto del mismo SF. En cambio, un *soft link* puede apuntar a un objeto de otro SF.

- La **creación de nombres junto con los objetos a los que hacen referencia**, que añade la problemática propia de la creación de los objetos en concreto. Se pueden crear ficheros, directorios o dispositivos de distintos tipos. Nosotros nos centraremos en la operación de crear un subdirectorio. En este caso, el sistema ha de proporcionar una operación que, dado el nombre del directorio que se quiere crear y el del directorio desde donde debe colgar, cree el fichero[11] que debe contener el directorio nuevo, su tabla de traducción y los enlaces "." y "..".

[11]Un directorio es un fichero con un formato especial que permite almacenar la tabla de traducción del directorio.

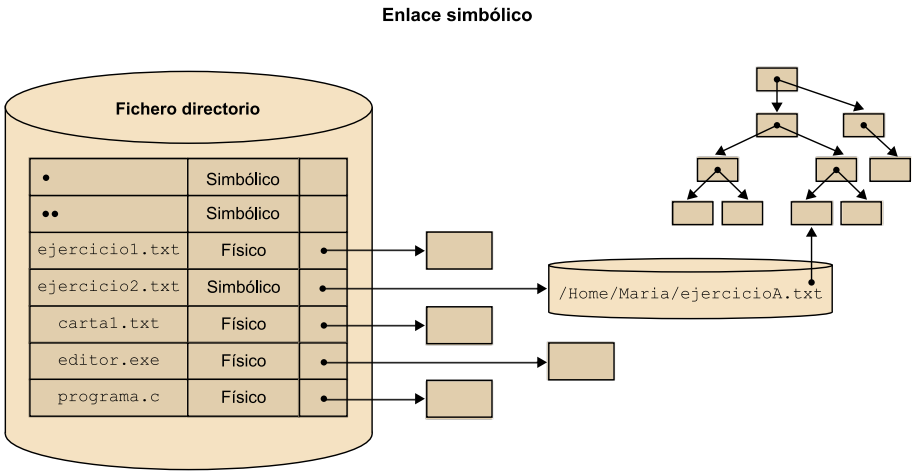

Figura 7

d) Operación *destruir_nombre*. Consiste en localizar el directorio que contiene el último componente del nombre que se quiere destruir y borrar la entrada de la tabla de traducción que lo contiene. Además, si el fichero al que hacía referencia este nombre ya no tiene ningún nombre más asociado, el SO lo destruirá y liberará los recursos que ocupa. Al comprobar si el fichero tiene otro nombre, encontramos las dos situaciones siguientes:

- Detectar si un fichero no tiene ningún nombre más es sencillo si nos referimos a los enlaces físicos, ya que los enlaces físicos existentes están contabilizados en las características del fichero y están incluidos en el dispositivo de almacenamiento que contiene el SF.

- Por contra, la detección de enlaces simbólicos no es trivial, dado que los enlaces simbólicos del fichero que se quiere destruir no tienen por qué estar en el mismo dispositivo, y éste no ha de estar necesariamente montado en el sistema. Por lo tanto, puede resultar imposible detectarlos, y el sistema ignora su existencia durante la operación de destruir.

Como resultado de esto, puede ser que aparezcan enlaces simbólicos que no señalen ningún fichero existente o, todavía peor, que apunten a un fichero de destino distinto al que querían apuntar, pero que ha reutilizado el nombre del fichero destruido. Esto último, sin embargo, puede tener su utilidad práctica porque permite la instalación de nuevas versiones de los ficheros destino de manera transparente al usuario.

En el caso de que se quiera destruir un directorio, es necesario que éste esté vacío. Se considera vacío si sólo tiene las entradas correspondientes al propio directorio "." y al superior ".." dentro del que se encuentra.

e) Operación *ver_nombres*. Permite consultar el contenido de un directorio y también ver el tipo y las propiedades de los objetos a los que hace referencia.

Así pues, las operaciones de manipulación de directorios pueden quedar de la manera siguiente:

- Estado = localizar_objeto (*nombre, información_interna*).
- Estado = modificar_*nombre (nombre_nuevo*, *nombre_viejo*).
- Estado = crear_*nombre (directorio*, *nombre_nuevo*, *simbólico_físico*, *objeto*).
- Estado = destruir_nombre (*nombre*).
- Valores = ver_nombres (*directorio*).

3.3.3. Operaciones de manipulación del directorio de trabajo

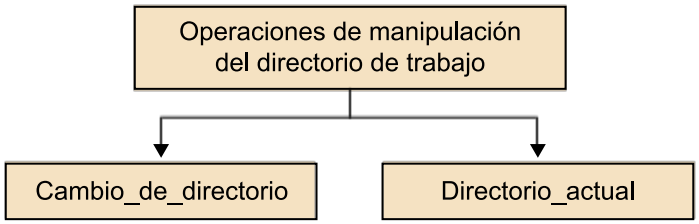

Figura 8

Hay dos operaciones de manipulación del directorio de trabajo: *cambio_de_directorio* y *directorio_actual*. La primera sirve para cambiar el directorio de trabajo. La segunda nos indica en qué directorio estamos trabajando en un instante determinado.

Los parámetros que podrían tener las operaciones de manipulación del directorio de trabajo son los siguientes:

- Estado = cambio_de_directorio (*nuevo_directorio*).
- Directorio_actual = directorio_actual ().

4. La protección

Igual que en el apartado anterior, enfocaremos el tema de la protección desde el punto de vista del acceso a los ficheros, aunque los conceptos y las técnicas que estudiaremos se pueden extender fácilmente al caso general de todos los objetos gestionados por el SO.

Como ya sabemos, un SO gestiona un conjunto de recursos y objetos sobre los que se pueden efectuar acciones. Es responsabilidad del sistema operativo garantizar su buen funcionamiento e impedir que la dinámica de unos usuarios esté afectada por las acciones de otros o por las acciones de agentes externos al sistema computador.

El concepto de **protección** hace referencia al primero de estos aspectos. Es el mecanismo que proporciona el sistema para autorizar o denegar los accesos que en un instante concreto solicitan los usuarios.

El concepto de **seguridad del sistema**, en cambio, se centra básicamente en el segundo aspecto e incluye temas como la identificación y autoidentificación de los usuarios, y también cuestiones administrativas y organizativas relacionadas con la gestión de un sistema informático.

Como es evidente, seguridad y protección son dos conceptos fuertemente relacionados. A pesar de ello, en este módulo nos centraremos en el mecanismo de protección y supondremos que los usuarios han estado debidamente identificados y que, por lo tanto, son realmente quienes dicen que son.

4.1. La protección: concepto y objetivos

> El concepto de protección se refiere al control que lleva a cabo el SO sobre las distintas maneras que tienen los usuarios de acceder a los objetos del sistema.

Hay casos en los que no es necesaria la existencia de un mecanismo de protección:

a) El estudio de la protección tiene sentido desde el momento en el que hay más de un usuario en el sistema, y éstos pueden decidir cómo y con quién comparten los ficheros que crean. En consecuencia, la protección no tiene sentido en un sistema donde hay un sólo usuario. En este caso podemos admi-

tir que no es necesario ningún tipo de control sobre los accesos que se efectúan a los ficheros. Todos los ficheros pertenecen al mismo usuario y él controla el uso que hace de ellos.

b) Un caso similar es el de un sistema multiusuario en el que se deja que todos los usuarios puedan acceder a todos los ficheros. No existe el concepto de propiedad de un fichero y, por lo tanto, no hay restricciones para acceder a ellos. En esta situación tampoco sería necesario un sistema de protección.

En cambio, sí que se necesita un mecanismo de protección que controle la identidad de los procesos que intentan acceder a los ficheros y a su propiedad cuando cada usuario sólo puede acceder a los ficheros que ha creado, o en el caso general en el que se permite a los usuarios compartir fichero mediante un acceso controlado.

Nos centraremos en este último supuesto más general para analizar los diferentes mecanismos de protección.

Como hemos visto, puede haber diferentes necesidades a la hora de proteger un objeto, que dependen del objeto en concreto, de los usuarios que pueden acceder a él y del tipo de acciones que pueden realizar con él. Por este motivo, el sistema debe proporcionar mecanismos que sean lo suficientemente flexibles a fin de que los usuarios puedan implementar diferentes políticas de protección según sus necesidades.

Antes de ver los mecanismos de protección en los apartados que siguen, debemos analizar los diferentes **elementos que intervienen en la protección**. Son los siguientes:

1) Los **objetos**: son todos aquellos elementos gestionados por el SO sobre los que se pueden efectuar distintas acciones de las que deben ser protegidos. Por ejemplo, son objetos los ficheros, los dispositivos, los directorios, los procesos, etc.

2) Los **dominios**: son los distintos agentes activos del sistema que pueden actuar sobre un objeto. Por ejemplo, son dominios los procesos que pertenecen a un usuario o a un grupo predefinido de usuarios, etc.

3) Los **derechos**: son acciones permitidas por parte de un dominio sobre un objeto. Por ejemplo, son derechos leer, escribir, ejecutar, crear, destruir, etc.

Los mecanismos de protección que analizaremos a continuación son maneras distintas de relacionar los derechos que cada dominio tiene sobre cada objeto.

4.2. La matriz de accesos

La matriz de accesos está formada por tantas filas como dominios (D_i) hay, y tantas columnas como objetos (O_j) hay. Cada celda D_{ij} de la matriz contiene los derechos de acceso del dominio D_i sobre el objeto O_j.

Matriz de accesos											
Dominios	**Objetos**										
	ejercicioA.t	**ejercicioB.t**	**ejercicio1.t**	**ejercicio2.t**	**edit**	**Directorio Joan**	**Dominio Joan**	**Directorio María**	**Dominio María**	**Dominio estudiant**	**Dominio profesores**
	Leer	Leer	Leer	Leer	Leer	Leer	Propietario	Leer			
Joan			Escribir	Escribir	Ejecutar	Escribir		Ejecutar			
			Propietario	Propietario		Ejecutar Propietario					
	Leer	Leer	Leer	Leer	Leer	Leer		Leer	Propietario		
Maria	Escribir	Escribir			Ejecutar			Escribir			
	Propietario	Propietario						Ejecutar Propietario			
Estudiantes	Leer	Leer	Leer	Leer	Leer Ejecutar	Leer Ejecutar		Leer Ejecutar		Propietario	
Profesores					Leer Ejecutar						Propietario

En esta matriz, el dominio Joan tiene derecho de lectura y escritura sobre el objeto ejercicio1.txt; sin embargo, en cambio, sólo tiene derecho de lectura sobre el objeto ejercicioB.txt, y de lectura y ejecución sobre edit. El hecho de tener derechos de escritura y ejecución sobre el directorio Joan le permitirá crear nuevos ficheros en este directorio. El permiso de ejecución sobre un directorio permite llevar a cabo acciones sobre un directorio, como por ejemplo cambiar el directorio actual de trabajo de un proceso a aquel directorio y, siempre que conozcamos el nombre de los ficheros contenidos en el directorio y tengamos permiso para acceder al fichero concreto, mostrar su contenido. Si además

tenemos permiso de escritura sobre el directorio, podremos renombrarlos o crear nuevos. El permiso de lectura sobre el directorio indica que se permite pedir al sistema el contenido del directorio y éste devolverá información sobre los ficheros y subdirectorios que contiene. El derecho de propietario que tiene sobre el fichero ejercicio1.txt le permite modificar cualquier derecho que esté en la columna asociada a este fichero. Por ejemplo, puede dar el derecho de escritura al dominio Maria sobre este fichero.

En la tabla vemos que los dominios aparecen como objetos. El motivo es que un dominio también es un objeto sobre el que se pueden llevar a cabo acciones y, por lo tanto, deben ser sometidas al mecanismo de control.

Para simplificar, no se ha incluido en la tabla el dominio Administrador. Hay que recordar, sin embargo, que el Administrador tiene todos los derechos posibles sobre cualquier objeto y dominio del sistema.

La matriz de accesos es una buena manera de ver las relaciones que hay entre los distintos elementos que intervienen en el mecanismo de protección. No obstante, no es una buena herramienta de trabajo por los motivos siguientes:

- Su volumen crece rápidamente a medida que aumentan el número de usuarios y ficheros del sistema.

- El número de objetos sobre los que un dominio tiene derecho es pequeño en relación con el número total de objetos del sistema y, por lo tanto, la matriz es una estructura muy grande con la mayoría de las celdas vacías.

Para solucionar estos problemas, se han propuesto dos métodos que aparecen al partir la matriz por columnas o por filas. En los próximos subapartados analizaremos los efectos que tienen estas soluciones.

4.3. Las listas de control de accesos (*access control lists*)

Una lista de control de accesos (LCA) es una lista de parejas (*derechos*, *dominio*) asociada a un objeto. Esta lista es el resultado de partir la matriz de accesos en columnas y eliminar todas las celdas vacías. De esta manera, el sistema ahorra espacio y obtiene una estructura más dinámica.

Con este esquema, como la LCA se encuentra asociada a los objetos y no a los procesos, cada vez que un proceso quiere llevar a cabo un acceso a un objeto se debe verificar si tiene derecho a ello recurriendo a la LCA. Eso quiere decir que en una sesión de trabajo con un fichero se verifican los derechos en cada operación de lectura o escritura que se quiere realizar. Eso representa un volumen de sobrecarga importante para el sistema.

Ved también

En el apartado 4.5 veremos cómo se puede resolver el problema de la sobrecarga.

Una ventaja de este esquema es que todos los derechos que tienen los dominios sobre un objeto se encuentran en un mismo lugar, la LCA. Esta circunstancia genera que la asignación y la revocación de derechos por parte del propietario del objeto se puedan realizar de manera sencilla.

4.4. Las listas de *capabilities* (capacidades)

Una lista de *capabilities* es una lista de parejas *(objeto*, *derechos*) asociada a un dominio, cada una de las cuales se denomina *capability*. Las listas de *capabilities* son el resultado de partir la matriz de accesos en filas y eliminar, igual que en las listas de control de acceso, todas las celdas vacías.

Las listas de *capabilities* son estructuras de datos asociadas a procesos. El sistema las puede gestionar de las dos maneras siguientes:

a) Protegiéndolas en su espacio y proporcionando llamadas con el fin de que los procesos las puedan utilizar.

b) Dejando que los procesos las gestionen directamente. En este caso las *capabilities* se protegen mediante técnicas de cifrado que evitan que se modifiquen los derechos que contienen.

En cualquiera de los dos casos, cuando un proceso accede a un objeto, ha de presentar al sistema la *capability* que le da derecho a llevarlo a cabo. El sistema verifica su validez y efectúa el acceso. El acceso a este mecanismo es más ágil que en las LCA.

En cambio, la modificación de los derechos que tienen los dominios sobre un objeto por parte del propietario no es trivial, ya que los derechos se encuentran repartidos por todos los procesos del sistema y, por lo tanto, en el caso de querer revocar un derecho de todos los dominios, se debe realizar un recorrido por todos y eliminar las *capabilities* asociadas.

4.5. Mejoras y modelos combinados

Como acabamos de ver, los modelos de LCA y de *capabilities* tienen algunos inconvenientes. Para solucionarlos, la mayoría de los sistemas utilizan una combinación de los dos modelos que permite disfrutar de las ventajas de cada esquema.

Como mecanismo básico utilizan las LCA, que se suelen presentar haciendo una reducción de todos los dominios posibles a unos cuantos. Esto se consigue agrupando los dominios, como por ejemplo en UNIX, donde hay tres grupos:

el propietario (por ejemplo Joan), todos los dominios asociados al grupo de trabajo del propietario (por ejemplo Alumnos) y el resto de dominios del sistema.

Los primeros accesos a cada objeto se verifican mediante las LCA. Este primer acceso se puede concretar, por ejemplo, en la operación *abrir* que se efectúa al iniciar una sesión de trabajo con un fichero o un dispositivo. Hay que recordar que en el momento de abrir el objeto se especifica el tipo de acceso que se desea realizar. Por lo tanto, es posible comprobar si el proceso que realiza la operación tiene permisos para llevar a cabo este tipo de acceso. Una vez verificado el derecho, el sistema genera una *capability* que se utilizará durante la sesión de trabajo. La *capability* generada permitirá realizar únicamente el tipo de operación indicado en el momento de abrir el objeto. Normalmente, esta *capability* se asocia al dispositivo virtual que se genera durante la operación *abrir*. Cada vez que se realiza una operación sobre el dispositivo virtual, se comprueba si la *capability* asociada a él lo permite. Esto, por ejemplo, permite evitar que se intente escribir un fichero que se ha abierto sólo para lectura. Una vez finalizada la sesión de trabajo, la *capability* se destruye con la operación *cerrar*. De esta manera se comprueba cada uno de los accesos de lectura y/o escritura que se realizan durante la sesión de trabajo sin necesidad de acceder a la LCA por cada acceso[12].

(12)Acceder a la *capability* asociada al dispositivo virtual sobre la que se intenta realizar una operación de lectura o escritura es mucho más rápido que acceder a la LCA asociada al fichero.

Ved también

En el módulo didáctico 4 podéis ver la operación *abrir* (en el subapartado 5.2.2) y los dispositivos virtuales (en el subapartado 4.3).

Otra mejora, en este caso en el mecanismo de *capabilities*, consiste en dotarlo de una herramienta de revocación eficiente. Una posibilidad es el llamado **mecanismo de cerradura y clave**. En este esquema cada *capability* tiene asociado un número denominado **clave**. Los objetos tienen asociada una colección de números llamados **cerraduras**. Para que una *capability* sea correcta, la clave ha de coincidir con alguna de las cerraduras que tiene el objeto. Para revocar un conjunto de derechos, lo único que debe hacer el propietario del objeto es eliminar una o más cerraduras. Evidentemente, este mecanismo no tiene la flexibilidad de las LCA, pero soluciona parcialmente el problema de tener que recurrir a todos los dominios para eliminar las *capabilities* asociadas.

Otra alternativa consiste en asociar información temporal a una capability de modo que se establezca una fecha de caducidad de ésta.

5. Ejemplos de sistema de ficheros y protección

5.1. UNIX

5.1.1. El sistema de ficheros en UNIX

En UNIX existen los tipos de ficheros siguientes:

1) Los **ficheros ordinarios**, que son los ficheros tal como los hemos estudiado en este módulo.

2) Los **ficheros directorio**, que son los que configuran la función de traducción del espacio de nombres.

3) Los **ficheros especiales** o **dispositivos**, que como su nombre indica son los dispositivos del sistema.

4) Los ***soft links*** o ***symbolic links***, que como su nombre indica permiten realizar enlaces simbólicos a otros ficheros.

El sistema operativo UNIX reconoce un único tipo de fichero ordinario que percibe todos los ficheros como una secuencia de *bytes*, a los que se accede mediante operaciones específicas. Esta regla sólo tiene una excepción: los ficheros ejecutables. Un fichero ejecutable tiene una estructura muy concreta que el SO reconoce y utiliza a la hora de cargar un programa en la memoria. UNIX utiliza los primeros *bytes* de esta estructura del fichero para dejar una marca que distingue un fichero normal de uno ejecutable:

Ved también

Podéis ver las operaciones de acceso a los ficheros en el subapartado 5.2.1 del módulo didáctico 4 y los ficheros ejecutables en el subapartado 1.3 del módulo didáctico 2.

a) El hecho de que no esté la marca garantiza que el fichero no es ejecutable.

b) La existencia de la marca, sin embargo, no garantiza que el fichero sea ejecutable, ya que puede haber ficheros binarios que por azar tengan la misma combinación de *bytes* al principio.

El sistema de ficheros de UNIX tiene una estructura de grafo dirigido organizada en directorios, tal como hemos visto. Cada fichero puede tener más de un nombre utilizando el mecanismo de enlaces físicos o simbólicos. A la hora de dirigir un fichero se pueden utilizar nombres absolutos o relativos. Cada uno de los nombres de subdirectorios que componen el nombre de un fichero están separados por el carácter "/".

Ved también

Podéis ver el SF con estructura de grafo dirigido en el subapartado 3.2 de este módulo didáctico.

Cada dispositivo de almacenamiento puede contener un SF que se crea mediante la utilidad mkfs. Esta utilidad crea dentro del dispositivo la estructura del directorio, las estructuras de datos necesarios para gestionar el espacio libre y también las estructuras para almacenar las características de los ficheros (denominadas *inodes*), como el espacio que ocupará un fichero, o qué *major* y *minor* se debe utilizar para localizar un dispositivo, quién es su propietario, etc.

Ved también

Podéis ver el *mayor* y el *menor* en el apartado 7.1 del módulo didáctico 4.

Aunque cada dispositivo que contiene un SF tiene su propia estructura de directorios, UNIX da la visión de un espacio en forma de grafo totalmente conectado. Ello significa que hay un único directorio raíz, y que siempre hay un camino desde este directorio raíz hasta cualquier fichero, como se ve en la figura 9.

Para poder dar esta visión única, el sistema tiene un SF permanentemente accesible[13] sobre el que se irán añadiendo el resto de SF antes de poder acceder a él. La operación mount es la encargada de asociar el directorio raíz del SF que se quiere montar con un directorio de un SF que ya esté montado. Aparte de eso, la operación mount lleva a cabo operaciones de inicialización de estructuras internas del SO que agilizan los accesos.

[13]El SF del sistema se ha montado en el tiempo de inicialización del sistema.

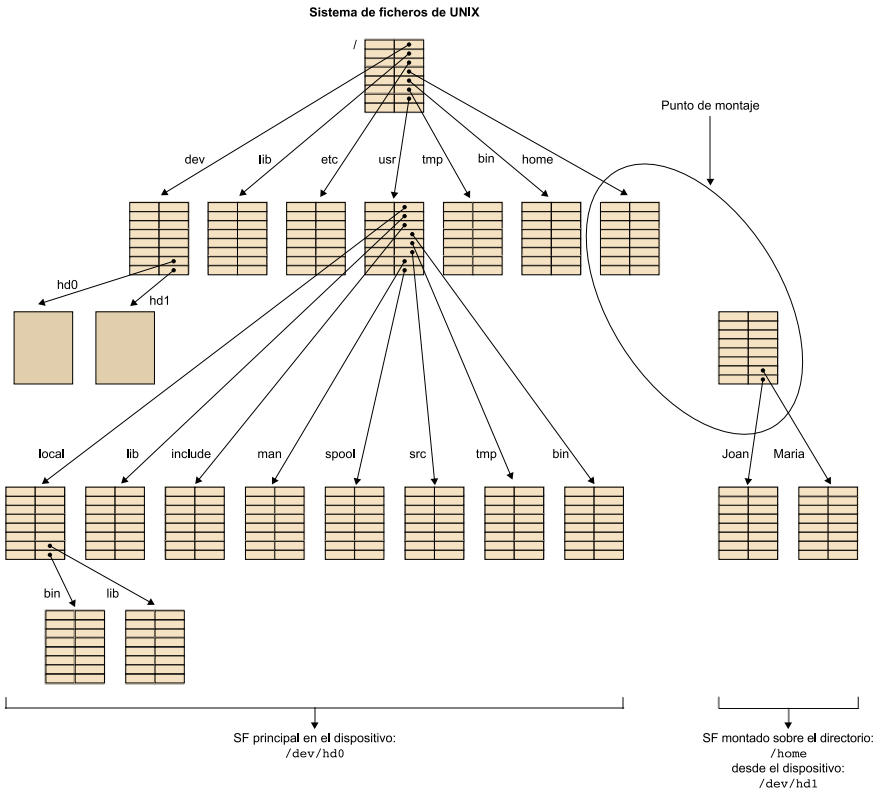

Figura 9

Las llamadas al sistema y los comandos que nos ofrece UNIX para gestionar el SF siguen la misma estructura que hemos descrito en el apartado del espacio de nombres. Es la que podéis ver en la tabla siguiente:

Ved también

Podéis ver el espacio de nombres en el apartado 3 de este módulo didáctico.

Operación	Llamada al SO	Comando	Comentarios
Crear_SF	-	mkfs	-
Montar_SF	mount	mount	-
Desmontar_SF	umount	umount	-
Verificar_SF	-	fsck	-
Localizar_objeto	stat	stat	-

Operación	Llamada al SO	Coman-do	Comentarios
Modificar_nombre	rename	mv	-
Crear_nombre	link	ln	Crean enlaces físicos
	symlink	ln -s	Crean enlaces simbólicos
	mkdir	mkdir	Crean directorios
	mknod	mknod	Crean dispositivos
Destruir_nombre	unlink	rm	Borra ficheros
	rmdir	rmdir	Borra directorios
Ver_nombres	getdents	ls	Ver el contenido de un directorio
Cambio_de_directorio	chdir	cd	-
Directorio_actual	getcwd	pwd	-

5.1.2. El mecanismo de protección de UNIX

Las protecciones de UNIX se basan en el modelo mixto que ya hemos descrito. Nosotros nos centraremos en las LCA y las operaciones que se ofrecen para gestionarlas.

Dominios de protección y bits de permiso

En UNIX los dominios de protección están asociados a usuarios (UID) y a grupos de usuarios (GID). Los grupos de usuarios son dominios en los que se organizan los usuarios del sistema en función de sus características, del trabajo que han de realizar, etc. Un usuario se identifica en el sistema mediante un nombre de usuario y una contraseña. Una vez dentro del sistema, todos sus procesos tendrán asociados los dominios de usuario y de grupo de usuarios

Ved también

Podéis ver el modelo mixto en el subapartado 4.5 de este módulo didáctico.

al que pertenezca. En un momento determinado, un usuario sólo puede pertenecer a un grupo; sin embargo, puede estar autorizado a cambiar de grupo mediante la orden newgrp.

Los dominios de los estudiantes

Todos los estudiantes que estén dados de alta en un sistema tendrán un dominio de usuario propio y podrían estar organizados en un dominio conjunto denominado *estudiantes*.

Todos los ficheros están asociados al dominio de usuario y de grupo de usuarios de su propietario. Para determinar los derechos de un usuario sobre un fichero, UNIX utiliza una LCA con tres dominios: el usuario propietario del fichero, los usuarios que pertenecen al mismo grupo que el propietario y, finalmente, cualquier otro usuario. Para cada uno prevé básicamente tres derechos diferentes: de lectura, de escritura y de ejecución.

La visualización de las LCA se lleva a cabo mediante tres grupos de tres caracteres cada uno, uno para cada derecho, como se muestra en la figura 10. El primer grupo hace referencia al propietario; el segundo, al grupo del propietario, y el tercero, al resto. Las letras r, w y x (lectura, escritura y ejecución) determinan si se tiene el derecho. El comando chmod permite al propietario conceder o revocar este derecho.

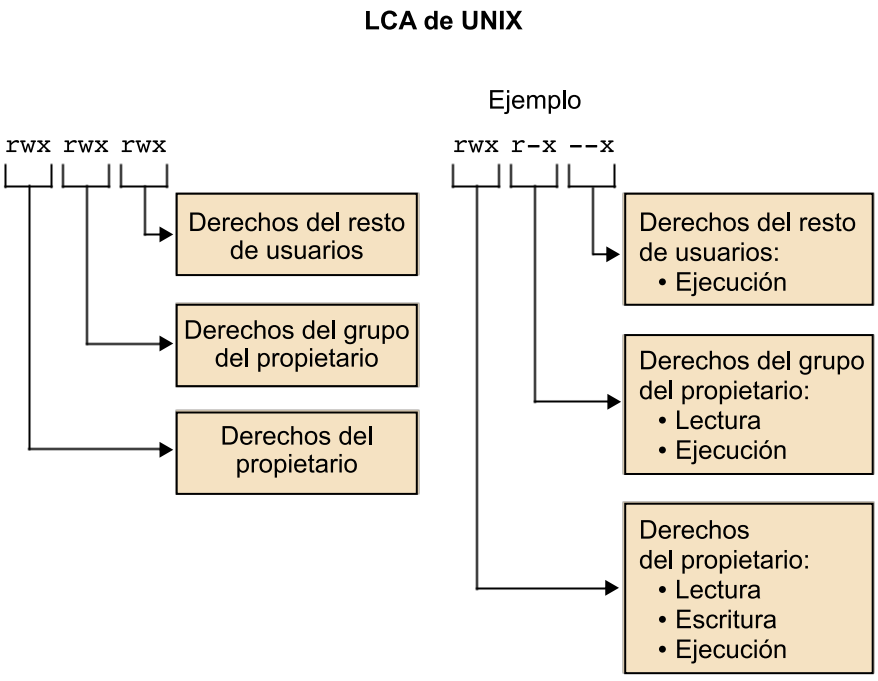

Figura 10

El derecho de ejecución puede ser reforzado con el derecho setuid o con el derecho setgid, que permiten que durante la ejecución de un programa el proceso que lo ejecuta cambie respectivamente al dominio del usuario propietario o al del grupo del usuario propietario del fichero ejecutable. Estos derechos permiten construir aplicaciones que acceden de manera controlada a bases de datos o ficheros en general sobre los que no se quiere dar un derecho de escritura generalizado.

Como hemos comentado en el apartado 4.2, el significado de los derechos r, w y x varía si el fichero es un directorio. En este caso, el derecho de lectura permite visualizar los nombres contenidos en el directorio, el derecho de escri-

tura (condicionado a tener también permiso de ejecución sobre el directorio) permite añadir nombres nuevos al directorio, renombrar ficheros o borrarlos y, finalmente, el derecho de ejecución posibilita que el usuario utilice el directorio como un directorio de trabajo.

Por defecto, el valor de los bits asignados a un fichero o directorio en el momento de su creación corresponde a los bits resultantes de aplicar una máscara llamada *umask* que forma parte del entorno en ejecución de cada proceso y que puede ser consultada y/o modificada mediante el comando umask y la llamada al sistema homónimo. Posteriormente, los bits que controlan los permisos pueden ser modificados, si se tienen derechos para hacerlo, mediante la llamada y el comando chmod.

LCA de POSIX

El estándar POSIX define también LCA, que es un superconjunto de los permisos especificados por los bits de permiso asociados a los ficheros para los dominios propietario, grupo y resto de usuarios. Por una parte, hay una correspondencia directa con respecto a los permisos indicados por los bits de permiso al propietario, su grupo y el resto de usuarios. Un cambio en uno de ellos deriva en la modificación de las entradas de la LCA de POSIX y viceversa. Pero, por otra parte, las LCA de POSIX permiten especificar de manera mucho más fina los permisos que el propietario de un fichero o directorio quiere conceder a otros usuarios o grupos concretos.

5.1.3. Integración al sistema de ficheros de dispositivos e información del sistema

Existen algunos objetos que aparecen en el SF de algunas variantes del sistema operativo UNIX que tienen apariencia de ser ficheros ordinarios pero que en realidad no lo son. Por ejemplo, muchos dispositivos pueden ser accedidos y manipulados en UNIX mediante unos ficheros especiales llamados ficheros de dispositivo (*device files*). Estos ficheros de dispositivo se encuentran en el directorio /dev y permiten acceder al código gestor del dispositivo (*device driver*) utilizando llamadas al sistema de Entrada/Salida. Se puede acceder a dispositivos que existen físicamente, por ejemplo particiones de un disco (como puede ser /dev/sdal), una impresora (como puede ser /dev/lp0) o un terminal (/dev/tty1). Y también se puede acceder a pseudo-dispositivos que no existen físicamente pero que sirven para proporcionar una determinada funcionalidad. Por ejemplo, el dispositivo nulo /dev/null que acepta y descarta todo aquello que se le envía, o el dispositivo /dev/zero, que genera ceros.

A menudo existe el pseudo-sistema de ficheros procfs, que permite obtener información, y en algunos casos manipular, sobre los procesos del sistema. Mediante los subdirectorios y ficheros que se encuentran a partir de /proc el usuario puede acceder a mucha información sobre los procesos que se están

ejecutando en la máquina. Hay que advertir que todos estos directorios y ficheros no se almacenan en ningún disco, sino que se trata de información que proporciona el sistema operativo en el momento en el que se solicita.

Otro caso de interés es el sysfs, que permite configurar distintos aspectos del sistema, principalmente en lo referente a *drivers* y dispositivos mediante los directorios que aparecen a partir de /sys. En este caso tampoco se guarda nada en el disco, sino que todo está almacenado en la memoria.

5.2. Windows

Existen distintos formatos para almacenar la información en un sistema Windows. El más habitual hoy día en los discos de los ordenadores que usan Windows es el formato NTFS. Sigue usándose bastante el formato FAT, pero sobre todo para memorias USB, ya que este formato puede ser utilizado en muchos dispositivos multimedia. El formato FAT es más simple que NTFS: permite tamaños de ficheros más pequeños que NTFS y no incorpora demasiados mecanismos de protección. Aunque algunas características son comunes a los dos formatos, en este apartado asumimos que trabajamos con NTFS.

A diferencia de UNIX, en el caso de Windows el usuario no tiene la visión de un único sistema de ficheros. Existe un espacio de nombres lineal para indicar diferentes *volúmenes*, donde un volumen da acceso a un dispositivo concreto. Los nombres posibles se identifican con una letra mayúscula seguida del carácter ":". Por ejemplo A:, C:, etc. El usuario indica explícitamente el volumen en el que quiere trabajar.

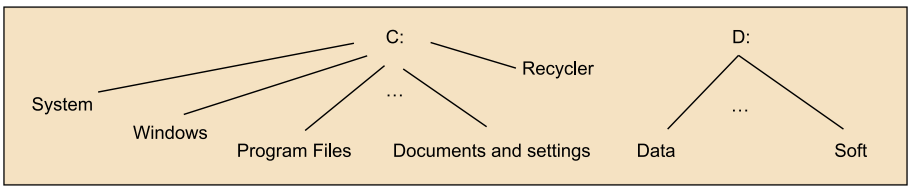

Figura 11

Los nombres posibles para los ficheros y directorios son similares al caso de UNIX. En Windows, sin embargo, los directorios se separan por el carácter "\". Por ejemplo C:\Windows\Documents and Settings.

Es posible crear enlaces físicos (*hard links*) con el comando fsutil o la llamada CreateHardLink del API estándar de Windows. Se pueden crear enlaces simbólicos (*soft links*) creando accesos directos (*shortcuts*) utilizando la opción que aparece en los menús contextuales.

Los sistemas de protecciones de ficheros y directorios (carpetas) está basado en LCA. Se puede manipular desde el intérprete de comandos con el comando cacls o mediante la pestaña correspondiente a la configuración de seguridad

a partir de las propiedades de un fichero o directorio. El administrador puede también definir cuotas de disco con el fin de limitar el espacio de disco máximo que un usuario puede llegar a ocupar.

NTFS incorpora también compresión y cifrado de ficheros, que se pueden encontrar accediendo a las opciones avanzadas de las propiedades de los ficheros. Otros aspectos incluidos son el tratamiento de ficheros *dispersos* con el fin de reducir el espacio ocupado del disco cuando hay cadenas muy largas de ceros y la implementación de mecanismos de *Journaling*[14] que mejoran la consistencia y recuperación del sistema de ficheros en caso de fallos o caídas del sistema, como cuando se produce una interrupción del suministro eléctrico.

(14)Muchos sistemas de ficheros de Linux también soportan *Journaling*.

Resumen

En este módulo hemos estudiado el concepto de **sistema de ficheros**, mediante el que hemos visto que el SO ofrece a los usuarios los aspectos siguientes:

- El concepto de fichero.
- La gestión de los nombres de los ficheros.
- El mecanismo de protección que los ficheros tienen asociado.

Estos dos últimos puntos los hemos extendido a todos los objetos que son gestionados por el SO. Con respecto a los ficheros, hemos visto que el SO puede distinguir los distintos tipos de ficheros y que las operaciones de acceso que se pueden realizar no siempre son las mismas.

Hemos visto que la misión básica de los espacios de nombres es implementar una función de traducción que permita utilizar nombres estructurados y próximos a nuestras costumbres. Para conseguirlo, podemos utilizar básicamente dos esquemas: el **espacio lineal** y el **espacio jerárquico**. Éste último es el más utilizado en su variante de grafo dirigido.

Finalmente, hemos analizado dos **mecanismos de protección**: las listas de control de accesos (LCA) y las *capabilities*. Hemos visto que los dos mecanismos presentan inconvenientes y que la mejor solución es combinarlos, utilizando la LCA para abrir un fichero y generando una *capability* asociada al canal (*file descriptor*) abierto para posteriores comprobaciones al realizar operaciones de lectura o escritura.

Hemos visto también que mediante el sistema de ficheros a veces se puede acceder a ciertos dispositivos, a información sobre los procesos e incluso a configurar el sistema.

Finalmente, hemos comentado algunos aspectos concretos de los sistemas de ficheros más habituales de los sistemas UNIX y Windows.

Actividades

1. Estudiad en el manual qué tipos de SF soporta LINUX.

2. Analizad y probad los comandos de UNIX que hemos presentado en este módulo.

3. Pensad qué llamadas al SO tenemos que utilizar y cómo debemos hacer los comandos analizados en la actividad anterior.

4. Como usuarios de Windows, analizad qué espacio de nombres y qué operaciones tenéis para acceder al sistema de ficheros.

5. Probad a crear un directorio y algún fichero dentro. Utilizando el comando chmod activad y desactivad los bits de protección r, w y x del directorio para el propietario. Para cada combinación de valores intentad leer el directorio, ved el contenido de un fichero contenido en el directorio del que conocemos el nombre, cread nuevos ficheros, movedlos o borradlos, o cambiad el directorio de trabajo a aquel directorio.

6. En un sistema Linux consultad la información disponible sobre las LCA de POSIX en el manual en línea haciendo man acl.

7. En un sistema Windows y trabajando como administrador, definid un nuevo usuario a partir del Panel de Control y Cuentas de Usuario. Cambiad de usuario yendo al menú de Inicio y yendo a la parte de Cambio de Usuario. Por defecto el directorio de *login* propio de cada usuario está protegido de manera que el resto de los usuarios no pueden acceder a su cuenta. Si el administrador desactiva la Opción *Use simple file sharing* de las opciones de carpeta, entonces los usuarios pueden activar o desactivar los permisos de acceso a usuarios concretos mediante la pestaña correspondiente a la configuración de seguridad a partir de las propiedades de un fichero o directorio.

Ejercicios de autoevaluación

1. ¿Cuáles son las características o propiedades principales de un fichero? Comentad cuáles pueden ser sus funciones.

2. Basándoos en lo que hemos visto en este módulo didáctico y en vuestra experiencia, decid qué tipo de espacio de nombres tienen los sistemas de ficheros de los SO siguientes:

a) Windows.

b) UNIX.

3. ¿Qué tipo de enlace representan los ficheros que tienen la característica "acceso directo" de Windows? Justificad la respuesta.

4. En MS-DOS los directorios, aparte de la función de traducción, contenían información con respecto a las características de los ficheros a los que hacen referencia, como el número de caracteres que contiene el fichero. ¿Qué inconvenientes podía tener este hecho?

5. ¿Qué tipo de espacio de nombres creéis que es el más adecuado para los tipos de objetos siguientes?

a) Ficheros.

b) Dispositivos físicos o lógicos.

c) Dispositivos virtuales.

d) Procesos.

e) Usuarios.

6. ¿Cómo sería la matriz de accesos siguiente si...

a) cada usuario sólo pudiera acceder a sus ficheros con todos los derechos (lectura, escritura y ejecución)?

b) todos los usuarios pudieran acceder con todos los derechos a todos los ficheros?

Dominios	**Objetos**							
	Fichero A	**Fichero B**	**Fichero C**	**Fichero D**	**Fichero E**	**Fichero F**	**Fichero G**	**Fichero H**
Joan	Propietario	Propietario						
Maria			Propietario	Propietario				
Josep					Propietario	Propietario		
Marta							Propietario	Propietario

7. ¿En los dos casos del ejercicio 6, es necesario tener un mecanismo de protección? Y en caso de que haga falta, ¿tiene sentido el derecho de propietario? Justificad la respuesta.

8. Decid entre qué momentos de una sesión de trabajo con un fichero se verifican los derechos de acceso si tenemos un sistema operativo con un sistema de protección basado en...

a) LCA.

b) listas de *capabilities*.

Justificad las respuestas.

9. ¿Qué restricciones tiene el mecanismo de protección de UNIX basado en dominios de protección y bits de permiso respecto a un mecanismo de LCA en el que cada usuario constituye en sí mismo un dominio? Justificad la respuesta.

Solucionario

Ejercicios de autoevaluación

1. Un fichero tiene un conjunto de características o propiedades que hacen referencia a los aspectos siguientes:

a) La información relativa al contenido del fichero y a su modificación: tamaño del fichero, fecha de creación, última fecha de acceso, última fecha de modificación, tipo de información, etc. Su función es básicamente estadística, a excepción del tamaño del fichero y del tipo de información. El sistema utiliza el tamaño del fichero para determinar cuándo debe dar la marca de final de fichero durante los accesos de lectura. El tipo de información que contiene el fichero puede condicionar las operaciones que se pueden realizar en él.

b) La ubicación de esta información dentro del dispositivo de almacenamiento. Contiene información necesaria para que el sistema pueda localizar el fichero dentro del dispositivo de almacenamiento.

c) La accesibilidad del fichero. Contiene información relacionada con el usuario propietario del fichero y las operaciones que se pueden hacer en él: escribir, leer, ejecutar, etc. Todas estas informaciones están relacionadas con el mecanismo de protección.

2.a) Windows tiene un espacio de nombres en forma de grafo dirigido donde se pueden formar ciclos porque un fichero puede tener más de un nombre gracias a los enlaces. El espacio es disjunto: cada dispositivo que contiene un sistema de ficheros tiene un espacio de nombres separado.

b) UNIX tiene un espacio de nombres con características similares a las de Windows, con la diferencia de que en UNIX el espacio no es disjunto, sino que hay un único espacio que une todos los dispositivos activos que contienen un sistema de fichero.

3. Los ficheros de acceso directo de Windows presentan enlaces simbólicos. Estos enlaces hacen referencia a un fichero mediante otro nombre del espacio de nombres. Cuando se borra el fichero mediante un enlace físico, los accesos directos se mantienen, ya que no se pueden localizar fácilmente a partir del fichero borrado. Cuando se accede por el nombre de acceso directo, el sistema se da cuenta de que ya no existe el fichero y avisa de esta circunstancia.

4. El principal inconveniente de este hecho es que se mezcla el concepto de función de traducción con la estructura del fichero. Eso provoca que el espacio de nombres pierda flexibilidad. Con un sistema de estas características no se puede asignar más de un nombre a un fichero, ya que en caso de hacerlo se duplica la información, con el riesgo o el coste de mantenerla coherente. Por ejemplo, en MS-DOS se guardaba el número de caracteres en el directorio. Si tenemos dos nombres para el mismo fichero, tendremos dos copias de este número, y si éste se actualiza en función del nombre por el que se accede al fichero, nos podemos encontrar con informaciones incoherentes.

5. Todas las decisiones dependerán en gran medida del volumen de objetos que se quiere dirigir mediante los espacios de nombres y del tipo de usuario de estos objetos.

a) Ficheros: tal como hemos visto en el módulo, el más adecuado es un espacio jerárquico en forma de grafo, que da la máxima flexibilidad a la hora de organizar los objetos. Los nombres deben estar formados por cadenas de caracteres a fin de que se puedan adaptar a nuestras necesidades, ya que, en general, los usuarios somos las personas y el volumen de ficheros que se quiere dirigir es importante.

b) Dispositivos físicos o lógicos: en este caso puede ser suficiente un espacio con estructura en árbol o un espacio lineal, ya que las necesidades de organización de los objetos no son tan importantes como en el caso anterior porque el volumen de objetos es mucho más pequeño. Como los usuarios somos las personas, igual que en el caso anterior, sería conveniente que los nombres los formaran cadenas de caracteres.

c) Dispositivos virtuales: el espacio más indicado es el espacio lineal con nombres sencillos, como números, a causa del hecho de que los usuarios de este tipo de dispositivos son los procesos; además, los dispositivos virtuales que son visibles desde un proceso no necesitan ser visibles desde los otros. Se trata de un conjunto de objetos reducido y con unas necesidades de organización muy bajas.

d) Procesos: con un espacio lineal con nombres sencillos, como números, hay suficiente, ya que, como en el caso anterior, nos encontramos con un conjunto de objetos dirigidos casi exclusivamente por el SO o por los procesos. El número de objetos puede ser importante, pero no tienen necesidades de organización.

e) Usuarios: el más adecuado es un espacio lineal. Los nombres deberían estar formados por cadenas de caracteres en consideración a los usuarios humanos, ya que en este caso los usuarios son los objetos dominio y los usuarios de estos objetos dominio pueden ser las personas y el sistema. El número de objetos puede ser muy grande, pero la necesidad de organización es pequeña.

Finalmente, se debe tener presente que, en todos los casos en los que se ha propuesto un espacio con nombres formados por cadenas de caracteres, es necesario disponer de un segundo espacio de nombres interno, generalmente lineal y con nombres basados en números, próximos a las necesidades de gestión del SO. Los nombres de este espacio interno son aquéllos a los que hace referencia la función de traducción una vez traducido un nombre del espacio externo.

6.a) Si cada usuario sólo pudiera acceder a sus ficheros con todos los derechos (lectura, escritura y ejecución), la matriz de accesos sería la siguiente:

Dominios	**Objetos**							
	Fichero A	**Fichero B**	**Fichero C**	**Fichero D**	**Fichero E**	**Fichero F**	**Fichero G**	**Fichero H**
Joan	Propietario Lectura Escritura Ejecución	Propietario Lectura Escritura Ejecución						
Maria			Propietario Lectura Escritura Ejecución	Propietario Lectura Escritura Ejecución				
Josep					Propietario Lectura Escritura Ejecución	Propietario Lectura Escritura Ejecución		
Marta							Propietario Lectura Escritura Ejecución	Propietario Lectura Escritura Ejecución

b) Si todos los usuarios pudieran acceder con todos los derechos a todos los ficheros, la matriz de accesos sería la siguiente:

Dominios	**Objetos**							
	Fichero A	**Fichero B**	**Fichero C**	**Fichero D**	**Fichero E**	**Fichero F**	**Fichero G**	**Fichero H**
Joan	Propietario Lectura Escritura Ejecución	Propietario Lectura Escritura Ejecución	Lectura Escritura Ejecución	Lectura Escritura Ejecución	Lectura Escritura Ejecución	Lectura Escritura Ejecución	Lectura Escritura Ejecución	Lectura Escritura Ejecución
Maria	Lectura Escritura Ejecución	Lectura Escritura Ejecución	Propietario Lectura Escritura Ejecución	Propietario Lectura Escritura Ejecución	Lectura Escritura Ejecución	Lectura Escritura Ejecución	Lectura Escritura Ejecución	Lectura Escritura Ejecución
Josep	Lectura Escritura Ejecución	Lectura Escritura Ejecución	Lectura Escritura Ejecución	Lectura Escritura Ejecución	Propietario Lectura Escritura Ejecución	Propietario Lectura Escritura Ejecución	Lectura Escritura Ejecución	Lectura Escritura Ejecución

Dominios	Objetos							
	Fichero A	Fichero B	Fichero C	Fichero D	Fichero E	Fichero F	Fichero G	Fichero H
Marta	Lectura Escritura Ejecución	Lectura Escritura Ejecución	Lectura Escritura Ejecución	Lectura Escritura Ejecución	Lectura Escritura Ejecución	Lectura Escritura Ejecución	Propietario Lectura Escritura Ejecución	Propietario Lectura Escritura Ejecución

7.a) En el caso de que cada usuario sólo pueda acceder a sus ficheros con todos los derechos (lectura, escritura y ejecución), efectivamente hace falta un mecanismo de protección. Sería tan restrictivo como fuera posible: no hay ninguna posibilidad de compartir ficheros entre usuarios. Por lo tanto, el derecho de propietario está implícito en la estructura del mecanismo propuesto, y, por consiguiente, su existencia explícita no es importante.

b) En el caso de que todos los usuarios puedan acceder con todos los derechos a todos los ficheros, no hay ningún mecanismo de protección, ya que todos los usuarios pueden hacer lo que quieren sobre cualquier fichero. Así, como no es necesario ningún mecanismo de protección, tampoco se necesita el derecho de propietario.

8.a) En el caso de LCA, la verificación de derechos se realiza en cada acceso al dispositivo, ya que la información de protección se encuentra físicamente en el dispositivo. Esto quiere decir que, en un SO con un mecanismo de protección basado exclusivamente en LCA, todos los accesos al fichero de una sesión de trabajo deben ser verificados.

b) En el caso de las listas de *capabilities*, la verificación de derechos está asociada al proceso que quiere efectuar los accesos. Esta verificación se lleva a cabo de acuerdo con la existencia o no de la *capability* asociada al objeto. Así pues, en un SO basado exclusivamente en *capabilities*, en una sesión de trabajo sólo se debe verificar el derecho de acceso en la operación de abrir, que es la encargada de verificar la existencia de la *capability*. De todos modos, se puede generar una nueva *capability* y asociarla al canal para que se pueda comprobar rápidamente si se tiene derecho a leer o escribir sobre el canal cuando se intenta hacer la operación en cuestión.

9. En un mecanismo como el de UNIX los dominios que recogen las LCA se han restringido en los tres siguientes: propietario, grupo del propietario y el resto de usuarios. Esta simplificación de todos los dominios posibles permite a UNIX efectuar una gestión de las protecciones más eficiente respecto al espacio y al tiempo de gestión. No obstante, esta simplificación restringe la flexibilidad a la hora de especificar los derechos individuales de cada usuario. Por ejemplo, con UNIX no es posible distinguir entre los usuarios que pertenecen al grupo del propietario y, por lo tanto, no se les puede dotar de derechos distintos. Sucede lo mismo con los usuarios que pertenecen al dominio "el resto de usuarios". De todas maneras, esta restricción se puede suavizar con una buena política de definición de grupos de usuarios, ya que un usuario de UNIX puede estar autorizado, dentro de un conjunto predeterminado, a cambiar de grupo de usuarios.

Glosario

directorio *m* Ficheros donde se almacena la estructura de datos que da lugar a la función de traducción.

directorio de trabajo *m* Directorio donde se encuentran los ficheros con los que un usuario está trabajando en un instante concreto. Cuando un usuario empieza la sesión de trabajo, el directorio de trabajo es el directorio inicial.

directorio inicial *m* Directorio desde donde un usuario puede crear su estructura de directorios y donde colocará los ficheros que él cree.

dominio *m* Agentes activos del sistema que pueden actuar sobre un objeto.

derecho *m* Acciones permitidas por parte de un dominio sobre un objeto.

enlace físico (*hard link*) *m* Enlace que relaciona directamente un nombre del sistema de ficheros con el objeto al que hace referencia. O, lo que es lo mismo, relaciona directamente un nombre del sistema de ficheros con el nombre interno del SO del objeto al que hace referencia.

enlace simbólico (*symbolic link* o *soft link*) *m* Enlace que no relaciona directamente un nombre del sistema de ficheros con un objeto, sino que lo hace indirectamente mediante un enlace a otro nombre del sistema de ficheros desde el que se puede localizar el objeto. Este tipo de enlaces se utilizan fundamentalmente para dar un nombre a un objeto que se encuentra en un dispositivo de almacenamiento distinto de aquél donde está el directorio que contiene el enlace simbólico.

espacio de nombres *m* Espacio configurado por el conjunto de todos los nombres posibles de un objeto.

espacio de nombres jerárquicos *m* Espacio formado por una jerarquía de directorios, con un directorio raíz del que cuelgan otros directorios o ficheros que configuran las hojas del árbol. Los espacios jerárquicos pueden tener forma de árbol o, más generalmente, de grafo dirigido.

espacio de nombres lineales *m* Espacio de nombres, con una sola dimensión, donde todos los nombres están al mismo nivel, en un único directorio. En este tipo de espacio no se pueden hacer clasificaciones entre los distintos objetos.

fichero *m* Dispositivo lógico formado por una agrupación lógica de información almacenada en un dispositivo físico, como un disco, un lápiz USB o la memoria, y que puede ser manipulada como un todo. La información que incluye tiene en común un conjunto de propiedades que la caracterizan.

función de traducción *f* Función unívoca que relaciona un nombre con un objeto.

journaling *m* Técnica utilizada en sistemas de ficheros con el fin de evitar la corrupción del SF. Se basa en guardar información en un lugar determinado, sea interno o externo al SF, sobre todo un conjunto de cambios que se realizarán sobre el SF. Esta información se guarda antes de realizar los cambios. Eso permite que en el caso de que se produzca un fallo se pueda detectar el punto en el que se ha producido y se pueda recuperar la consistencia del SF.

lista de *capabilities* *f* Lista de parejas *(objeto*, *derechos*) asociada a un dominio. Cada una de estas parejas se denomina *capability*. Las listas de *capabilities* son el resultado de partir la matriz de accesos en filas y eliminar, igual que en las listas de control de acceso, todas las celdas vacías.

lista de control de accesos (LCA) *f* Lista de parejas (*derechos*, *dominio*) asociada a un objeto. Esta lista es el resultado de partir la matriz de accesos en columnas y eliminar todas las celdas vacías.

matriz de accesos *f* Matriz formada por tantas filas como dominios (*Di*), y tantas columnas como objetos (*Oj*), donde cada celda *Dij* contiene los derechos de acceso del dominio *Di* sobre el objeto *Oj*.

nombre absoluto *m* Nombre formado por la ruta que va desde la raíz, pasando por los distintos subdirectorios, hasta llegar al fichero. Así pues, el nombre absoluto está compuesto por una secuencia de cadenas de caracteres, o componentes del nombre, que están contenidas en los subdirectorios que configuran la ruta. Los componentes están separados por un carácter delimitador.

nombre relativo *m* Nombres formados por el recorrido que va desde el directorio de trabajo hasta el fichero. Distintos ficheros pueden tener el mismo nombre relativo, siempre que éste se obtenga a partir de directorios de trabajo distintos.

objeto *m* Elemento gestionado por el SO sobre el que se pueden llevar a cabo acciones en función de unos derechos.

cerradura y clave *m* Mecanismo de protección basado en el uso de unas parejas de números llamadas *cerradura* y *clave* que han de coincidir con el fin de dar por válido un derecho. Si estos números no coinciden, el derecho queda invalidado.

sistema de ficheros *m* Sistema encargado de gestionar el conjunto de ficheros contenidos en un mismo dispositivo de almacenamiento.

Bibliografía

Bibliografía básica

Silberschatz, A.; Galvin, P.; Gagne, G. (2008). *Operating Systems Concepts* (8.ª edición). John Wiley & Sons.

Tanembaum, A. (2009). *Modern Operating Systems*. Prentice-Hall.