INSTITUTO POLITÉCNICO NACIONAL
CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

No. 1 Serie: Verde Fecha: Mayo 2009

SISTEMAS DE ARCHIVOS DE SISTEMAS OPERATIVOS DISTRIBUIDOS

Reyna Elia Melara Abarca
Felipe Rolando Menchaca García[2]
Rolando Quintero Téllez[3]

RESUMEN

En este informe se presenta un resumen de los sistemas de archivos que implementan los actuales sistemas operativos distribuidos, su evolución, desde la aparición de este importante recurso de la computación y detalles de sus características y principales procesos de operación.

Un objetivo importante ha sido remarcar como fueron evolucionando estos sistemas hasta lograse la impresionante compatibilidad que se ha logrado en este aspecto y la interoperabilidad, de tal manera que hoy en día es una realidad el trabajo colaborativo sobre un archivo, desde diversos nodos de la red; y también la transferencia de archivos entre prácticamente toda la diversidad de los sistemas operativos de red, actualmente en uso.

1 Estudiante de Doctorado en Ciencias de la Computación del C. I. C.
2,3 Profesor Investigador del Laboratorio de Redes y Telecomunicaciones del C. I. C.

SISTEMA DE ARCHIVOS

ÍNDICE

Administración de Archivos

1. Introducción

La utilización de archivos es inherente a cualquier operación que realice una computadora: las aplicaciones, imágenes, películas, documentos, y toda esta información debe poder almacenarse de alguna manera.

La administración de archivos en un sistema operativo se refiere a la manera en que organiza y mantiene a los archivos; está a cargo del sistema de archivos, que es la parte encargada de realizar las tareas para crear, manipular, almacenar y recuperar datos de una computadora.

A alto nivel el sistema de archivos es la manera en que se organiza, almacena, recupera y en general se administra la información en los dispositivos de almacenamiento; para la mayoría de las computadoras en disco duro es el dispositivo por defecto, pero los archivos existen en muchos otros tipos de dispositivos: memorias flash USB, CD, DVD e incluso en algunos casos podría prevalece el uso de cintas de respaldo.

2. Funciones

El sistema de archivos en un sistema operativo, es el mecanismo de abstracción de los dispositivos físicos de almacenamiento, que permite que sean manejados a un nivel lógico sin necesidad de conocer su arquitectura de hardware particular, de esta manera es posible la administración de archivos y directorios relativa a:

- creación,
- eliminación,
- organización,
- lectura,
- escritura,
- modificación,
- ubicación y
- control de acceso.

Los sistemas de archivos se ocupan principalmente de cuestiones como:

1. Definir la interfaz del sistema de archivos hacia el usuario, es decir, cómo se constituye un archivo, como se nombran y se protegen los archivos y qué operaciones se pueden aplicar a los archivos.
2. Crear algoritmos y estructuras de datos para mapear la estructura del sistema de archivos con los dispositivos físicos, de esta manera, el sistema de archivos se encarga de saber en que lugar del dispositivo de almacenamiento se alojan los bytes que conforman la totalidad de un archivo.

Los archivos pueden estructurarse en diferentes maneras siendo comunes,

- como secuencia de bytes [Figura 1(a)],
- como secuencia de registros [Figura 1 (b)],
- como árboles [Figura 1 (c)].

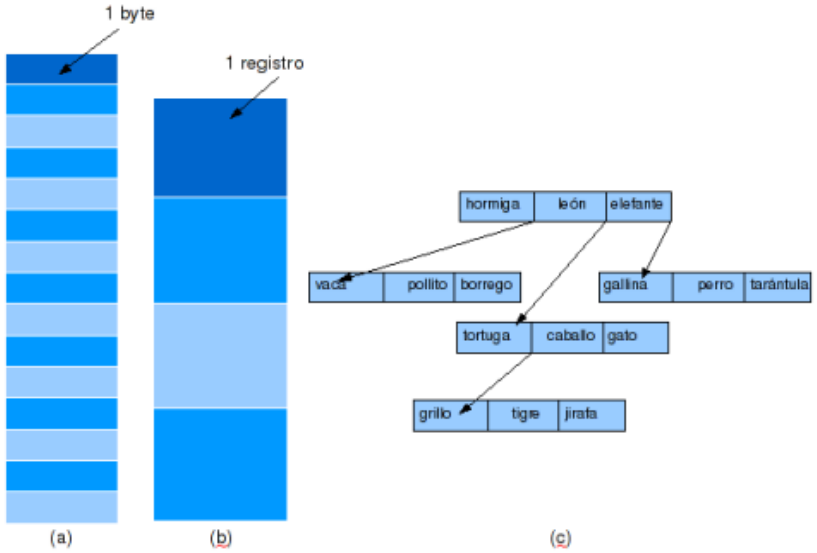

Figura 1: Tres tipos de archivos.

El dato más importante que maneja el sistema de archivos para la recuperación de archivos es el nombre. El sistema de archivos se encarga de crear una estructura jerárquica, conocida como directorios, en la cuál organizar los archivos.

Algunos sistemas operativos dan el mismo tratamiento a directorios y archivos, como en el caso de UNIX y sistemas derivados de este; otros sistemas operativos como los de Microsoft implementan llamadas al sistema separadas para directorios y para archivos, tratándolos de manera diferente.

Además de los archivos y sus nombres, el sistema de archivos almacena los metadatos de los mismos. Algunos de los metadatos que se registran son:

- La ubicación o ubicaciones exactas del archivo en el dispositivo físico (información que es crucial para el sistema de archivos),
- Atributos de acceso y permiso a los archivos,
- Propietario,
- Fechas de creación y modificación,
- Tamaño del archivo, entre otros.

En algunos sistemas operativos modernos es posible que existan metadatos de acuerdo al tipo de archivo, como en el caso de los archivos de audio, que pueden incluir datos del material discográfico y artista; en el caso de las imágenes etiquetas de identificación que facilitan su ordenamiento, por mencionar algunos ejemplos.

Con el tiempo los sistemas operativos han tenido cambios y nuevas características han

sido añadidas a los sistemas de archivos; tal como ha sucedido con la inclusión de nuevos metadatos, se han incorporado opciones de mejora como lo son, el manejo de índices para mejorar las búsquedas, nuevos métodos de almacenamiento, reducción de fragmentación de archivos, capacidades de corrección de errores más robustas y entre estas una adición muy relevante el *journaling,* que es el mecanismo que permite asegurar el correcto funcionamiento de las estructuras de datos en disco, asegurándose de que cada actualización de las estructuras de datos se lleve a cabo completamente o no se realice, aún cuando las actualizaciones sean sobre múltiples bloques de dispositivos de almacenamiento.

3. Evolución Inicial

El surgimiento de los sistemas de archivos se dio como una evolución natural para hacer persistentes los datos en los medios de almacenamiento. En el inicio de la utilización de las primeras computadoras digitales, se podía atender a un usuario por vez, es decir, el poder de cómputo se concentraba en un sólo programa por vez. Los usuarios escribían sus propios programas en lenguaje ensamblador, incluyendo aquellas rutinas de las operaciones de entrada/salida de muy bajo nivel de abstracción, los cuáles estaban ligados estrechamente a la arquitectura de los equipos.

Los programadores eran los encargados de las rutinas que permitían hacer persistentes los datos y recuperarlos; esta tarea debía ser cuidadosamente ejecutada para evitar sobrescribir datos previamente almacenados.

Este tipo de cuestiones dio como resultado que algunos programas de entrada/salida se convirtieron en piezas de código reutilizables y comenzaran a surgir métodos de administración de archivos.

Los primeros sistemas de archivos se conciben de esta manera, es decir, utilizando programas específicos que permitían hacer permanentes los datos en los medios físicos de almacenamiento.

Surgieron los primeros sistemas operativos de tiempo real como la serie de RSX-11 que corrían en las PDP-11 de 16 bits. Estos sistemas operativos multitarea corrían en 32 KB de memoria y tenían un sistema de archivos jerárquico, alternancia de aplicaciones (“applications swapping”), agendado en tiempo real (“real time scheduling”) y una serie de utilidades de desarrollo. Estos sistemas operativos de la serie RSX-11 y sus utilidades fueron parte de la línea de las PDP-11 hasta la PDP-11/70 que contaba con hardware de mapeo de memoria para soportar hasta 4 MB de memoria principal (David Cutler, *Inside Windows NT*).

Esos primeros sistemas operativos manejaban el acceso al contenido del archivo mediante la lectura de los bytes o registros en orden secuencial, como en el caso del sistema de archivos DECTape, de la Digital Equipment Corporation, que se utilizó en la familia de máquinas PDP (Programmed Data Processor), como la PDP-8 en la que se podían almacenar 184 kilobytes por cinta.

Al surgir otros dispositivos de almacenamiento, como el disco, fue posible acceder a los bytes o registros de los archivos en orden diferente al secuencial, incluso acceder por medio de llaves en vez de posiciones, lo que se conoce como acceso aleatorio de archivos.

Además los equipos podían atender a más de un programa, por lo cuál fue imperante controlar la manera de cómo se debían guardar y recuperar los datos.

Con el uso de las primeras microcomputadoras, Gary Kildall desarrolló CP/M (Control Program for Microcomputers) en 1973, que le permitía guardar archivos y ejecutar programas desde un disco Shugart de 8 pulgadas que podía llevar de la computadora de su casa a la de su trabajo. Tenía un sistema de archivos jerárquico plano sin directorios, que permitía guardar archivos cuyos nombres estaban limitados a 8 caracteres mas 3 de la extensión, está extensión determinaba el tipo de archivo. Kildall fundó la empresa Digital Research quién estuvo en negociaciones con IBM para el desarrollo de su sistema operativo para la IBM PC, sin embargo, esto no se concretó.

CP/M consistía de un sólo directorio, así que todo lo que tenía que hacer el sistema de archivos CP/M (Figura 2) para localizar un archivo era buscar en ese único directorio. Cuando la entrada era localizada, se obtiene los números de bloques de disco que ocupa, así como sus atributos. Si el archivo utiliza más bloques que los que caben en una entrada, se alojan entradas adicionales en el directorio.

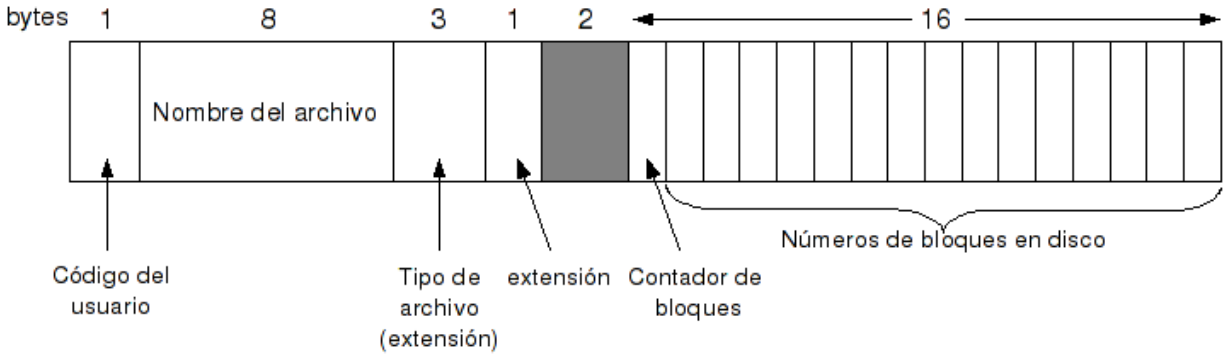

Figura 2: Entrada de directorio del sistema CP/M.

El campo código de usuario en la figura 2 identifica al propietario del archivo. Durante una búsqueda, sólo se revisan las entradas que pertenecen al usuario en sesión. El campo extensión es necesario, ya que si un archivo ocupa más de 16 bloques de datos ocupará más de una entrada en el directorio. El contador de bloques dice cuantos de los 16 bloques están en uso. El último bloque del archivo puede no estar lleno; por lo tanto, el sistema no tiene manera de determinar el tamaño exacto del archivo, ya que contiene el tamaño en bloques, no en bytes.

Por su parte, basado en este sistema operativo, Tim Patterson escribió su propia versión de sistema operativo al que denominó QDOS (Quick and Dirty Operating System), pero para computadoras de 16 bits, ya que no existía una versión de CP/M que cumpliera con esta característica. QDIS tenía un sistema de archivos similar a CP/M y tampoco implementaba directorios, estaba basado en el Microsoft Disk Basic, que era una versión de Basic que permitía grabar archivos en el disco extraíble utilizando el método de organización llamado File Allocation Table, que se conociera como el sistema de archivos FAT.

Este sistema operativo fue comprado por Bill Gates quien lo rebautizó como MS-DOS y finalmente fue el sistema operativo vendido a IBM para que fuera lanzado con sus equipos IBM PC.

MS-DOS tenía una estructura jerárquica y con entradas de directorios de 32 bytes de largo, que contienen el nombre del archivo, los atributos y el número del primer bloque en disco. Mantiene la secuencia de los bloques de datos que usa mediante una lista ligada con un índice, el número de bloque le permite acceder al resto de bloques del archivo (Figura 3).

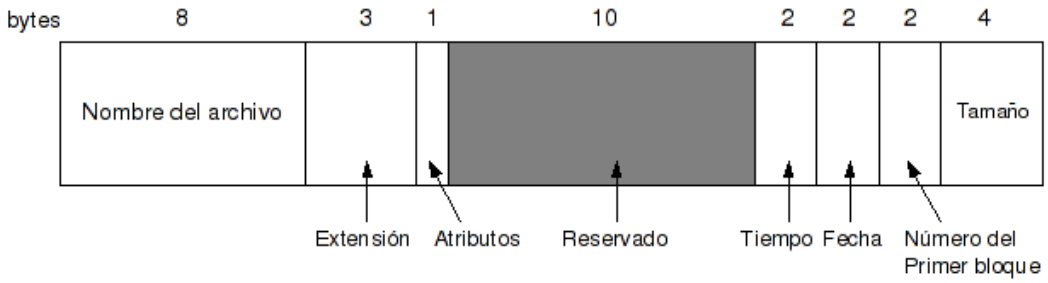

Figura 3: Entrada de directorios de MS-DOS.

Tabla 1: Algunos SO y los sistemas de archivos soportados.

Sistema Operativo	Tipo de sistemas de archivos soportado
Dos	FAT16
Windows 95	FAT16
Windows 95 OSR2	FAT16, FAT32
Windows 98	FAT16, FAT32
Windows NT4	FAT, NTFS (version 4)
Windows 2000/XP	FAT, FAT16, FAT32, NTFS (versions 4 and 5)
Linux	Ext2, Ext3, ReiserFS, Linux Swap (FAT16, FAT32, NTFS), Ext4
MacOS	HFS (Hierarchical File System), MFS (Macintosh File System)
OS/2	HPFS (High Performance File System)
SGI IRIX	XFS
FreeBSD, OpenBSD	UFS (Unix File System)
Sun Solaris	UFS (Unix File System)
IBM AIX	JFS (Journaled File System)

Como se mencionó anteriormente, uno de los primeros sistemas de archivos, conocido como tal fue el RXS-11 de la PDP-11 de la DEC (Digital Equipment Corporation). Después de que apareciera este sistema, surgieron muchos otros, soportados por las distintas empresas. En este trabajo destacamos tres familias de sistemas de archivos que son las

que terminaron por imponerse como estándares de facto en el mercado de la computación. La Tabla 1 relaciona los sistemas de archivos ligados a los sistemas operativos de las familias DOS/Windows, Unix·Linux y MacOS.

4. Sistemas de Archivos de Windows

Al surgir los sistemas operativos de la empresa Microsoft, DOS y Windows, surgió también una familia de sistemas de administración de archivos. A continuación se describen la evolución de esta familia.

4.1 FAT (Files Allocation Table)

Sistema de archivos de MS-DOS, el cuál consiste de una tabla de asignación de archivos (FAT), en la se apunta a las áreas de disco en la que se encuentran los archivos. Cada volumen lógico tiene su FAT.

La tabla de asignación de archivos describe que áreas del disco están ocupadas, las que están libres y aquellas que están dañadas. La primera versión de FAT fue denominada FAT-12 pues utilizaba un número de 12 bits para contar los clústeres (2^{12}=4096, y cada clúster era de 8KB, el tamaño máximo del volumen era de 32MB). Las particiones FAT tienen un tamaño limitado a un máximo de 4 GB bajo Windows NT y 2 GB en MS-DOS.

Todos los nombres de archivo se sujetan a la convención de nombres 8.3, deben de crearse con el juego de caracteres ASCII, los cuáles tendrán una longitud máxima de 8 caracteres sin espacios en blanco, seguidos de un punto (.) y una extensión de 3 caracteres. Las restricciones sobre nombres de archivos y directorios del sistema de archivos FAT es una característica heredada de CP/M.

Un disco formateado con FAT se asigna en clústeres, cuyo tamaño está determinado por el tamaño del volumen. Cuando se crea un archivo, se crea una entrada en el directorio y se establece el primer número de clúster que contiene datos. Esta entrada de la tabla FAT indica que éste es el último clúster del archivo o señala al clúster siguiente.

FAT no tiene una organización determinada en cuanto a la estructura de directorios y se asigna a los archivos la primera ubicación libre de la unidad. Para los discos extraíbles esto no representaba ningún inconveniente, sin embargo, cuando IBM iba a lanzar a la PC-XT con 20MB de disco duro se requirió que la FAT pudiera manejar con mayor eficiencia el espacio; para la versión 2.0 de MS-DOS Microsoft añadió los directorios anidados los cuáles utilizaban como separador el símbolo backslash (\), por ejemplo un archivo podía estar almacenado en:

C:\Directorio\NestedDir

Utilizaron este símbolo (\) que ya venían utilizando como modificador en la versión 1.0 de MS-DOS y la letra C para denotar al disco duro puesto que las máquinas ya tenían los manejadores (“drives”) A y B. Con la introducción del disco duro FAT-12 se volvió obsoleta; para la versión DOS 3.31 Microsoft lanzó una versión de 16 bits, lo que implica que las direcciones de clúster no pueden ser mayores a 16 bits. El número máximo de cúmulos o

clústeres al que se puede hacer referencia con el sistema FAT es, por consiguiente, 2^{16} (65536) cúmulos. Un cúmulo o clúster se compone de un número fijo de sectores de 512 bytes contiguos, el tamaño máximo de la partición FAT se puede determinar multiplicando el número de cúmulos o clústeres por el tamaño de un clúster. Con cúmulos de 32Kb, el tamaño máximo de una partición es de 2GB.

Con el aumento de espacio en disco y los problemas que representaba el uso de la FAT, Microsoft no solucionó el problema de la fragmentación de discos, surgieron entonces herramientas como Norton Utilities y PC-Tools para defragmentar los discos (Figura 4).

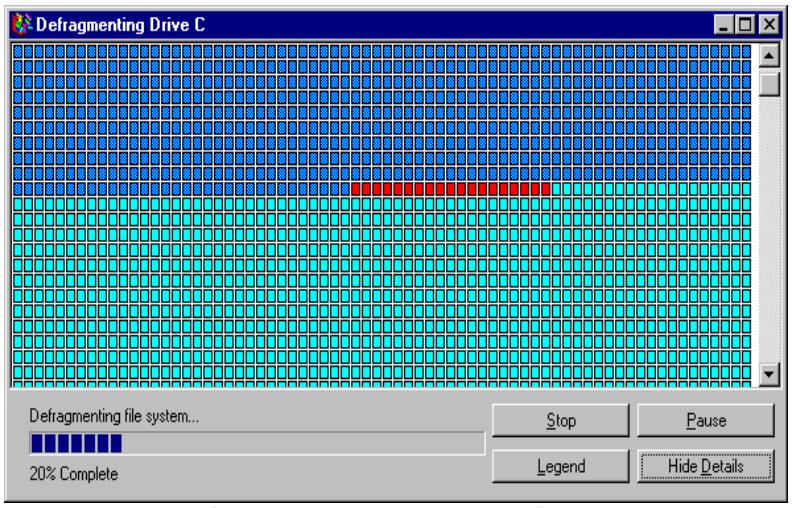

Figura 4: Retícula de defragmentación de disco duro.

Los discos duros fueron creciendo en capacidad, y los nombres del tipo 8.3 se fueron volviendo obsoletos. Microsoft introdujo entonces FAT-32 con límite de 8TB e introdujo VFAT para aumentar la longitud de los nombres de archivos.

Para proteger el volumen, se conservan dos copias de la FAT por si una de ellas resulta dañada. Además, las tablas de FAT y el directorio raíz deben almacenarse en una ubicación fija para que se puedan encontrar correctamente los archivos de inicio del sistema.

4.2 FAT32

Con el lanzamiento de Windows 95 OSR2 se introduce el sistema de archivos FAT32, para solucionar en buena parte las deficiencias que presentaba FAT16; FAT32 admite un tamaño de clúster mínimo predeterminado de 4 KB y es compatible con los discos duros de más de 2GB de tamaño.

FAT32 aprovecha el espacio de forma más eficiente, utiliza clústeres de menor tamaño (de 4 KB a 8 KB), lo que representa de un 10% a un 15% de mejora en el uso del espacio con respecto a unidades grandes con sistemas de archivos FAT o FAT16.

Los formatos FAT16 y FAT32 a pesar de sus inconvenientes, tienen una gran ventaja, y es

que son accesibles (cuando menos para lectura) por una gran cantidad de sistemas operativos, entre los que destacan Unix, Linux, Mac OS. Esta compatibilidad es aun mayor en FAT16 que en FAT32; algunos discos portátiles y pendrives se siguen formateando con FAT16.

Otros sistemas operativos de Microsoft como Windows 98 y Windows Millennium Edition también incorporan la versión del sistema de archivos FAT32.

FAT32 siendo utilizado como puente común para compartir archivos entre diferentes sistemas operativos a través de discos externos y dispositivos USB.

4.3 VFAT

VFAT (Virtual File Allocation Table) es una extensión del sistema de archivos FAT utilizado por primera vez en Windows 95 que funciona como interfaz entre las aplicaciones y la FAT. Es un sistema de 32 bits compatible con FAT pero tiene diferencias como la longitud de nombres de archivos, que puede ser hasta de 255 caracteres incluyendo espacios. Cuando se crea un nombre largo en realidad se crean dos nombres de archivo diferentes, el primero se conoce como nombre largo que es visible a Windows 95, Windows 98 y Windows NT (4.0 y posteriores); el segundo nombre se conoce como alias MS-DOS, que es una forma abreviada de este nombre largo, el cuál se forma con los primeros 6 caracteres del nombre del archivo sin contar espacios en blanco seguida de un tilde ~ y un número, por ejemplo, el nombre de archivo, por ejemplo:

Taxonomía de los sistemas de archivos.txt

Tendría por alias:

Taxono~1.txt

Cuando se crea un nombre largo en VFAT se usa una entrada de directorio para el alias y otra entrada por cada uno de los caracteres del nombre largo.

Microsoft no solucionó del todo los problema de la segmentación con las distintas versiones posteriores de FAT, pero a partir de la versión de Windows 95 incluyó su propia herramienta de defragmentación, que gracias a la función multitarea (“multitasking”), aparentemente permitía seguir ejecutando otras tareas, aunque en realidad esto hacía que la herramienta notificara que los archivos cambiaban mientras se llevaba a acabo la defragmentación teniendo que iniciarla nuevamente.

4.4 HPFS (High Performance File System)

HPFS es un sistema de archivos desarrollado conjuntamente por Microsoft e IBM para el OS/2 1.2. Se desarrolló, como muchos otros sistemas de archivos, para solventar problemas que representaba FAT, aunque HPFS comenzó sin basarse en FAT y con la intención aprovechar los ambientes multitarea.

Algunas cuestiones que implementaba este sistema de archivos son que permitía mejorar el acceso a los discos duros de mayor capacidad que estaban apareciendo en el mercado; organizar los datos en dispositivos de bloques de almacenamiento aleatorio; el espacio en disco se aprovechaba con mayor eficiencia gracias al uso de sectores de 512

bytes en vez de clústeres; aunque mantenía la organización de directorios de FAT, incorporaba la ordenación automática de directorios basada en nombres de archivo, los cuáles podían tener una longitud de hasta 254 caracteres de doble byte; también extendía el sistema de nomenclatura, la organización y la seguridad de las crecientes demandas del mercado de servidores de red.

Bajo HPFS, las entradas del directorio contienen más información que bajo FAT y 64 KB de atributos extendidos o metadatos, también maneja atributos diferentes para fecha de creación, modificación y acceso a los archivos.

El sistema de archivos HPFS se compone de ciertas estructuras fijas, que son el bloque de arranque, el súper bloque y el bloque disponible (Figura 5).

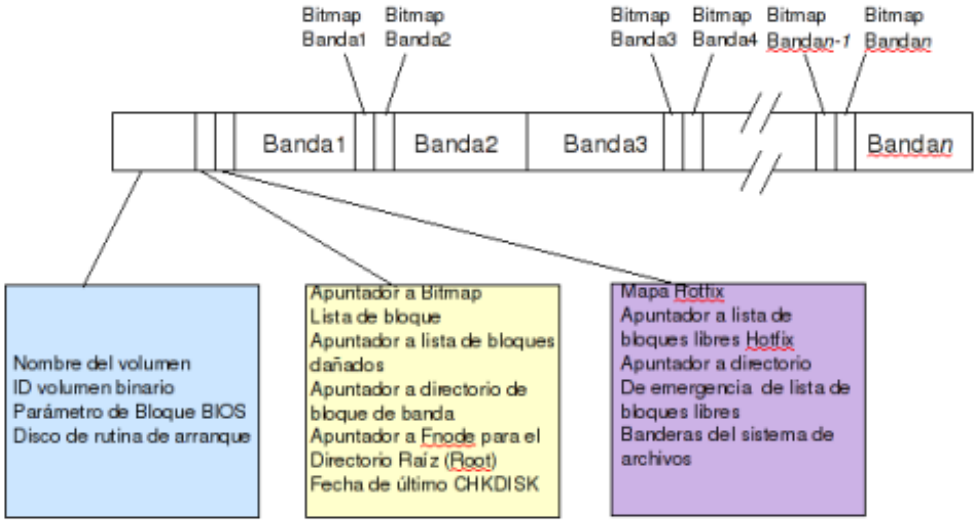

Figura 5: Estructura General de un volumen HPFS.

Los sectores del 0 al 15 de un volumen (8KB) abarcan el bloque de arranque (“Boot block”) y contienen el nombre del volumen, el ID del volumen de 32-bits y un disco de rutina de arranque. Los sectores 16 y 17 se conocen como el Súper Bloque (Super Block) y el Bloque Disponible (Spare Block) respectivamente.

El Súper Bloque solo es modificado por utilidades de mantenimiento del disco y contiene los apuntadores al mapa de espacios libres, la lista de bloques dañados, la banda del bloque del directorio y el directorio raíz, también incluye la fecha de la última vez que el volumen fue revisado y corregido con CHKDSK /F.

El Bloque Disponible contiene banderas y apuntadores del sistema. El resto del disco se divide en bandas de 8Mb, cada banda tiene su propio mapa de espacios libres en el que un bit representa a cada sector (0 si está en uso, 1 si está disponible).

Los archivos y directorios de un volumen HPFS están sujetos a un objeto fundamental del sistema de archivos que se conoce como FNODE. En lugar de señalar al primer clúster del archivo, las entradas del directorio señalan a los FNODE. Cada FNODE ocupa un sector y contiene la información histórica de control y acceso que utiliza el sistema de archivos, atributos y listas de control de acceso, la longitud y los primeros 15 caracteres del nombre del archivo o directorio.

La estructura de directorios de HPFS es implementada por árboles B+.

HPFS intenta asignar en sectores contiguos la mayor cantidad de datos de un archivo que sea posible. De esta forma aumenta la velocidad al hacer un procesamiento secuencial de un archivo.

4.4.1. NTFS (New Technology File System)

Fue creado para el sistema operativo de red y multitarea Windows NT de Microsoft.

Maneja un sistema de archivos jerárquico similar al del sistema operativo UNIX. A diferencia del separador de nombres de archivos que maneja UNIX /, NT utiliza \, maneja también el concepto de directorio de trabajo vigente, y los nombres de ruta pueden ser relativos o absolutos.

A diferencia del sistema de archivos de UNIX que permite montar sistemas de archivos que estén en diferentes máquinas y discos cómo si fueran parte del mismo árbol, lo que oculta la estructura de discos del resto del software, la versión NT 4.0 no contemplaba esta propiedad, por lo que los nombres de archivos absolutos debían comenzar con la letra de la unidad que indicaba de que disco lógico de trataba, por ejemplo,

C:\windows\system\componente.dll

El montaje de discos se añadió en la versión 5.0 de NT al estilo de UNIX.

Cada disco se divide en volúmenes, equivalentes a las particiones del sistema operativo UNIX, en los que hay archivos, directorios, mapas de bits y estructuras de datos para manejar la información. Cada volumen se organiza como una secuencia lineal de clústeres cuyo tamaño fijo puede variar entre los 512 bytes y 64 KB dependiendo del tamaño del volumen. Los cúmulos son referenciados por su distancia al principio del volumen empleando números de 64 bits.

La estructura de datos principal de cada volumen es la MFT (Master File Table), la tabla maestra de archivos (Figura 6), que es en realidad un archivo que puede localizarse en cualquier parte del volumen y en la que se guarda una entrada (análoga a los i-node de UNIX) por cada archivo y directorio del volumen.

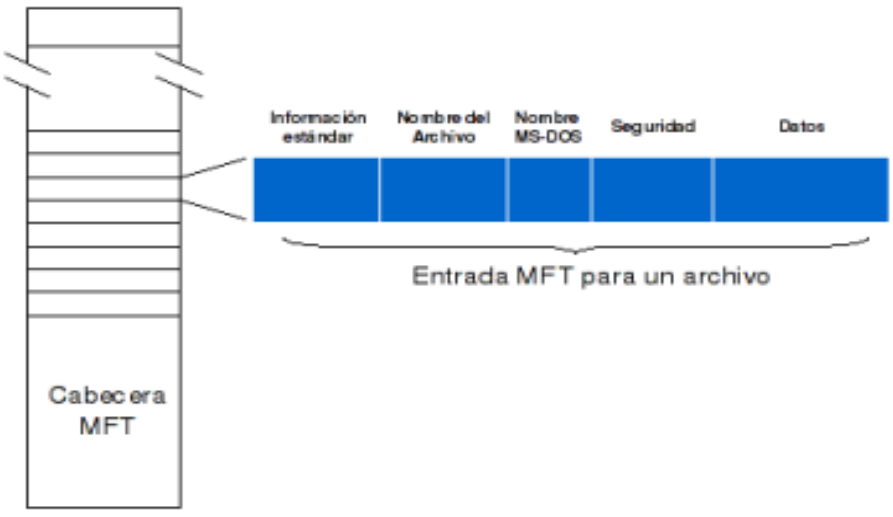

Figura 6: Master File Table de NTFS.

La MFT contiene una cabecera con la información del volumen, la cuál consiste en apuntadores a al directorio raíz, el archivo de arranque, el archivo de bloques defectuosos, la administración de la lista libre. Cómo ya se mencionara, la tabla incluye una entrada por cada archivo o directorio, equivalente a 1KB, excepto cuando el tamaño del cúmulo es de 2KB o más.

Por cada entrada se guardan los metadatos del archivo, entre los cuáles destacan: información estándar, nombre del archivo, nombre MS-DOS, información de seguridad y la sección de datos. En la tabla 2 se muestra la información correspondiente a estos metadatos.

Tabla 2: Metadatos de una entrada

Campo de metadatos	**Datos que contiene**
Campo de información estándar	- las marcas de tiempo que necesita POSIX, - número de vínculos al disco duro, - bits de sólo lectura y almacenamiento, - entre otros.
Nombre del archivo	Los nombres de archivos en NTFS pueden tener hasta 255 caracteres Unicode y son sensibles al tipo de caracter (case sensitive).
Nombre MS-DOS	Nombre alternativo para los programas de 16 bits, con formato 8+3. Si los archivos ya se ajustan a la regla de nombres 8+3 no se utiliza un nombre MS-DOS secundario.
Información de seguridad	NT asigna una ficha de acceso cuando un usuario ingresa al sistema, que contiene el identificador de seguridad, SID (security ID) a cada usuario, una lista de los grupos de seguridad a los que pertenece el usuario, privilegios especiales, entre otros datos. Esta ficha de acceso concentra toda la información de seguridad en un sólo lugar, lo que facilita su localización, cada proceso creado posteriormente, hereda la misma ficha de acceso. Cuando se crea un objeto uno de los parámetros que pueden proporcionarse es su descriptor de seguridad que contiene la lista de control de acceso ACL (Access Control List) en la que se especifican los privilegios de cada SID. En NT 4.0 el campo de seguridad contenía el descriptor de seguridad. A partir de NT 5.0 la información de seguridad se concentró en un archivo, y el campo de seguridad apunta a la parte pertinente de este archivo
Datos	En el caso de archivos pequeños los datos están contenidos en la entrada correspondiente de la MFT, de este modo se ahorra el acceso a disco. En el caso de los archivos un poco más grandes, este campo guarda los apuntadores a los cúmulos que contienen los datos, o a una serie de cúmulos consecutivos, por lo que un sólo número de cúmulo y la longitud pueden representar una cantidad arbitraria de datos del archivo. Si una entrada MFT no basta para contener toda la información se pueden ligar una o más entradas adicionales. Un archivo puede tener como máximo 264-1 bytes.

Las principales funciones de la API Win32 para la gestión de archivos se muestran en la Tabla 3.

Tabla 3: Llamadas al sistema para manejo de archivos (NTFS)

Llamada	**Descripción**
CreateFile	Crear un archivo o abrir un archivo existente; regresa una manija.
DeleteFile	Destruir un archivo existente.
CloseHandle	Cerrar un archivo.
ReadFile	Leer datos de un archivo.
WriteFile	Escribir datos en un archivo.
SetFilePointer	Hacer que el apuntador de archivo apunte a un lugar específico del archivo.
GetFileAttributes	Devolver las propiedades del archivo.
LockFile	Poner candado a una región del archivo para tener exclusión mutua.
UnlockFile	Quitar el candado puesto antes a una región del archivo.

5. Sistemas de Archivos Mac

Desde un principio, Macintosh prestó una gran importancia al sistema de archivos. En general la empresa se orientó mayormente a satisfacer las necesidades del mercado académico y por ello adoptó sistemas más completos y funcionales. A continuación se describen los principales productos generados por esta organización.

5.1 MFS (Macintosh File System)

La máquina de 128K de la Macintosh tenía algunas limitantes comparada con equipos de otras marcas de esa época (alrededor de 1984), la Mac tenía solo una unidad de disco flexible mientras que el uso del disco duro se estaba convirtiendo en algo habitual. El sistema de archivos original de Mac se llamaba Macintosh File System (MFS) de un límite de 20MB y 4,096 archivos.

Los archivos Macintosh tienen dos componentes: *"data fork"*; y el *"resource fork";* El "*data fork"* contiene el flujo de bytes que componen al archivo. El *"resource fork"* es un archivo indexado (indexed file) que contiene segmentos de código, elementos de menú, cuadros de diálogo, datos estructurados que incluyen metadatos del archivo necesarios para el funcionamiento de la interfaz gráfica, como podía ser el icono que tenía asociado el archivo.

Además de los metadatos, los *"resource forks"* también podían incluir detalles de las interfaces de aplicación y código ejecutable.

Los archivos, directorios y la información sobre espacio disponible se almacenaba en un árbol-B.

Los nombres de archivos podían tener una longitud de hasta 255 caracteres, sin embargo, Finder, que es la aplicación responsable del acceso a los usuarios al sistema de archivos y que automáticamente mantiene la asociación entre aplicaciones y archivos de datos, equivalente al shell en otros sistemas operativos pero implementa una interfaz gráfica de usuario, solo permitía a los usuarios crear nombres de 63 caracteres de longitud.

Este sistema operativo no manejaba directorios, al menos no de la manera en que los concebimos normalmente. Los directorios eran visibles en el Finder, pero no en los cuadros de diálogo de salvar o abrir; se trataba de una simulación en tiempo de ejecución que se lograba sólo a través del Finder. MFS almacenaba todo el listado de información correspondiente al archivo y directorio en un único archivo y el Finder aparentaba almacenar los archivos en un manejador de directorios y para desplegar los contenidos del directorio, el sistema de archivos escaneaba el directorio para obtener todos los archivos de ese manejador, sin necesidad de buscar en un archivo diferente el contenido del directorio.

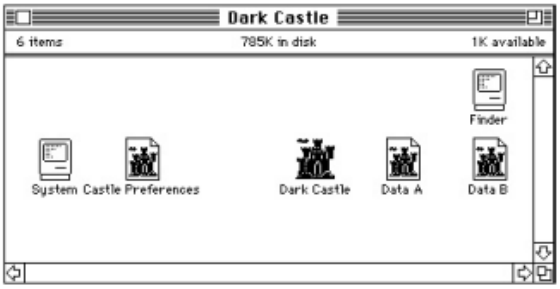

Figura 7: Vista del Finder de MFS.

5.2 HFS (Hierarchical File System)

Este sistema de archivos fue el reemplazo de MFS. HFS tenía soporte para discos MFS.

HFS tenía una estructura de directorios jerárquica, que permitía la creación de subdirectorios anidados.

Al igual que MFS, HFS para manejar los archivos utilizaba los "forks", el *“data fork”*; y el *“resource fork”.*

El Finder de HFS tenía una apariencia similar a MFS, sin embargo, a diferencia de MFS las carpetas si eran subdirectorios reales y no simulaciones. Además el Finder incluía un comando de una vista adicional de “iconos pequeños” (“Small Icon"), que permitía que un icono pequeño estuviera asociado a un tipo de archivo al cuál identificaba (Figura 8).

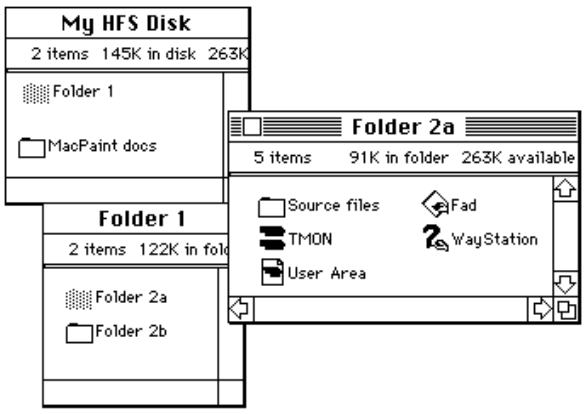

Figura 8: Vista del Finder de HFS.

Un disco puede contener varias particiones, y el sistema trata cada una como discos virtuales separados. Cada partición contiene información similar a la de la Figura 9.

Cada volumen HFS comienza con dos bloques de arranque (boot). Los bloques de arranque en el volumen de inicio se encuentran dos estructuras adicionales leen en el momento de inicio y contienen instrucciones de arranque y otros datos como el nombre del archivo de sistema y el Finder.

Después de los bloques de arranque están dos estructuras adicionales: el bloque maestro de directorio (master directory block) y el mapa de bits del volumen (volume bitmap). El

primero contiene información sobre el volumen como lo es la fecha y hora de la creación del volumen y el número de archivos que el volumen contiene. Cuando se lee el MDB, el volumen queda montado y en memoria se crea un área llamada el Bloque de control del volumen (VCB). El volumen de mapa de bits contiene un registro sobre los bloques que están en uso en el volumen.

El mapa de bits del volumen es un registro de los bloques lógicos asignados a cada archivo. Contiene un bit para cada bloque que es posible asignar en el volumen. Si el bloque está ocupado, el bit está activado, si está disponible el bit esta desactivado.

La porción más grande del volumen consiste de cuatro áreas de información:

- aplicaciones y archivos de datos,
- el catalogo de archivos,
- el desborde de archivo de área contigua (“extents overflow file“),
- espacio no utilizado.

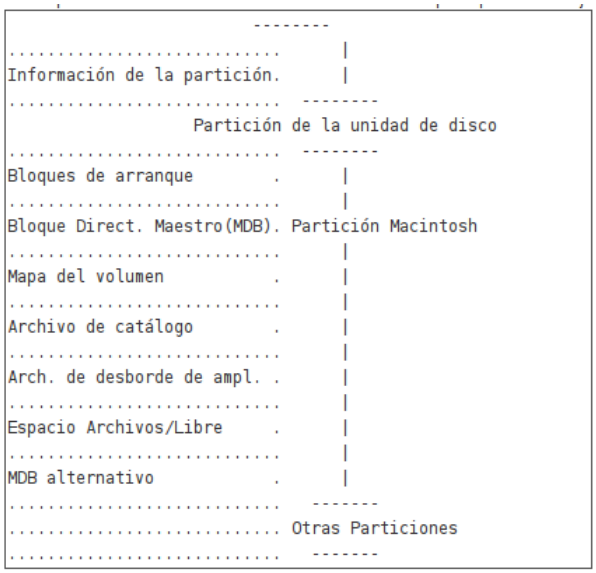

Figura 9: Estructura general de un volumen HFS.

El archivo de catálogo contiene información jerárquica sobre las relaciones y la estructura de los archivos y las carpetas, y su ubicación en el volumen. Específicamente contiene el directorio padre de cada archivo. Para determinar el camino completo al archivo, se busca cada uno de los directorios padres hasta que se alcanza el directorio raíz. El Archivo de catálogo y el Archivo de extensiones tienen ambos la forma de un árbol binario.

Una "extensión" es una zona de bloques lógicos contiguos que están asignados a un archivo. El Archivo de extensiones (también llamado "Archivo de desbordamiento de extensiones") mantiene guardada la ubicación de los registros que no pueden colocarse de forma contigua. Esta información se utiliza para localizar los trozos de archivo cuando hay que cargarlo. Parte de la información de las extensiones está guardada en el MDB y el VCB. Las primeras tres extensiones del archivo se mantienen continuamente en memoria en el VCB.

Esto hace que la optimización o "defragmentación" de un disco sea eficaz. El hecho de tener los archivos en bloques contiguos no sólo reduce el tiempo de búsqueda sino que elimina el acceso continuo al archivo de extensiones pues toda la información necesaria está continuamente disponible en la memoria.

Todos los elementos listados antes de los archivos de catálogo y de extensiones son contiguos. Los archivos de catálogo y de extensiones pueden estar en cualquier zona del volumen y no son contiguos.

El archivo de catálogo también guarda de cada archivo la información para el Finder. Dicha información es la siguiente:

- Tipo de archivo: Identificador que indica una de las diferentes categorías posibles.
- Creador del archivo: El nombre de la aplicación que lo creó.
- Indicadores del Finder: Estos elementos de información son bits que pueden estar activados o desactivados.

El Finder tiene esto en cuenta cuando lee los archivos. Los tipos específicos de indicadores del Finder son:

- isInvisible: El archivo no aparecerá en las ventanas ni diálogos con listas de archivos.
- hasBundle: El archivo tiene asociado un icono.
- nameLocked: El usuario no puede cambiarle el nombre ni asignarle otro icono.
- isStationery: El archivo es una plantilla.
- isShared: El archivo está siendo compartido en la red.
- hasCustomIcon: El archivo tiene su propio icono.

Ubicación del archivo en la ventana: La posición relativa cuando se abre la ventana. Directorio que contiene el archivo: La ruta de acceso al directorio en el que está el archivo.

El resto del volumen contiene archivos de aplicaciones y datos, el Archivo de catálogo, el Archivo de extensiones y espacio libre.

5.2.1. Árboles B en HFS

El Archivo de catálogo y el Archivo de extensiones están organizados en árboles B, una estructura que permite optimizar la velocidad de lectura.

Ambos archivos contienen sólo forks de datos y no de recursos. La ubicación del comienzo del árbol B del catálogo y de las extensiones, está al inicio del MDB y está almacenado en memoria.

La Figura 10 muestra la estructura del disco correspondiente en el Archivo de catálogo y en el Archivo de extensiones:

El archivo de catálogo es una lista de todos los archivos y carpetas almacenadas en un volumen. El archivo de catálogo mantiene las relaciones entre los archivos y directorios en un volumen de directorio jerárquico. Se corresponde con el directorio de archivo en un volumen directorio plana. El archivo de catálogo está organizado y tener acceso mediante una estructura de árbol B.

Los archivos se organizan en orden alfabético equilibrada uniformemente en el árbol para que buscar una Z no tomar cualquier más largo que buscar a una. Esta estructura es el elemento aglutinador que mantiene el archivo de catálogo junto.

El archivo de árbol de las extensiones contiene las ubicaciones de todos los archivos en un volumen. (Una extensión es una serie de bloques contiguos de asignación). El archivo de árbol de extensiones es donde se almacena la información sobre los archivos de datos de que haya creado (como dónde encontrar archivo y extensiones cuántos un archivo está dividido en). Puede dividir cualquier archivo que cree en varias extensiones. Las extensiones están vinculadas, en segundo plano, con información desde el archivo de árbol de extensiones para darle el aspecto de un archivo de datos.

La sección siguiente y mayor, del volumen contiene todos los archivos de datos reales y las aplicaciones que se hace referencia con los archivos anteriores.

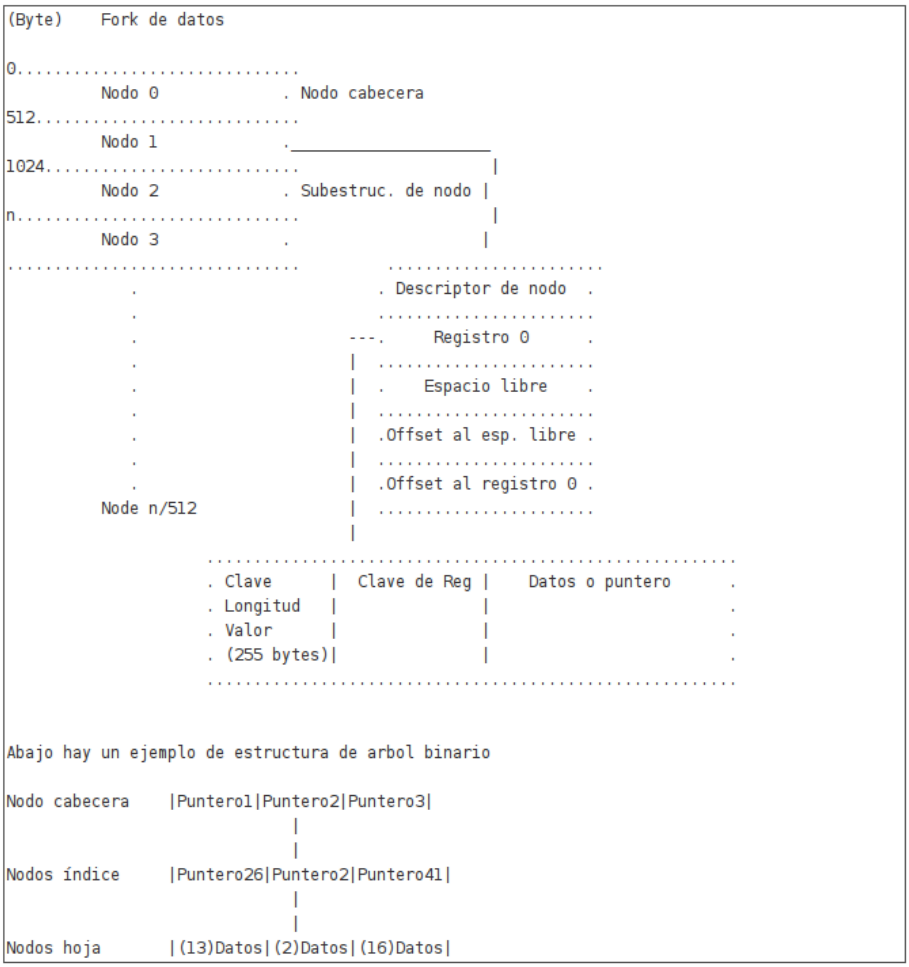

Figura 10: Estructura del disco correspondiente en el Archivo de catálogo y el Archivo de extensiones.

La siguiente al último bloque en el disco duro contiene el directorio principal alternativo. Esta alternativa es una copia de seguridad para la MDB mantiene en bloque lógico 2. Se utiliza cuando el Administrador de archivos determina que la MDB está dañada y debe escribirse con la información correcta.

El último bloque está vacío. Se utiliza para comprobar secciones incorrectas del disco duro.

5.3 HFS+ (Hierarchical File System Plus)

HFS Plus o HFS+ es un formato para volúmenes de MAC OS que fue liberado para la versión 8.1 de MAC OS, su arquitectura es similar a la de HFS pero con algunas diferencias introducidas; algunas se mencionan en la Tabla 4. Esta versión soporta indexado de datos, cuotas, tamaños de archivos mucho más grandes (bloques de 32 bits en vez de los usuales de 16).

La importancia de los cambios que guiaron el diseño de HFS Plus son:

- mejorar el aprovechamiento del espacio en disco,
- formato internacional de nombres de archivos,
- soporte a futuro de los *"forks"* y
- facilitar el arranque en sistemas operativos no MAC.

Un volumen HFS contiene cinco archivos especiales que almacenan las estructuras de archivos necesarias para acceder a la carga del sistema de archivos: carpetas, archivos de usuario y atributos. Los archivos especiales son el catalogo de archivos, el archivo de sobreflujo de extensiones ("extents overflow file"), el archivo de asignación, el archivo de atributos y el archivo de arranque. Los archivos especiales tienen un único fork (el data fork) y el alcance del fork se especifica en el encabezado del volumen.

El archivo de catálogo es un archivo especial que describe la carpeta y la jerarquía de archivo de un volumen; contiene información vital acerca de todos los archivos y carpetas de un volumen, así como la información de catalogo, para los archivos y carpetas que se almacenan en el archivo de catalogo. El archivo de catalogo está organizado por un árbol B (árbol balanceado B-tree) que permite que las búsquedas sean más rápidas y eficientes en una jerarquía grande de carpetas.

El catalogo de archivo almacena el archivo y los nombres de carpeta que contienen hasta 255 caracteres Unicode.

El archivo de atributos contiene datos adicionales de un archivo o carpeta y también está organizado por medio de un árbol B.

HFS Plus rastrea que bloques de asignación pertenecen a un "fork" mediante una lista de los "forks" de área contigua. Un área contigua es un área de bloques contiguos para algunos "fork", representados por un par de números: el número de los primeros bloques de asignación y el número de bloques de asignación. Para un archivo de usuario, las primeras ocho áreas contiguas de cada "fork" se almacenan en el catalogo de archivo del volumen. Cualquier otra área contigua se almacena en el archivo de extensión contigua, que también está organizado por medio de un árbol B.

El archivo de asignación es el que especifica si un bloque de asignación está libre o está siendo ocupada.

El archivo de arranque facilita el arranque desde computadoras que no tienen MAC OS desde volúmenes HFS Plus.

Finalmente el archivo de bloques averiados previene que el volumen utilice ciertos bloques de asignación debido a que existan bloques defectuosos.

Los volúmenes HFS Plus consisten de siete tipos de información o áreas (Figura 11):

- “forks” de archivos de usuario,
- archivo de asignación (bitmap),
- el catálogo de archivo,
- la extensión contigua de un archivo,
- el archivo de atributos,
- el archivo de arranque,
- el espacio no utilizado.

Tabla 4: Comparativa entre HFS y HFS+.

Característica	HFS	HFS Plus	Beneficio/Comentario
Nombre visible al usuario	Mac OS Standard	Mac OS Extended	
Número de bloques de asignación	16 bits	32 bits	Decremento radical en espacio de disco utilizado en volúmenes grandes, y un número mayor de archivos por volumen.
Nombres largos de archivo	31 caracteres	255 caracteres	Beneficio obvio para el usuario; además mejora compatibilidad entre plataformas.
Codificación de nombres de archivos	MacRoman	Unicode	Permite nombres de archivos en formato internacional, incluidos scripts nombres mezclados
Atributos de archivos/ carpetas	Soporte para atributos de tamaño delimitado (FileInfo y ExtendedFileInfo)	Permite futuras extensiones a los metadatos.	Futuros sistemas podrán utilizar metadatos para enriquecer los beneficios del Finder.
Soporte de inicio del SO	ID de carpeta de sistema	Soporta también un archivo dedicado para el inicio	Permitirá a SO no MAC a arrancar desde volúmenes HFS.
Tamaño del catalogo de nodos	512 bytes	4 KB	Mantiene la eficiencia frente a otros cambios. (Este catálogo de nodos más grande se debe a la mayor longitud de los nombres de archivos [512 bytes contra 32 bytes], y a registros de catalogo más grandes (por campos mas grandes)).
Tamaño máximo del archivo	2^{31} bytes	2^{63} bytes	Por supuesto el beneficio en pro de los usuarios, especialmente en archivos multimedia.

5.3.1. Árboles B en HFS+

Las estructuras de árboles B son utilizadas en los archivos de catálogo, de extensión contigua y de atributos. Un árbol B es almacenado en un archivo de fork de datos. Cada árbol B tiene una estructura llamada HFSPlusForkData en el encabezado del volumen que describe el tamaño y la extensión contigua del fork de datos.

Un archivo de árbol B está dividido en nodos de tamaño fijo, cada uno de los cuales contiene registros que consisten de una llave y algunos datos. El propósito de los árboles B es realizar eficientemente la correspondencia entre llaves y datos. Para lograr eso, las llaves deben de estar ordenadas, esto es debe de existir una manera bien definida de decidir si una llave es más pequeña, igual o más grande que alguna otra llave.

El tamaño del nodo (que se expresa en bytes) debe ser una potencia de dos, desde 512 a 32,768. El tamaño del nodo de un árbol B se determina cuando el árbol B es creado. La longitud lógica de un archivo de árbol B es el número de nodos por el tamaño del nodo.

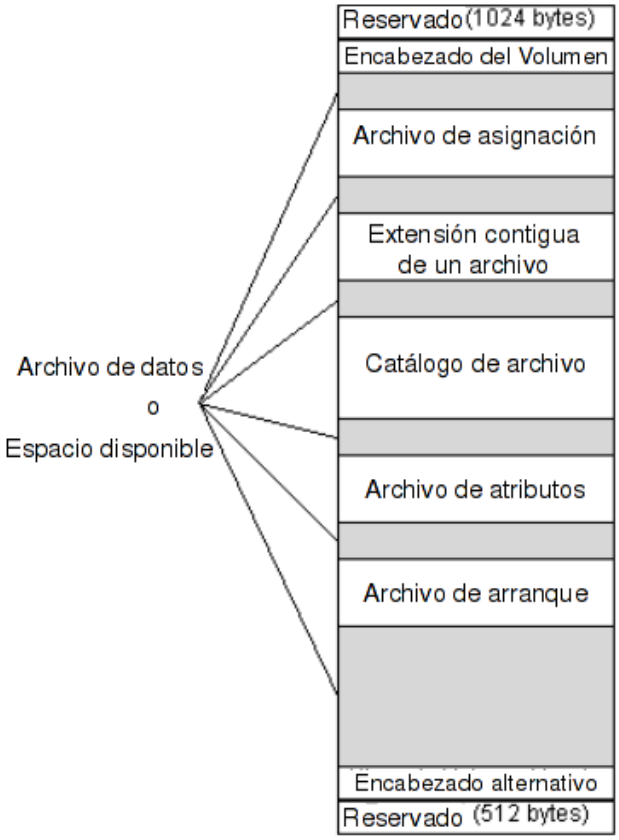

Figura 11: Estructura general de un volumen HFS Plus.

```
struct HFSPlusForkData {
    UInt64                  logicalSize;
    UInt32                  clumpSize;
    UInt32                  totalBlocks;
    HFSPlusExtentRecord     extents;
};
typedef struct HFSPlusForkData HFSPlusForkData;

typedef HFSPlusExtentDescriptor HFSPlusExtentRecord[8];
```

Figura 12: Estructura HFSPlusForkData.

Existen cuatro tipos de nodos:

- Cada árbol B contiene un nodo de encabezado, que es el primer nodo en el árbol. Contiene la información necesaria para encontrar cualquier otro nodo del árbol.
- Nodos mapa, que contiene registros de mapa que guardan datos de asignación (un mapa de bits que describe los nodos libres en el árbol B) que sobrecarga el mapa de registro en el nodo de encabezado.
- Nodos índice que contiene apuntadores a registro que determinan la estructura del árbol B.
- Nodos hoja, que tienen los registros de datos que contienen los datos asociados con una llave. La llave para cada registro de datos debe ser único.

Los nodos tienen la siguiente estructura (Figura 13):

```
struct BTNodeDescriptor {
    UInt32    fLink;
    UInt32    bLink;
    SInt8     kind;
    UInt8     height;
    UInt16    numRecords;
    UInt16    reserved;
};
typedef struct BTNodeDescriptor BTNodeDescriptor;
```

Figura 13: Descriptor del nodo.

Los campos tienen el siguiente significado:

flink	El número de nodo del siguiente nodo de ese tipo o 0 si es el último nodo.
blink	El número de nodo del nodo previo de ese tipo o 0 si es el primer nodo.
kind	El tipo de un nodo de acuerdo a los cuatro tipos mencionados.
height	el nivel o profundidad del nodo en la jerarquía del árbol B. Para el nodo de encabezado, este campo debe ser 0. Para los nodos hoja, el valor debe ser 1, para los nodos índice, este campo es un valor más grande que el nivel de los nodos hijos a los que apunta. El nivel del nodo mapa es 0, como el nodo de encabezado (Los nodos mapa son extensiones de los mapas de registro en el nodo de encabezado).
NumRecords,	El número de registros contenidos en ese nodo.
Reserved	Un campo reservado.

Los árboles B son una estructura de datos que facilita las búsquedas, inserción y eliminación eficiente, la cuál tiene efecto sobre los registros de llave (apuntadores a registro y registros de datos) y nodos en los que están almacenados (nodos índice y nodo hoja). A continuación de listan requerimientos de ordenamiento para nodos índice y hoja:

- registros de llave deben de colocarse en los nodos de manera tal que las llaves deben de estar en orden ascendente.
- Todos los nodos en un nivel dado (cuyo campo **height** es el mismo) debe de estar ligado por medio de los campos **fLink** y **bLink**. El nodo con la llave más pequeña debe ser el primero en la cadena y el campo **bLink** debe ser 0. El nodo con la llave más grande debe ser el último en la cadena y el campo **fLink** debe ser 0.

- Para un nodo dado, todas las llaves en el nodo debe deben ser menores que todas las llaves en el nodo siguiente dentro de la cadena (que es apuntado por **fLink**). Similarmente, todas las llaves en el nodo debe ser mayor que todas las llaves en el nodo previo de la cadena (que es apuntado por b**Link**).

Tener las llaves ordenadas de este modo posibilita búsquedas rápidas en el árbol B para encontrar datos asociados a una determinada llave (Figura 14).

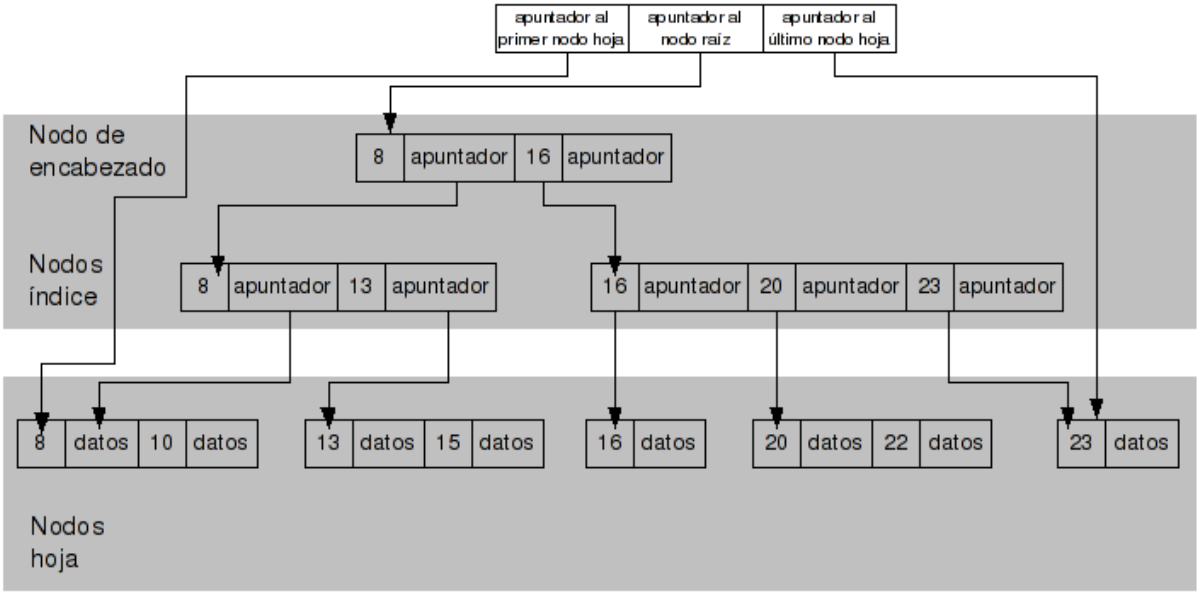

Figura 14: Árbol B.

Cuando se requiere encontrar datos asociados a una llave particular de búsqueda, se comienza a buscar en el nodo raíz (“root node”). Empezando con el primer registro, se busca el registro con la llave más grande que es menor o igual a la llave de búsqueda. Se mueve al nodo hijo (típicamente un nodo índice) y repite el mismo proceso. El cuál continúa hasta que se alcanza un nodo hoja. Si la llave encontrada en un nodo hoja es igual a la llave de búsqueda, el registro encontrado contiene los datos asociados con la llave de búsqueda. Si la llave encontrada no es igual a la llave de búsqueda, la llave de búsqueda no está en el árbol B.

Los reproductores de vídeo y música digital, iPod, utilizan HFS+; HFS+ a su vez, dio paso al nuevo sistema operativo de Apple, el Mac OS X.

El sistema de archivos HFS+ es un sistema de archivos más robusto y como la mayoría de los sistemas indexados, únicamente indexa los metadatos de la información. Aunque este sistema de archivos no es el más avanzado en el mercado, sí ha sido de gran uso y utilidad para la compañía Apple ya que sus ventajas y funciones le han provisto de la capacidad de desarrollar un sistema operativo que se considera robusto, eficiente y seguro que le han permitido competir directamente con Microsoft.

MAC OS X tiene en su núcleo un conjunto de directorios heredado del sistema operativo BSD (Berkeley Software Distribution). Aunque muchos de los directorios de MAC OS X

están ocultos por el “Finder” muchos de los de BSD siguen presentes. El modelo de permisos, los enlaces simbólicos y los directorios del “user home” son conceptos derivados de BSD.

Mac OS X utiliza un sistema multi-usuario, el cual controla el acceso a los recursos del sistema ya que es de suma importancia mantener la estabilidad. Mac OS X define varios ámbitos del sistema de archivos, cada uno de ellos dispone de recursos de almacenamiento en un conjunto establecido de directorios. El acceso a los recursos en cada dominio está determinado por los permisos del usuario actual.

Mac OS X cuenta con cuatro ámbitos del sistema de archivos:

- **Usuario**. El usuario de dominio contiene recursos específicos para el usuario que se registra en el sistema. Este dominio está definido por el directorio home del usuario, que puede ser en el volumen de inicio /users) o en una red de volumen. El usuario tiene control completo de lo que sucede en este ámbito.
- **Local**. El dominio local contiene recursos, como las aplicaciones y documentos que son compartidos entre todos los usuarios de un determinado sistema, pero no son necesarios para ejecutar el sistema. El dominio local no corresponde a un único directorio físico, sino que consta de varios directorios en el arranque (y la raíz) del volumen. Los usuarios con privilegios de administrador del sistema puede añadir, eliminar y modificar los artículos de este dominio.
- **Red**. El dominio de red contiene recursos tales como las aplicaciones y documentos que son compartidos entre todos los usuarios de una red de área local. Los artículos de este dominio se encuentra normalmente en la red y servidores de archivos se encuentran bajo el control de un administrador de red.
- **Sistema**. El sistema de dominio contiene el software del sistema instalado por Apple. Los recursos en el sistema de dominio son requeridos por el sistema para ejecutarse. Artículos en este ámbito se encuentran en el arranque (y la raíz) del volumen. Los usuarios no pueden agregar, eliminar o modificar los artículos de este dominio.

El dominio de un recurso determinado determina su aplicabilidad o la accesibilidad a los usuarios del sistema. Por ejemplo, una fuente instalada en el directorio home del usuario sólo está disponible para ese usuario. Si un administrador instala la misma fuente en el dominio de red, todos los usuarios de la red tienen acceso a ella.

Dentro de cada dominio, Mac OS X proporciona un conjunto de guías para la organización inicial de la figura recursos. Mac OS X utiliza nombres idénticos de directorios a través de dominios para almacenar los mismos tipos de recursos. Esta coherencia simplifica el proceso de búsqueda de recursos tanto para el usuario y para el sistema de los métodos que utilizan esos recursos. Cuando el sistema debe encontrar un recurso, busca en los dominios de forma secuencial hasta que encuentra el recurso. Las búsquedas las comienza en el inicio del apartado de usuario hasta proceder a través de los locales, red y sistema de dominios que preceden en ese orden.

Como se ha mencionado el “Finder” es la herramienta o aplicación con la cual la mayoría de los usuarios interactúan con los archivos en Mac OS X.

Mac OS X es compatible con varios tipos diferentes de sistemas de archivos de forma nativa utilizando el “Finder”. Los más comunes son los formatos HFS + (Mac OS ampliado), HFS (Mac OS estándar) y UFS. Otros formatos de archivos incluyen el

Formato de disco universal (UDF) para discos de DVD y el formato que utiliza la norma ISO 9660 para el CD-ROM de volúmenes. Cada uno de estos sistemas de archivos de almacenamiento de archivos se aplica en forma algo diferente, pero el “Finder” enmascara estas diferencias para proporcionar una experiencia de usuario sin problemas.

Operaciones como el copiar y mover en las que el origen y el destino utilizan el mismo formato de volumen, ocurren muy a menudo como se podría esperar. Las cosas se ponen interesantes cuando los archivos son copiados o movidos a través de volúmenes que apoyan diferentes formatos de volumen.

¿Qué sucede cuando un usuario copia un archivo desde un volumen HFS+ a un volumen UFS? El archivo en el volumen HFS+ incluye información adicional, como el tipo de archivo y el creador. Cuando se produce una operación de ese tipo, el “Finder” no divide ninguna información que no se encuentra en la posesión de los datos del archivo y lo escribe en un archivo oculto en el volumen de destino. El nombre de este archivo es el mismo que el archivo original, excepto que tiene un "punto de relieve" como prefijo. Por ejemplo, si copia un archivo llamado HFS+ MyMug.jpg a un volumen UFS, habrá un archivo ._MyMug.jpg Además de la MyMug.jpg archivo en el mismo lugar.

Al copiar un archivo de un volumen a un UFS HFS o HFS+, el “Finder” busca una correspondencia "punto-de relieve" correspondiente del archivo. Si existe, el “Finder” crea el archivo utilizando un formato HFS+ (o HFS), usando la información contenida en el punto-de relieve para recrear el archivo de recursos. Si el archivo oculto no existe, estos atributos no son recreados.

Además, las interfaces históricamente asociados a cada sistema de archivos a veces tienen diferentes comportamientos. Por ejemplo, utilizando una interfaz de algún BSD se puede borrar un archivo que está abierto, por el otro lado, un programa de carbono (en este caso del “Finder”) sólo puede borrar un archivo que está cerrado.

Los sistemas de archivos HFS y HFS+ incluyen la entidad conocida como un alias. Un alias tiene algunas similitudes con un enlace simbólico en un sistema de archivos UFS, pero las diferencias son significativas.

El “Finder” reconoce los vínculos simbólicos, pero sólo crea alias (cuando se utiliza el comando adecuado). Incluso cuando se encuentra un enlace simbólico en el sistema de archivos, que lo presenta como un alias, es decir, no existe diferenciación visual entre los dos. La única manera de hacer un enlace simbólico en el Mac OS X es usar el comando BSD ln -s desde una ventana de terminal o de script de shell.

En un nivel fundamental, Mac OS X es un sistema BSD, es decir, se aplica la propiedad de BSD, y los permisos para los archivos y carpetas en el sistema de archivos. Este modelo, a su vez, controla que los usuarios pueden leer, escribir, cambiar el nombre, y ejecutar archivos además de que los usuarios pueden copiar y mover archivos desde y hacia las carpetas. Aunque el modelo de la propiedad de los archivos es convencionalmente asociados con UFS o similares sistemas de archivos, el Mac OS X se extiende a todos los sistemas de archivos soportados, incluyendo Mac OS Estándar (HFS) y Mac OS Extended (HFS +).

Para cada carpeta y archivo en el sistema de archivos, BSD tiene tres categorías de usuarios: propietario, grupo y otros. Para cada uno de estos tipos de usuarios, tres afectan a los permisos de acceso al archivo o carpeta: leer, escribir y ejecutar. Si un

usuario no tiene permisos de lectura en cualquiera de las categorías de un archivo, el usuario no puede leer el archivo. Del mismo modo, si un usuario no tiene permisos de ejecución para una aplicación, el usuario no puede ejecutar la aplicación.

En Mac OS X, las cuentas administrativas y cuentas root se les otorgan acceso ampliado a modificar el sistema de archivos.

Desde Mac OS X v10.4 se ha introducido soporte para listas de control de acceso (ACL). Las ACL BSD complementan el actual modelo de permisos en los casos en que se necesita más control sobre quién tiene acceso a un archivo o directorio y las acciones que puede realizar.

Ahora bien comparando el sistema de archivos, podemos encontrar que existen muchas diferencias significativas entre los dos principales sistemas de archivos en Mac OS X: HFS+ y UFS. En muchos casos, estas diferencias tienen alguna incidencia en los programas desarrollados para Mac OS X. En la lista siguiente se resumen las principales diferencias entre estos sistemas de archivos (muchos de estos estados se aplican a HFS y HFS +):

Mayúsculas y minúsculas. UFS es sensible al caso, aunque HFS+ es sensible a las mayúsculas.

Múltiples forks. HFS+ es compatible con múltiples forks (y metadatos adicionales), mientras que UFS apoya un solo fork. (Simula múltiples fork de carbono en sistemas de archivos que no los soportan, como es el caso de UFS).

Separadores de rutas. HFS+ utiliza como ruta de colones separadores UFS que sigue la convención de barras inclinadas. El sistema se traduce entre estos separadores.
Modificación fechas. HFS + es compatible tanto con la creación y modificación de fechas de archivo de metadatos; UFS soporta fechas de modificación, pero no la creación de fechas. Si copia un archivo con un comando que comprende fechas de modificación, pero no la creación de fechas, el comando podría restablecer la fecha de modificación, ya que crea un nuevo archivo de la copia. Debido a este comportamiento, es posible tener un archivo con la creación de una fecha posterior a su fecha de modificación.

Sparse files y llenado cero soporta archivos UFS escasos, que son una forma en la cual el sistema de archivos almacena los datos en el almacenamiento de archivos, sin utilizar espacio asignado para los archivos. HFS+ no es compatible con archivos de poco y, de hecho, llena todos los octetos con cero para un archivo hasta llegar al final del archivo.

6. Sistemas UNIX

"...the number of UNIX installations has grown to 10, with more expected..."
Dennis Ritchie & Ken Thompson

"... When BTL withdrew from the project, they needed to rewrite an operating system (OS) in order to play space war on another smaller machine (a DEC PDP-7 [Programmed Data Processor] with 4K memory for user programs). The result was a system which a punning colleague called UNICS (UNiplexed Information and Computing Service)—an 'emasculated Multics'; no one recalls whose idea the change to UNIX was"

Unix es un sistema operativo desarrollado por los Laboratorios Bell de AT&T a principios de los 70 por Ritchie y Thompson. En sus inicios el código de este sistema operativo era libre y propició que surgieran diferentes versiones. UNIX fue rápidamente abriéndose paso en universidades, institutos de investigación, cuerpos de gobierno y compañías de cómputo para desarrollar tecnologías que forman parte de los sistemas UNIX actuales como son:

- **AIX**, Unix comercial basado en el System V desarrollado por IBM en febrero de 1990.
- **Sun Solaris**, Unix comercial basado en el System V y en BSD desarrollado por SUN Microsystems.
- **HP-UX**, Unix comercial basado en BSD desarrollado por Hewlett Packard a partir de 1986.
- **Ultrix**, Unix comercial desarrollado por DEC.
- **IRIX**, Unix comercial desarrollado por SGI.
- **Unixware**, Unix comercial desarrollado por Novell.
- **Unix SCO**, Unix comercial basado en el System V desarrollado por Santa Cruz Operations y Hewlett Packard a partir de 1979.
- **Tru64 UNIX**, Unix comercial desarrollado por Compaq.

Para fines del año 1977, investigadores de la Universidad de California desarrollaron otra versión Unix a partir del código fuente provisto por AT&T para poder ejecutar el sistema en su plataforma VAX y lo denominaron *BSD*, que significa *Berkeley Software Development (Desarrollo del Software Berkeley)*.

De esta forma se conformaron dos ramas de desarrollo para el código fuente (Figura 15):

- La rama de AT&T que se convertiría en **System V** de los Laboratorios del Sistema UNIX (USL). Se trata de un sistema multiusuario y multitarea que permite que se ejecuten simultáneamente diferentes programas a cargo de uno o varios usuarios.
- La rama de **BSD** (Berkeley Software Development [Desarrollo del Software Berkeley]), desarrollado por la Universidad de California.

Sin embargo a principios de los 80's AT&T tuvo el derecho de comercializar su Unix, lo que marcó la aparición del UNIX System V, la versión comercial de su sistema Unix.

Fueron liberadas cuatro versiones (de la 1 a la 4), siendo la versión 4 la más importante (SVR4), que fue un desarrollo conjunto entre SUN Microsystems y AT&T. SVR4 es la compilación de las mejores funcionalidades de AT&T, UNIX System V ver. 3, Berkeley Software Distribution 4.3, Sun OS, Microsoft XENIX. El SVR4 implementa las nociones de interfaces de aplicación de programa consistentes (Application Programming Interface) y una única interfaz de aplicaciones binaria (ABI) para cada plataforma de hardware.

El lanzamiento 4 (SVR4) del sistema V de UNIX proporciona la base del código para la mayoría de los sistemas operativos comerciales de Unix hoy. El documento estándar que describe el sistema V es la definición de interfaz del sistema V (SVID).

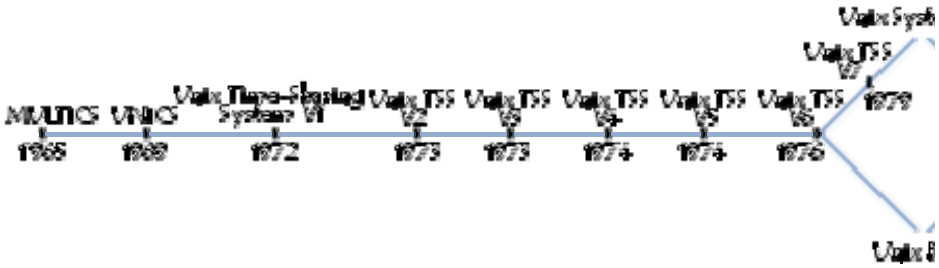

Figura 15: Evolución general de las versiones de UNIX.

Este desarrollo bifurcado de Unix causó diferencias importantes entre las llamadas del sistema, las bibliotecas de sistema, y los comandos básicos de Unix. Un ejemplo es la diferencia entre las interfaces del establecimiento de una red que cada sistema operativo proporcionaba. Los sistemas del BSD utilizaron un interfaz conocido como **zócalos** para permitir que los programas interactúen con otros sobre una red. Por el contrario, el System V proporcionó una **interfaz de la capa de transporte** (TLI), incompatible con los zócalos, la cuál se definió en la interfaz de transporte de X/Open (XTI). Este desarrollo divergente disminuyó la portabilidad de programas a través de las versiones de Unix, aumentando costos y disminuyo la disponibilidad de productos compatibles entre versiones de Unix.

Un intento de estandardizar todos los aspectos de Unix fue la propuesta de definir las interfaces que Unix proporciona. Este esfuerzo por propiciar la interoperabilidad entre los diversos tipos de UNIX se denominó **POSIX** que es el acrónimo de **P**ortable **O**perating **S**ystem **I**nterface, la que **X** hace alusión al sistema operativo UNIX, este término fue sugerido por Richard Stallman. POSIX fue publicado por el Instituto de Ingenieros en Electricidad y Electrónica (IEEE).

La estandarización de la línea de comandos y de la interfaz de scripts se hizo en base del Korn shell. Muchos programas a nivel de usuario, servicios y utilidades, incluidas awk, echo, ed también se estandarizaron así como los servicios a nivel de programa, incluidas los servicios básicas de Entrada/Salida (archivos, terminal y red).

A diferencia del UNIX actual, anteriormente el sistema operativo UNIX era un producto compuesto de cuatro elementos (Figura 16):

1. La especificación (ie. SVID),
2. La tecnología (ie. SVR4),
3. La marca registrada (UNIX),
4. Un producto o versión específica (ie.. UNIX Ware).

Otro intento por unificar es la **Single UNIX Specification**, que es una colección de documentos parte de X/Open Common Applications Environment (CAE) que es administrada por la industria X/Open Company, lo cuál implica que UNIX no es un sistema operativo producto de AT&T, documentado por la Definición de Interfaz del System V (System V Interface Definition – SVID) y tampoco es una colección de productos diferentes pertenecientes a diferentes desarrolladores. La especificación actual de UNIX está concensuada y estable para desarrollo de aplicaciones portables que pueden convivir en sistemas que se conforman en base a la Especificación Única de UNIX. Existe también la marca registrada para identificar a los productos de UNIX que conforman a esta única especificación.

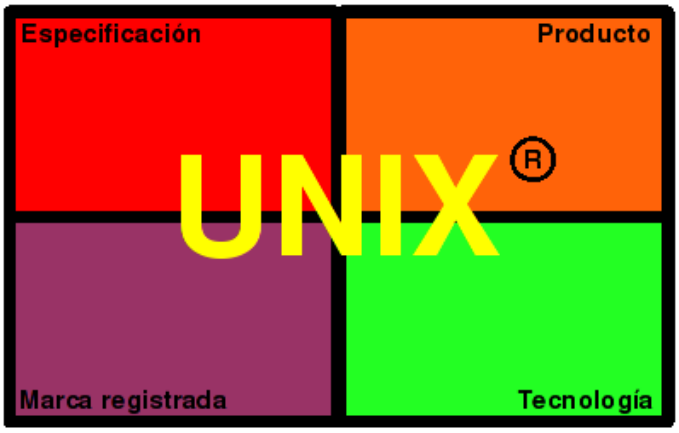

Figura 16: Las cuatro partes de la concepción de UNIX.

6.1 El sistema de archivos de UNIX

En los sistemas UNIX cualquier elemento se representa en forma de archivos. Todos los archivos están ordenados en una única estructura jerárquica, de tipo arborescente y recursiva, en la que los nodos pueden ser tanto archivos como directorios, y estos últimos pueden contener a su vez directorios o subdirectorios. El sistema de archivos de UNIX se caracteriza por:

- Poseer una arquitectura jerárquica.
- Realizar un tratamiento consistente de los datos de los archivos.
- Poder crear y eliminar archivos.
- Permitir un crecimiento dinámico de los archivos.
- Proteger los datos de los archivos.
- Tratar a los dispositivos periféricos como si fuesen archivos.

Las funciones del sistema de archivos están orientadas al manejo de archivos, pero también proporciona los siguientes servicios:

1). **Protección**: permitiendo a los usuarios controlar el acceso a sus archivos.
2). **Organización**: los datos de un archivo pueden ser manipulados independientemente de los datos de otros archivos.
3). **Acceso múltiple:** los sistemas modernos permiten que sea posible acceder a varios archivos de manera concurrente y que sistemas diferentes accedan consistentemente al mismo archivo.
4). **Manejo del espacio**: el sistema mantiene el control del espacio disponible y del utilizado en el disco. Cuando los archivos requieren crecer el sistema de archivos es el encargado de encontrar un espacio libre apropiado para alojar a esos archivos.

6.2 Implementación física de un sistema de archivos

Se espera que un sistema de archivos no sólo encuentre un espacio para alojarlos, sino que ayude a optimizar el ancho de banda del disco; y así, trate de encontrar espacios contiguos o al menos rotacionalmente cercanos para minimizar el tiempo que toma leer y

escribir el archivo.

Existen varias técnicas para mantener el registro de los bloques que conforman físicamente un archivo, de ellas podemos mencionar lo siguiente:

1). **Alojamiento contiguo**: almacena un archivo en bloques de datos contiguos en el disco (Figura 17). Esta cuenta con dos ventajas: posee la implementación más simple dado que mantiene el registro de los bloques usados por un archivo en un contador y sólo almacena la dirección en disco del primer bloque. Y en segundo lugar su desempeño es excelente porque el archivo entero puede ser leído del disco en un solo acceso. De igual forma tiene dos desventajas: su uso no es factible a menos que se conozca el tamaño de archivo desde el momento de creación y en segundo lugar, fomenta la fragmentación de disco.

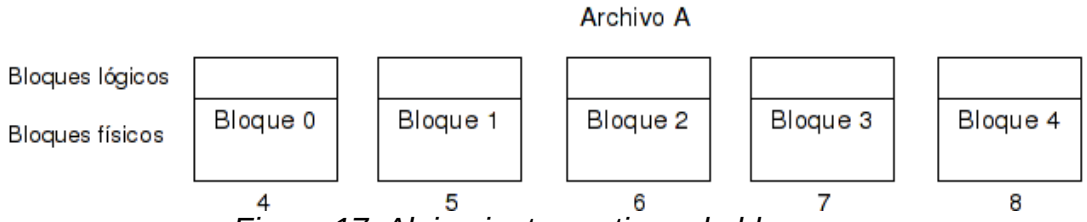

Figura 17: Alojamiento contiguo de bloques.

2). Almacenamiento de bloques da datos en forma de lista ligada: Aquí no es necesario que los bloques sean contiguos físicamente. La primer palabra (2 bytes) de cada bloque se usa como un apuntador al siguiente bloque del archivo y el resto de su espacio lo ocupan bytes de datos (Figura 18). Aquí se evita el desperdicio de espacio debido a la fragmentación (excepto por fragmentación interna del último bloque). Para la ubicación de los bloques es suficiente con almacenar la dirección de disco del bloque que inicia la lista. Por otro lado, a pesar que es sencillo leer el archivo secuencialmente, el acceso aleatorio es extremadamente lento.

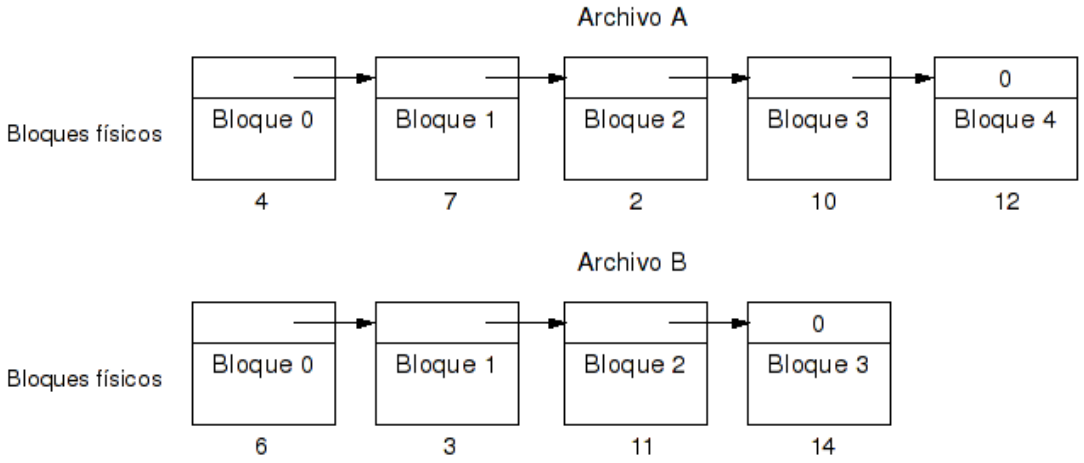

Figura 18: Lista ligada de bloques de archivos.

3). **Listas ligadas con un índice**: Las desventajas de la lista ligada se pueden eliminar indexando los bloques de datos en una tabla en memoria principal (Figura 19); el recorrido por los bloques del archivo A inicia en el bloque 4, donde encuentra la dirección del bloque 7, aquí lee la del 2, después la del 10, hasta que finalmente el 0 en el bloque 12 indica que es el último de ese archivo. Al usar esta organización, el bloque almacena exclusivamente el contenido del archivo; además, que el acceso aleatorio es mucho más fácil. A pesar de que debe recorrerse la lista para encontrar un desplazamiento dentro de un archivo, la tabla se encuentra en memoria, así se evitan lecturas al disco. Para el directorio es suficiente con mantener el número del bloque de inicio y es capaz de localizar los demás bloques, no importa que tan grande es el archivo. MS-DOS utilizaba este método. Su desventaja primaria es que la tabla debe estar siempre por completo en memoria para que funcione.

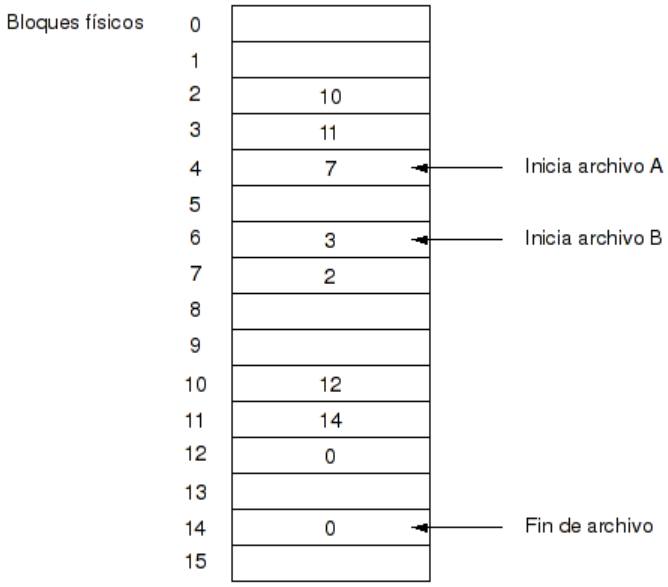

Figura 19: Listas ligadas con un índice.

4). **I-nodes**. A cada archivo se le asocia una pequeña tabla llamada inode (nodo índice) la cual tiene la lista de atributos y direcciones en disco de los bloques de datos del archivo (Figura 20). Las primeras direcciones de los bloques de datos del archivo se almacenan en el inode, así el acceso de archivos pequeños se realiza directamente. Estas direcciones se cargan en memoria principal cuando el archivo se abre. Una de las direcciones en el inode corresponde a la dirección de un bloque de disco llamado bloque indirecto, el cuál contiene direcciones de disco adicionales para archivos más largos. Si esto no fuera suficiente, existe otra dirección de bloque, que indica la posición del bloque doblemente indirecto, quien contiene las direcciones de bloques indirectos. Cada uno de estos bloques indirectos apunta a unas centenas de bloques de datos. Si aún esto no fuera suficiente, puede ser usado el bloque triple indirecto.

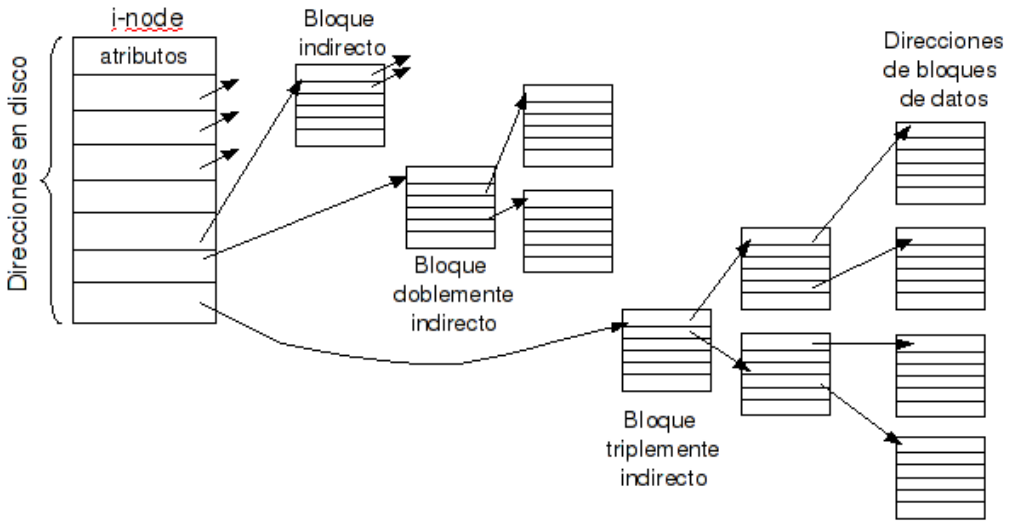

Figura 20: inode.

Los sistemas de archivos suelen estar situados en dispositivos de almacenamiento. Un sistema UNIX puede manejar uno o varios discos físicos, cada uno de los cuáles puede contener uno o varios sistemas de archivos. Los sistemas de archivos son particiones lógicas del disco.

El núcleo del sistema trabaja con el sistema de archivos a nivel lógico y no los trata a nivel físico. Cada disco es tratado como un dispositivo lógico al que se le asocia un número de dispositivo (*minor number* y *major number*). Estos identificadores se utilizan para indexar dentro de una tabla de funciones, que se emplea para acceder al manejador del disco, que es el encargado de transformar las direcciones lógicas (núcleo) del sistema de archivos a direcciones físicas del disco.

El sistema de archivos se compone de una secuencia de bloques lógicos, cada uno de los cuáles tiene un tamaño fijo, que es el mismo para todo el sistema de archivos y suele ser un múltiplo de 512 bytes; algunos valores típicos en bytes de los bloques son: 512, 1024 y 2048. El tamaño elegido para el bloque influye en las presentaciones globales del sistema. A continuación se muestra en un primer nivel de análisis la estructura del un sistema de archivos en el UNIX System V (Figura 21):

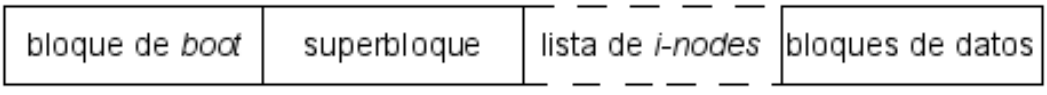

Figura 21: Primer nivel en la estructura del sistema de archivos UNIX.

- El bloque de *boot* o bloque de arranque suele ocupar el primer sector con la primera parte del sistema de archivos y contiene el código de arranque. Este código es un pequeño programa que se encarga de buscar el sistema operativo y

cargarlo en memoria para inicializarlo.

- El *superbloque* describe el estado de un sistema de archivos, contiene información sobre su tamaño, el número total de archivos que puede contener, el espacio libre disponible, etc...
- Lista de *inodes* o nodos índice, se encuentra a continuación del superbloque. Se construye con la información de cada uno de los archivos, es decir, incluye una entrada por cada uno de los archivos en donde se guarda una descripción del mismo: situación del archivo en el disco, propietario, permisos de acceso, fecha de actualización, etc... El tamaño de la lista de inodes es establecida por el administrador del sistema.
- Los bloques de datos están a continuación de la lista de i-nodes y ocupan el resto del sistema de archivos; aquí se encuentra el contenido de los archivos a los que hace referencia la lista de inodes. Cada uno de los bloques destinados a datos solo puede ser asignado a un archivo, aún cuando lo ocupe o no.

6.3 El superbloque

Al arrancar el sistema, se montan los sistemas de archivos locales y remotos y se lee el superbloque a memoria, el cuál tiene la descripción del estado del sistema de archivos – qué tan grande es, cuántos archivos puede almacenar, donde encontrar espacio libre para el sistema y otra información. El kernel lee el superbloque en memoria para acceder y escribir los datos de un sistema de archivos, en forma similar como lee un inode para poder leer y escribir datos de un archivo. La información del superbloque es entre otra la siguiente:

- tamaño del sistema de archivos,
- lista de bloques disponibles,
- índice del siguiente bloque libre en la lista de bloques libres,
- tamaño de la lista de los i-node,
- total de inodes disponibles,
- índice del siguiente inode disponible en la lista de inodes disponibles.
- Indicador que informa si el superbloque ha sido o no modificado.

Cada vez que, desde un proceso, se accede a un archivo, es necesario consultar el superbloque y la listas de inodes. Como el acceso a disco suele degradar bastante el tiempo de ejecución de un programa, lo normal es que el kernel realice la entrada/salida con el disco a través de un *buffer caché* y que el sistema tenga siempre en memoria una copia del superbloque y de la tabla de inodes. Sin embargo, los datos cargados en memoria del superbloque y de la tabla de inodes se actualiza antes que la copia en disco, lo cuál implica que el kernel escriba de manera periódica el superbloque y lista de inodes en el disco y de este modo se preserve la consistencia de los datos del sistema de archivos.

Esta tarea la realiza el demonio *syncer* que se arranca al inicializar el sistema y es el encargado de actualizar periódicamente los datos de administración del sistema. Obviamente, antes de apagar el sistema hay que actualizar el superbloque y la tabla de inodes del disco. Esta tarea es realizada por el programa *shutdown,* que es el encargado de que el sistema sea detenido de manera segura; es muy importante tener siempre en cuenta que el sistema no se detenga sin haber invocado previamente al programa *shutdown*, porque de lo contrario el sistema de archivos podría quedar seriamente dañado.

6.4 I-nodes

No existe una explicación clara para la denominación "i-node",
Dennis Ritchie precursor de UNIX lo explicó así:
'Realmente, tampoco lo sé. Era simplemente el nombre que comenzamos a utilizar:
"Índice" es lo mejor que se me ocurre, debido a la estructura algo inusual de un
sistema de archivos que almacenaba la información del acceso a los archivos
como una lista plana en disco, dejando al margen toda la información jerárquica de los directorios.
Así el número "i"' es un índice sobre la lista, el nodo "i" es el elemento seleccionado de la lista.

La descripción de un archivo en UNIX se guarda en los inode, que son las estructuras básicas del sistema de archivos.

Los i-node contienen los metadatos del archivo que posibilitan a los procesos acceder a los archivos. Comúnmente estos metadatos suelen ser: tamaño del archivo, información sobre su localización, propietario, fecha de creación y de última modificación, permisos, entre otros datos (Figura 22).

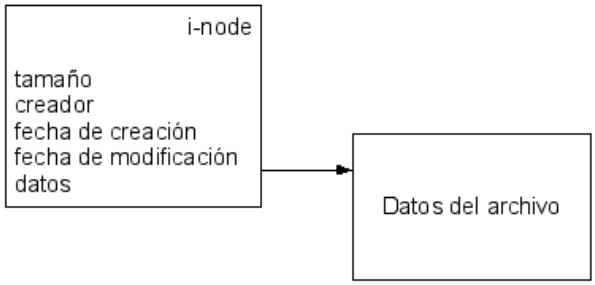

Figura 22: diagrama simplificado de un i-node y los datos a los que hace referencia.

Existen dos tipos de inodes: los estáticos en disco y los de memoria donde el kernel copia los de disco para manipularlos.

Estos dos tipos de estructuras de datos tienen sus propios algoritmos para manejarse tanto de manera individual como en conjunto.

Los i-node en disco contienen los siguientes atributos:

- identificador del propietario del archivo.
- tipo de archivo.
- permisos de acceso.
- tiempos de acceso.
- número de ligas al archivo.
- una tabla de contenido para las direcciones en disco de los datos del archivo.

El i-node en memoria es una copia del inode del disco, pero incluye otros datos adicionales:

- estado del i-node, que indica si está bloqueado, si ha sido modificado o si el archivo es un punto de montura.
- número de dispositivo lógico que contiene al archivo.

- número de i-node.
- dos apuntadores, uno a la lista de inodes libres y uno a su cola de hash.
- un contador de referencias.

Hay que hacer notar que el nombre del archivo no está especificado en el inode, es en los directorios donde cada nombre de archivo de asocia con su inode correspondiente.

Otra cuestión que debe señalarse es la diferencia entre escribir el contenido de un inode en disco y escribir el contenido del archivo. El contenido del archivo – sus datos – cambia sólo cuando se escribe en él. El contenido de un inode cambia cuando se modifican los datos del archivo o la situación administrativa del mismo – propietario, permisos, enlaces, etc. -

La tabla de inodes contiene un listado de todos los números de inode del correspondiente sistema de archivos. Cuando los usuarios buscan un archivo,,el sistema UNIX busca entre la tabla de inodes por el número de inode correspondiente. Cuando el número de inode es encontrado se accede al inode y realizar los cambios correspondientes.

Dado que los inodes se almacenan en un arreglo lineal en disco, el kernel los identifica por su posición en el arreglo; la posición particular en la que se encuentra un inode se llama número de inode.

El sistema de archivos mantiene una lista lineal de inodes en disco cómo se observa en la Figura 21, la cuál se encuentra a continuación del superbloque. Un inode está libre si en su campo de tipo de archivo tiene el valor cero (0). Cuando un proceso necesita un inode, el kernel debería buscar en la lista de inodes por uno libre. Sin embargo, esta actividad resulta costosa y lenta, por lo que para mejorar esta situación el superbloque mantiene un arreglo que sirve de caché para los números de inodes libres del sistema.

El kernel maneja los inodes libres en memoria como una cola circular que funciona como un caché de inodes inactivos; si un proceso intenta acceder a un archivo cuyo inode no está en la cola hash a la que responde ese inode, el kernel toma un inode de la lista de inodes libres y le copia los datos del inode de disco que tuvo que leer, ingresándolo a su cola hash correspondiente.

Las colas de hash de inodes consisten en un arreglo de colas circulares donde el kernel al aplicar una función de hash al número de inode y al número de dispositivo del archivo obtiene la posición en el arreglo, donde inicia la búsqueda del inode.

Cuando se libera un i-node, se decrementa su contador de referencias. Si el contador indica cero referencias, se actualiza el inode en disco si es que fue modificado; después el kernel coloca el inode en la lista de libres y libera todos los bloques de datos en memoria asociados con el archivo así como el inode de disco si es que su número de ligas tiene valor cero.

Durante el arranque el núcleo lee la lista de inodes del disco y carga una copia en memoria que es la tabla de inodes.

Las operaciones que haga el subsistema de archivos -parte del código de *kernel*- sobre los archivos involucra el uso de la tabla de inodes pero no la lista de inodes. Esto proporciona que la velocidad de acceso a los archivos sea mayor puesto que la tabla de inodes está siempre cargada en la memoria. En este caso el demonio *syncer* es también

el encargado de actualizar periódicamente el contenido de la lista de inodes con la tabla de i-nodes.

La tabla de i-nodes contiene la misma información que la lista de i-nodes y agrega información adicional:

- estado del inode que indica:
  - si el inode está bloqueado,
  - si hay algún proceso esperando a que el inode quede desbloqueado,
  - si la copia del inode que hay en memoria difiere de la que hay en el disco,
  - si la copia de los datos del archivo que hay en memoria difiere de los datos que hay en el disco – caso de la escritura en el archivo a través del *buffer caché.*
- el número de dispositivo lógico del sistema de archivos que contiene al archivo.
- el número de inode, relativo a la ista lineal de inodes en disco; el inode del disco no necesita esta información.
- apuntadores a otros inodes cargados en memoria.
- un contador que indica el número de copias de inode que están activas – por ejemplo, porque un archivo esté abierto por varios procesos -.

Además de los metadatos del archivo, los inode contienen apuntadores hacia los primeros bloques de datos del archivo. Si el archivo es grande, el apuntador indirecto contiene un apuntador a un conjunto de apuntadores a los demás bloques de datos. Si el archivo es aún grande, un apuntador doble indirecto contiene un apuntador a un conjunto de apuntadores indirectos a bloques de apuntadores directos a los bloques de datos después de los bloques de datos indirectos. Si se trata de un archivo demasiado grande, un apuntador indirecto triple contiene un apuntador a un bloque de apuntadores indirectos dobles.

6.4.1. Operaciones con i-nodes

Como se ha mencionado un inode es una estructura de datos del sistema de archivos de UNIX y de sistemas de archivos de tipo UNIX como lo es Linux, que tienen por objetivo almacenar toda la información acerca del archivo excepto su nombre y sus datos actuales.

Una estructura de datos es una estructura que permite almacenar datos de manera eficiente. Se utilizan estructuras de datos de acuerdo a las necesidades de las aplicaciones, y algunas de estructuras tienen un alto grado de especialización de acuerdo a las tareas específicas a las que sirven.

Los inodes son manejados en la memoria. Son administrados en una lista circular doblemente ligada empezando con el primer i-node.

Las operaciones que provee la estructura de i-nodes están enfocadas principalmente a manejo de archivos. Estas funciones son principalmente invocadas directamente de las implementaciones propias de las llamadas al sistema (Figura 23).

Figura 23: Algoritmos de bajo nivel del sistema de archivos.

Algunas de estas operaciones son:

- **iget.** Obtiene un inode, leído del cisco a través del buffer caché.
- **iput.** Libera el inode.
- **Bmap.** Fija parámetros del kernel para acceder a un archivo.
- **Namei.** Convierte el nombre de un archivo (i-node) utilizando los algoritmos iget, iput y bmap. Traslada una ruta de archivo a un número de i-nodo.
- **alloc y free.** Asignan y liberan bloques de disco para los archivos.
- **Ialloc e ifree.** Asignan y liberan los i-nodes para los archivos.

6.4.2. Bloque de datos

Los bloques de datos están situados a partir de la lista de inodes. Como se mencionó cada inode tiene entradas – bloques de direcciones – para localizar dónde están los datos de un archivo en el disco. Como cada bloque del disco tiene asociada una dirección, las entradas de direcciones consisten en un conjunto de direcciones de bloques de disco.

Cuando un proceso lee o escribe datos en un archivo, el kernel debe proporcionar los bloques de datos en disco que éste ocupa. Como se observa en la Figura 20 el i-node almacena directa o indirectamente las direcciones de los bloques de datos que ocupa el archivo.

El superbloque contiene un arreglo de números de bloque que funciona como caché de los bloques libres que hay en disco. El programa *mkfs* organiza los bloques de datos libres del sistema en una lista ligada, tal que cada liga de la lista es un bloque de disco, el cual contiene un arreglo con números de bloques libres y una entrada en éste arreglo contiene el siguiente número de bloque de la lista.

Cuando el kernel necesita un bloque del sistema de archivos, toma el siguiente bloque en la lista de bloques libres del superbloque.

Cuando se libera un bloque de datos, si la lista de bloques libres no está llena, el número del bloque liberado se agrega a la lista, si está llena se convierte en una liga a otro bloque; el kernel escribe la lista en ese bloque y lo escribe en disco; después coloca el número del bloque en la lista del superbloque.

6.5 Archivos

A nivel lógico, el sistema de archivos está organizado, en forma de árbol invertido (Figura 24), el nodo que está en la parte superior del sistema de archivos recibe el nombre de nodo raíz del sistema de archivos. El directorio raíz de un proceso denotado por el símbolo “/” es el nodo más alto en la jerarquía del árbol al que el proceso puede tener acceso.

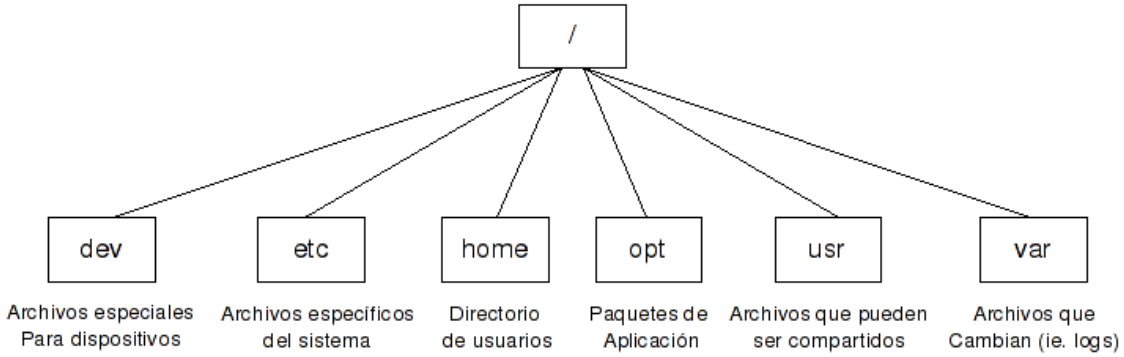

Figura 24: Estructura general del árbol de directorios de UNIX.

Cada nodo dentro del árbol es un directorio y puede contener a otros nodos, llamados subdirectorios, archivos normales o archivos de dispositivos.

Todos los nombres completos de las rutas de acceso en UNIX comienzan con el directorio raíz /.

Algunos de los subdirectorios de la estructura de directorios típica de UNIX incluyen a los subdirectorios *bin* para programas que se ejecutan frecuentemente, *dev* para archivos especiales de dispositivos de entrada/salida, *lib* para archivos de bibliotecas, *usr* para archivos de usuarios, *home* directorios de trabajo de los usuarios, *etc* configuración del sistema.

Los tipos de archivos que soporta UNIX son: regulares, que contienen información del usuario – la mayoría ASCII o binarios - ; los directorios, que son archivos de sistema que mantienen la estructura del sistema de archivos; y los archivos especiales de carácter y de bloque, usados para entrada y salida – llamados también archivos de dispositivos - .

Los archivos tiene asociados una serie de permisos para que sean accedidos; estos pueden colocarse independientemente para controlar la lectura, escritura y permisos de ejecución para tres clases de usuarios: el propietario, el grupo y el resto.

Desde el punto de vista del usuario UNIX trata a los dispositivos como si fueran archivos; los dispositivos se designan por archivos especiales de dispositivo, ocupando los nodos correspondientes en los directorios de la jerarquía, por lo que los programas acceden a los dispositivos con la misma sintaxis que usan para acceder a los archivos regulares.

El sistema de archivos que utiliza UNIX implementa las siguientes llamadas al sistema para manejo de archivos (Tabla 5).

Cada archivo abierto tiene asociado un apuntador al siguiente byte que se leerá o se escribirá. Las llamadas *read* y *write* leen y escriben datos a partir de la posición en el archivo indicada por tal apuntador. Ambas llamadas adelantan el apuntador después de la operación en una cantidad igual al número de bytes transferidos. Sin embargo, se puede acceder aleatoriamente a los archivos asignando explícitamente un valor específico al apuntador del archivo.

La llamada *open* sirve para abrir archivos existentes (y crear archivos nuevos). Se tiene una bandera que indica el modo en el cuál se abrirá el archivo, para lectura, para escritura. La llamada devuelve un entero pequeño llamado **descriptor de archivo** que sirve para identificar al archivo en llamadas subsecuentes. Las llamadas *read* y *wite* cada una tiene un descriptor de archivo que indica que archivo debe usarse, un buffer para colocar los datos leídos o por escribir y una cuenta de bytes que indica cuántos datos deben transmitirse,

Tabla 5: Tabla de llamadas al sistema para manejo de archivos.

Llamada	Descripción
create	El archivo se crea sin datos. El propósito de esta llamada es anunciar que existe el archivo y colocarle atributos.
delete	Cuando el archivo ya no se necesita, el espacio que ocupa en disco es liberado.
open	Antes de utilizar el archivo, un proceso debe de abrirlo. La llamada open permite ubicar los atributos del archivo y mapear las direcciones de disco en memoria principal para acelerar las llamadas posteriores.
close	Cuando se terminan los accesos al archivo, los atributos y direcciones en disco no son necesarios, por lo que el archivo debe cerrarse para liberar el espacio en la tabla interna.
read	Esta llamada sirve para leer los datos del archivo. El usuario debe de especificar cuántos datos se necesitan y proporcionar un buffer para almacenarlos.
write	Se escriben datos al archivo, comúnmente en la posición actual.
append	Llamada restringida del write, ya que solamente puede agregar datos al final del archivo.
seek	Para archivos de acceso aleatorio, se necesita un método para especificar desde donde se tomarán los datos. Esta llamada mueve el apuntador de la posición actual del archivo hacia la posición especificada dentro del archivo. Al concluir ésta llamada, los datos pueden ser leídos o escritos en la nueva posición.
get attributes	Obtiene las propiedades de un archivo.
set attributes	Modifica algunos datos modificables de los archivos.
rename	Cambia el nombre de un archivo existente.
mount	Montar un sistema de archivos.
umount	Desmontar un sistema de archivos.

Cuando se quiere copiar un archivo se utilizan las llamadas *read* y *write*. La llamada *read* tiene como parámetros el descriptor del archivo, un buffer y el contador de bytes; la llamada trata de leer el número de bytes del archivo indicado y colocarlos en el buffer. El número de bytes que realmente se leyeron se devuelve en el contador. La llamada *write* deposita los bytes recién leídos en el archivo de salida. El ciclo continúa hasta que el archivo de entrada se termina de leer, y entonces el ciclo termina y ambos archivos se cierran.

Cada archivo incluidos los directorios, está asociado a un mapa de bits que indica quienes pueden acceder al archivo. El mapa de bits contiene tres campos RWX; el primero controla los permisos de lectura (Read), escritura (Write) y ejecución (eXecute) del dueño; el segundo para los miembros del grupo del propietario del archivo y el último para el resto de los usuarios.

Así RWX R-X--X implica que el dueño tiene todos los privilegios sobre el archivo, los miembros del grupo pueden leerlo y ejecutarlo, y el resto de los usuarios sólo ejecutarlo. La asignación de usuarios a grupos es responsabilidad del rol de *superusuario* del sistema; este superusuario también puede modificar los permisos sobre los archivos de cualquier usuario.

Existen otras llamadas al sistema como *chmod* que cambia el mapa de bits de un archivo, permitiendo o prohibiendo a usuarios distintos del propietario que lo accedan en modo lectura, escritura o ejecución.

Los descriptores de archivo en UNIX son enteros pequeños, en general menores que 20. Los descriptores 0, 1 y 2 son especiales ya que corresponden a la *entrada estándar*, la *salida estándar* y el *error estándar*, respectivamente, las cuáles normalmente se relacionan al teclado, el monitor y la pantalla, también respectivamente, aunque el usuario puede redirigirlas a archivos. Muchos de los programas en UNIX reciben sus entradas de la entrada estándar y escriben los resultados procesados en la salida estándar.

Los programas que se ejecutan en UNIX no conocen el formato interno con el que el núcleo (kernel) almacena los datos. Cuando se accede al contenido de un archivo mediante una llamada al sistema - *read* -, el sistema lo presenta como una secuencia de bytes sin formato. Los programas son los encargados de interpretar estos datos. Por lo tanto, la sintaxis del acceso a los datos de un archivo está impuesta por el sistema y la semántica de los datos es responsabilidad de los programas que maneja los archivos.

El kernel contiene dos estructuras de datos:

1). la tabla de archivos,
2). la tabla de descriptores de archivos de usuarios.

La tabla de archivos es una estructura global del kernel. Del segundo tipo, se crea una por cada proceso.

Cuando un proceso crea o abre un archivo, el kernel crea una entrada en cada una de estas dos tablas, que corresponde al inode del archivo.

Las entradas en la tabla de descriptores, tabla de archivos y tabla de inodes mantienen el estado de los archivos y de los accesos de los usuarios.

La tabla de archivos mantiene el registro del desplazamiento dentro del archivo donde el usuario iniciará la siguiente lectura o escritura y los derechos de acceso para el proceso que lo abrió.

La tabla de descriptores identifica todos los archivos abiertos por un proceso. El kernel regresa un descriptor de archivo por cada llamada a create u open, el cual es un índice en la tabla de descriptores. Cuando se ejecuta una lectura o escritura, el kernel usa ese

número para acceder a la tabla de descriptores, sigue los apuntadores a las entradas de la tabla de archivos y a la tabla de inodes y con el inode encuentra los datos del archivo.

6.6 Directorios

Los directorios son un tipo especial de archivo y son los que proporcionan al sistema de archivos su estructura jerárquica de árbol invertido (Figura 24). Un directorio tiene una estructura de datos que se interpreta y mantiene por el kernel del sistema operativo.

Como cualquier otro archivo, un directorio puede ser abierto y leído, pero para ningún usuario está permitido escribir directamente sobre ellos.

El sistema operativo trata a los datos contenidos en un directorio como un flujo de bytes, esta información tiene un formato que el sistema operativo y los programas que manejan directorios pueden interpretar.

Un directorio contiene un número de entradas, una por cada archivo que esté en el directorio. Cada entrada contiene el nombre del archivo y una referencia al lugar donde se almacena el resto de los datos del archivo, es decir, el número de inode.

Cuando se abre un archivo, el sistema operativo busca en su directorio dentro de la jerarquía del árbol del sistema de archivos hasta que encuentra el nombre del archivo a abrirse. Extrae el número del inode, lo recupera y extrae los atributos y direcciones de disco que ocupa.

Este proceso tiene su coste de desempeño, ya que el sistema operativo carga las entradas del directorio y las entradas de los inode en la memoria principal. Por lo tanto las posteriores referencias se atienden con los datos cargados en la memoria.

Dado que el sistema de archivos se organiza en forma de árbol, los nombres de los archivos se especifican por medio de la ruta (*path name*), que describe cómo localizar un archivo dentro de la jerarquía del sistema. La ruta del archivo puede ser absoluta, en referencia al nodo raíz; o relativa, en referencia al directorio actual de trabajo (Current Work Directory – CWD).

El sistema UNIX que tiene un sistema de directorios jerárquico mantiene dos entradas especiales en cada directorio: *punto* (.) y *doble punto* (..), que hacen referencia al directorio actual y al directorio padre respectivamente.

La Tabla 6 contiene las llamadas al sistema encargadas del manejo de directorios:
Una representación de directorio como la de UNIX que contiene sólo nombres e i-nodes tiene varias ventajas:

- El cambio de nombre de un archivo requiere únicamente la modificación de la entrada en el directorio. El archivo puede moverse de un directorio a otro al trasladar la entrada del directorio siempre que el movimiento mantenga al archivo en la misma partición y segmento. El comando *mv* esta técnica para mover archivos a otros sitios dentro del mismo sistema de archivos dentro de la misma partición.
- Sólo es necesario tener una copia física del archivo en el disco pero el archivo puede tener varios nombres, o el mismo nombre pero encontrarse en diferentes directorios. Todas estas referencias deben de ser sobre la misma partición física.
- La longitud de las entradas del directorio es variable debido a que el nombre del

archivo tiene longitud variable. Las entradas del directorio son pequeñas, puesto que la mayor parte de la información se mantiene en el inode. Entonces, el manejo de estructuras pequeñas de longitud variable puede hacerse con eficiencia. Las estructuras de los inodes son de longitud fija.

Tabla 6: Llamadas al sistema para manejo de directorios.

Llamada	Descripción
create	El directorio se crea vacío. Solo incluye las entradas punto (.) y punto punto (..), que son colocados por el sistema.
delete	Sólo se puede eliminar un directorio vacío; se considera vacío aque directorio que solo incluye las entradas punto (.) y punto punto (..).
opendir	Los directorios pueden leerse como cualquier otro archivo; antes d ser leído debe abrirse de manera análoga a como se abre u archivo regular.
closedir	Cuando se termina de leer un directorio, debe liberarse el espaci que ocupaba en la tabla interna.
readdir	Esta llamada regresa la siguiente entrada de un directorio abierto. La entrada leída regresa en un formato estándar, sin importar la estructura que esté siendo utilizada.
rename	Tal como sucede con los archivos regulares, un directorio puede se renombrado.
link	El *ligado* es una técnica que permite a un archivo aparecer en más de un directorio. Crea una liga desde el archivo existente a otra rut específica.
unlink	Remueve una entrada en un directorio. Si el archivo que se*desliga* está presente sólo en ese directorio, se remueve del sistem completamente. Si está presente en otros directorios, solamente se remueve la ruta especificada.

6.7 Montar un sistema de archivos

Una unidad de disco física consiste de varias secciones lógicas que fueron particionadas por el controlador de disco y donde cada sección posee un nombre de dispositivo. Los procesos pueden acceder a datos de una sección abriendo el dispositivo de archivos apropiado y leyendo o escribiendo el archivo y tratándolo como una secuencia de bloques de disco. Cada sección del disco puede contener un sistema de archivos lógico con la arquitectura mostrada en la Figura 25.

Al montar un sistema de archivos, éste se conecta en una sección específica de disco a la jerarquía del sistema de archivos existente y al desmontarla se desconecta. El kernel posee una tabla de monturas con una entrada para cada sistema de archivos con que cuenta. Cada entrada en la tabla contiene la siguiente información:

- número de dispositivo que identifica al sistema de archivos montado,
- un apuntador al buffer que contiene el superbloque del sistema de archivos,
- un apuntador al inode raíz del sistema de archivos montado,
- un apuntador al inode del directorio del punto de montura.

La llamada mount permite a los usuarios acceder los datos de una sección de disco como un sistema de archivos en vez como secuencia de bloques.

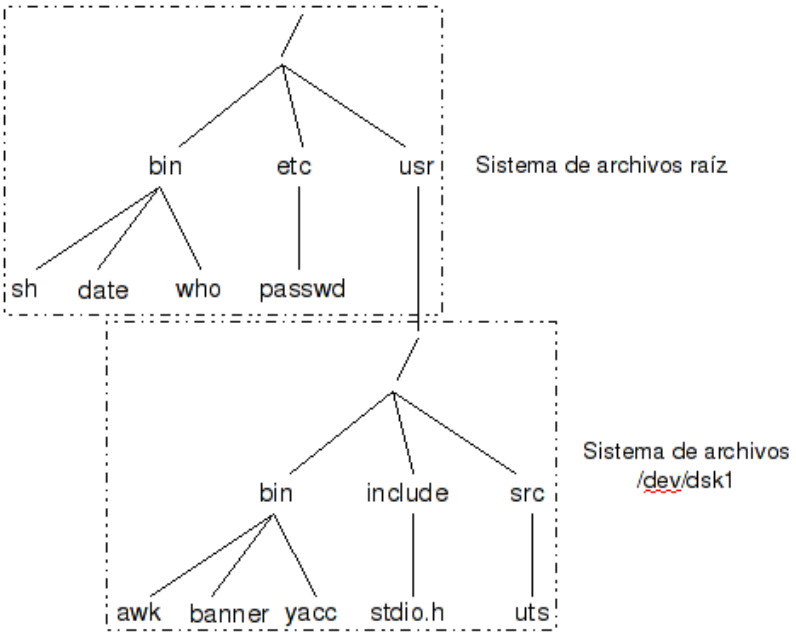

Figura 25: Estructura del sistema de archivos antes y después de montar un sistema de archivos.

6.8 Arquitectura VFS/Vnode

Sun Microsystems introdujo la interfaz MODE/VFS (Virtual File System), es la capa de software en el kernel que provee la interfaz del sistema de archivos a los programas de usuario. Además es la abstracción del kernel que permite que diferentes sistemas de archivos coexistan.

Esta arquitectura es parte del Unix System V en SVR4. Sus objetivos son:

El sistema debe soportar varios tipos de archivos simultáneamente. Incluyendo sistemas de archivos tipo UNIX y no UNIX.
Diferentes particiones de disco pueden contener diferentes tipos de sistema de archivos y a pesar de estar montada una sobre otra debe dar la apariencia de un sistema de archivos homogéneo.
Debe soportar compartir archivos sobre red. Acceder al sistema de archivos de una máquina remota como si fuera una máquina local.
Debe ser posible crear tipos de sistemas de archivos propios y agregarlos al kernel de forma modular.

El principal logro fue proporcionar un marco de trabajo en el kernel para el acceso de archivos, así como su manipulación y una interfaz bien definida entre el kernel y los módulos que implementan los sistemas de archivos específicos.

VFS describe los archivos del sistema en términos de superbloques e inodes tal como lo hace ext2 al utilizar superbloques e inodes. Como en ext2, los inodes de VFS describen a los archivos y directorios del sistema: el contenido y topología de los superbloques de VFS.

El Vnode o nodo virtual representa un archivo en el kernel de UNIX y el VFS o sistema de archivos virtual representa un sistema de archivos. Ambos pueden ser considerados

clases abstractas, de las cuales se pueden hacer implementaciones específicas para diferentes tipos de sistemas e archivos.

El nodo virtual base contiene información que no depende del tipo de sistema de archivos. Las funciones miembro que posee pueden ser divididas en un conjunto de funciones virtuales que definen la interfaz dependiente del sistema de archivos específico y en un conjunto de rutinas utilitarias que pueden ser usadas por otros subsistemas del kernel para manipular archivos, las cuales pueden invocar a las rutinas dependientes del sistema de archivos para realizar tareas de bajo nivel (Figura 26).

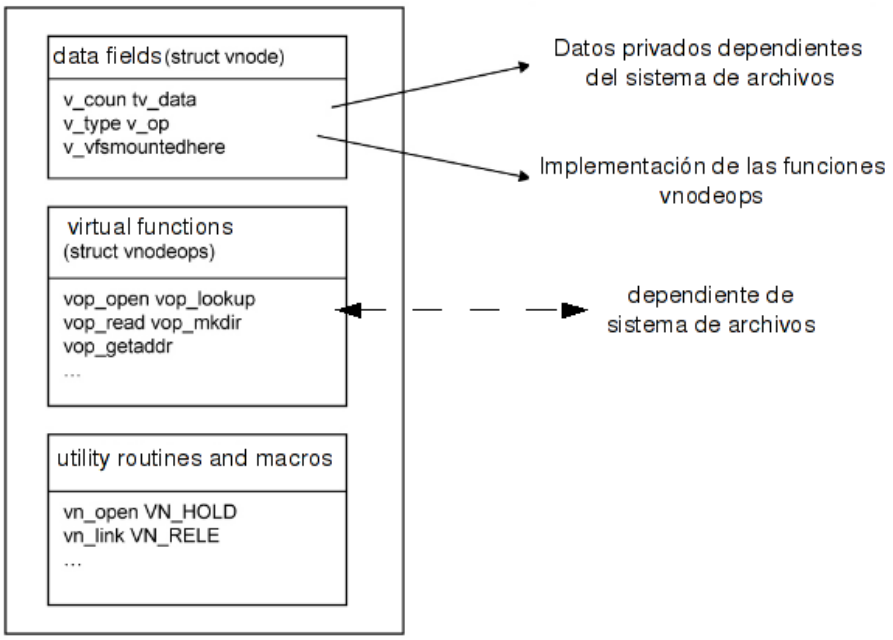

Figura 26: Abstracción del Vnode.

Existe un conjunto de objetivos a cumplir implicados en la implementación de éstas funciones para permitir el desarrollo de una interfaz flexible que pueda ser usada eficientemente por una variedad de sistemas de archivos:

- cada operación debe llevarse a cabo de acuerdo a la conducta del proceso actual.
- ciertas operaciones necesitan serializar el acceso a un archivo.
- la interfaz debe ser sin estado.
- la interfaz debe ser reentrante.
- las implementaciones de sistemas de archivos deben permitirse mas no forzarse a usar recursos globales.
- la interfaz debe ser accesible por el lado del servidor de un sistema de archivos remoto para satisfacer solicitudes del cliente.
- debe evitarse el uso de tablas estáticas de tamaño fijo.

6.8.1.Nodo Virtual

Es la abstracción fundamental que representa un archivo activo en el kernel. Defina la interfaz para el archivo y canaliza todas las operaciones que se realizan sobre él hacia las funciones específicas del sistema de archivos,

El nodo posee un campo que apunta a una estructura de datos privada, la cual mantiene

datos específicos del sistema de archivos; dado que esta estructura se accede indirectamente, es opaca a la clase base y sus campos sólo son visibles para las funciones internas del sistema de archivos específico. Otro campo apunta al vector de operaciones del Vnode, la cual posee las referencias a las funciones que forman la interfaz del nodo virtual.

Cada sistema de archivos implementa ésta interfaz a su manera proporcionando un conjunto de funciones para hacerlo.

Cuando el código independiente del sistema de archivos invoca una función virtual de un nodo arbitrario, el kernel deferencia el apuntador e invoca a la implementación de la función correspondiente a ese sistema de archivos.

Los apuntadores permiten el ligado a las subclases y proporcionan acceso a las funciones y datos dependientes del sistema de archivos en tiempo de corrida.

6.8.2.El objeto vfs

Representa un sistema de archivos. El kernel crea un objeto de éste tipo por cada sistema de archivos activo.

Como el Vnode, el vfs tiene apuntadores a datos privados y a un vector de operaciones. Este último es similar al del vnode, pero lista funciones del sistema de archivos en vez de funciones de archivos.

6.9 Sistema de archivos del sistema operativo MINIX

MINIX es un clon pequeño y gratuito de UNIX diseñado para una tener una fiabilidad muy alta. Es particularmente apropiado para PCs de bajo costo, sistemas con recursos limitados y aplicaciones embebidas. Distribuido junto con su código fuente y desarrollado por el profesor Andrew S. Tanenbaum en 1987. Fue creado para enseñar a sus alumnos el diseño de sistemas operativos en la Vrije de Ámsterdam. La razón de su desarrollo fue porque Unix estaba bajo restricciones de licencia de AT&T,

MINIX apareció en 1987 como un clon pequeño y fácil de entender de UNIX para ser utilizado en cursos de sistemas operativos. Linus Torvalds, entonces un estudiante en la Universidad de Helsinki, estudiaba MINIX en un curso de sistemas operativos y quedó lo suficientemente impresionado como para comprar una PC para poder ejecutarlo. Luego utilizó MINIX como plataforma, guía y fuente de inspiración para desarrollar un clon de MINIX, llamado Linux, el cual liberó públicamente en 1991.

Todo su código fuente está disponible, lo cual lo hace apropiado para su uso en cursos o para aquellos que desean aprender por su cuenta cómo funciona un sistema operativo.

El sistema de archivos del sistema operativo MINIX es un programa desarrollado en lenguaje C que corre en espacio de usuario. Para leer y escribir archivos, los procesos del usuario envían mensajes al sistema de archivos diciendo qué es lo que quieren hacer. El sistema de archivos hace el trabajo y envía de regreso una respuesta. El sistema de archivos es en efecto un servidor de archivos de red que está corriendo en la misma máquina que el cliente.

El sistema de archivos puede ser modificado y probado completamente independiente del resto del sistema operativo.

6.9.1. Mensajes

El sistema de archivos acepta 39 tipos de mensajes solicitando una tarea. Todos excepto dos, son llamadas al sistema. Las dos excepciones son mensajes generados por otras partes de MINIX. De las llamadas al sistema, 31 se aceptan de procesos del usuario y las 6 restantes las recibe primero el manejador de memoria y posteriormente éste llama al sistema de archivos para que haga una parte del trabajo.

Cuando llega un mensaje, se extrae su tipo y éste se usa como un índice en una tabla que maneja todos los tipos y contiene apuntadores a los procedimientos del sistema de archivos. Una vez identificado, se llama al procedimiento apropiado, el cual hace su trabajo y regresa un valor de estado. El sistema archivos envía este mensaje al cliente y regresa al principio del ciclo para esperar otro mensaje.

6.9.2. Capa de sistema de archivos

El sistema de archivos MINIX es una entidad lógica que contiene inodes, directorios y bloques de datos. Puede estar almacenada en cualquier dispositivo de bloque tal como un disco removible o una porción de disco duro. En ambos casos, mantiene la misma estructura (Figura 27).

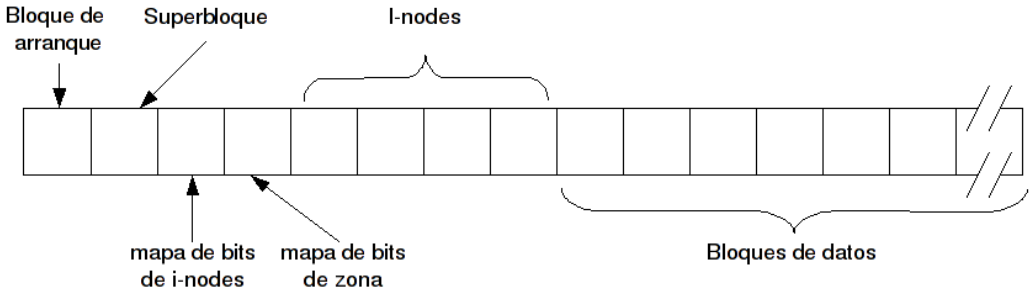

Figura 27: Estructura física del sistema de archivos MINIX.

La función principal del superbloque es informar al sistema de archivos que tan grande son las piezas que lo componen. El almacenamiento en disco puede ser alojado en unidades (zonas) de 1, 2, 4, 8 o en general 2^n bloques. El mapa de bits mantiene el registro de las zonas libres de almacenamiento, no de los bloques. El mapa de bits de zona incluye solamente los datos de la zona, es decir, los bloques que usan los mapas de bits e inodes no están en el mapa.

Cuando MINIX se inicializa, el superbloque del dispositivo raíz se lee en memoria. Similarmente, cuando otros sistemas de archivos son montados, sus superbloques son traídos a memoria. La tabla del superbloque mantiene un número de campos no presentes en el disco. Estos incluyen banderas que permiten especificar que el dispositivo es de solo lectura o que sigue un orden de bytes opuesto al convencional, así como campos que aceleran el acceso, ya que indican partes del mapa de bits donde todos los bits están marcados como usados.

Antes que un disco pueda ser usado por un sistema de archivos MINIX, debe tener la estructura de la Figura 27.

6.9.3.Mapas de bits

MINIX mantiene el registro de los inodes y zonas que están libres usando los mapas de bits. Cuando un archivo se elimina, es simple localizar el bloque del mapa que contiene el bit para el inode que se liberó y recuperarlo usando el mecanismo normal de caché. Una vez que se encuadra el bloque, el bit correspondiente a ese i-node se pone en cero. Las zonas se liberan de la misma forma en el mapa de bits de zona.

Cuando se va a crear un archivo, por lógica el sistema de archivos debería buscar en el mapa de bits el primer inode disponible y alojarlo para el nuevo archivo. Pero el superbloque en memoria tiene un campo que apunta al primer inode libre, así que esta búsqueda es innecesaria hasta después que se usa éste inode; es aquí cuando el apuntador debe actualizarse para apuntar al siguiente nodo libre, el cuál por lo regular es el más cercano o el siguiente. De igual forma, cuando un inode se libera, checa si el inode liberado es menor al primer inode libre y los apuntadores se actualizan si es necesario.

El proceso que se lleva a cabo para los inodes es el mismo que para las zonas.

La idea detrás del uso de las zonas es mantener los bloques de disco que pertenecen a un archivo en el mismo cilindro, lo cual proporciona alto desempeño cuando se lee el archivo secuencialmente. El método elegido es alojar varios bloques a la vez. Si por ejemplo, el tamaño de bloques es de 1K y el de zona es de 4K, entonces el mapa de bits de zona mantiene el registro de zonas, no de bloques.

Muchos de los sistemas de archivos trabajan con bloques. Las transferencias de disco son siempre un bloque a la vez y la caché de búferes también trabaja con bloques individuales. Solo unas pocas partes del sistema de archivos que mantienen el registro de direcciones de disco físicas saben acerca de las zonas.

6.9.4.i-nodes

La estructura de un i-node de MINIX se presenta en la Figura 28.

Los apuntadores de Zona son de 32 bits y hay 7 directos y 2 indirectos. El i-node ocupa 64 bytes, como los i-nodes estándar de UNIX y hay espacio accesible para un décimo apuntador.

Al abrir un archivo, si i-node se traslada a la tabla de i-nodes en memoria, donde permanece hasta que se cierra. La tabla de i-nodes tiene algunos campos adicionales que no están presentes en disco, tal como el dispositivo del i-node y su número, tal que el sistema de archivos sabe donde reescribirlo si fue modificado en memoria. También tiene un contador por i-node, el cuál se incrementa cada vez que se abre una copia del archivo y se decrementa cuando una copia se cierra. Sólo cuando el contador tiene el valor cero, el i-node se elimina de la tabla.

Los primeros siete números de zona están directamente en el i-node. El tamaño máximo de archivo en un sistema de archivos MINIX es un gigabyte.

El i-node mantiene información del modo, el cuál dice que tipo de archivo es y da los bits de protección SETUID y SETGID. El campo de ligas registra cuantas entradas de directorio apuntan al i-node, para que el sistema de archivos sepa cuando liberar el almacenamiento físico del archivo.

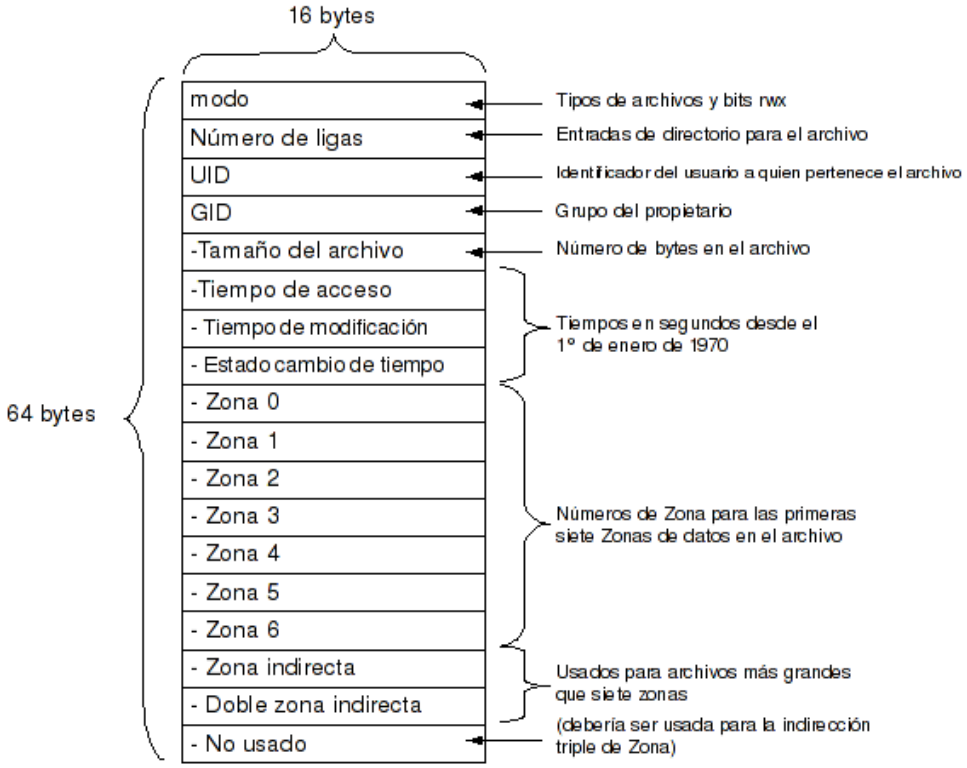

Figura 28: inode de MINIX.

7. Sistemas Linux

Mac OS X's file system is
"complete and utter crap, which is scary."
Linus Torvals

Linux es un sistema operativo de *tipo* Unix. Su desarrollo esta ligado a Minix, lo cuál permitía intercambiar discos entre los dos sistemas de archivos y no había necesidad de crear uno nuevo; MINIX apareció en 1987 como un clon pequeño y fácil de entender de UNIX para ser utilizado en cursos de sistemas operativos. Linus Torvalds, entonces un estudiante en la Universidad de Helsinki, estudiaba MINIX en un curso de sistemas operativos y quedó lo suficientemente impresionado como para comprar una PC para poder ejecutarlo. Luego utilizó MINIX como plataforma, guía y fuente de inspiración para desarrollar un clon de MINIX, llamado Linux, el cuál fue escrito en su primera versión por Linus y que liberó públicamente en 1991.

7.1 Estructura del sistema de archivos de Linux

Utiliza una estructura de árbol jerárquico que representa al sistema de archivos como una entidad única. Linux “monta” cada estructura de archivos en este árbol, cómo un único sistema de archivos virtual integrado, por ejemplo, /mnt/CDROM. Una particularidad de Linux es que puede soportar varios tipos de sistemas de archivos (por ejemplo MS-DOS,

EXT2), lo cual lo hace flexible y lo posibilita a coexistir con otros sistemas operativos.

Los bloques de dispositivos se perciben como colecciones lineares de bloques de datos.

7.2 Elementos estándar del árbol de directorios de Linux

Como se mencionó un sistema Linux reside bajo un árbol jerárquico de directorios muy similar a la estructura del sistema de archivos de plataformas Unix. En los inicios de Linux, este árbol de directorios no seguía un estándar específico, es decir, existían diferencias entre distribuciones. El proyecto FHS (File system Hierarchy Standard), o Estándar de Jerarquía de Sistema de Archivos) fue el antecedente que sentó las bases para el acercamiento a la estandarización del sistema de archivos de Linux, el cuál fue desarrollado por Rusty Russell, Daniel Quinlan y Christopher Yeoh, entre otros en otoño de 1993.

El File system Hierarchy Standard (FHS) es un documento colaborativo que tiene por objetivo definir nombres, localización y estructura de archivos y directorios de las distribuciones del sistemas GNU/Linux y más tarde, alrededor del año 1995, también para su aplicación en sistemas operativos UNIX y distribuciones relacionadas. Este estándar constituye una referencia y no un tutorial sobre como administrar sistemas de archivos o jerarquía de directorios, y representa ventajas de compatibilidad con varias de las distribuciones. El estándar está dirigido a desarrolladores de software independiente y de sistemas operativos, para que adopten una estructura de archivos con un buen nivel de compatibilidad, así como al usuario habitual, para que pueda familiarizarse con el significado y el contendido de cada uno de los elementos del sistema de archivos.

El estándar FHS se considera flexible, es decir, existe libertad al aplicar las normas, lo cuál explica las diferencias existentes entre distribuciones GNU/Linux.

Algunos de los principales objetivos que representa apegarse a FHS son:

- Presentar un sistema de archivos coherente y estandarizado.
- Facilidad para que el software recupere la localización de archivos y directorios instalados.
- Facilidad para que los usuarios recuperen la localización de archivos y directorios instalados.
- Especificar los archivos y directorios mínimos requeridos.

El adoptar una estructura de archivos como FHS permite diferenciar entre los diferentes tipos de archivos que puede haber en un sistema, como lo son los archivos disponibles a los usuarios y aquellos que son propios de la operación del sistema. Por ejemplo, pueden existir archivos propios de un host determinado y, archivos que pueden compartirse entre diferentes host:

- Archivos disponibles para ser compartidos: los contenidos en /var/www/html (que es el DocumentRoot por defecto del servidor Web Apache. Donde se almacena inicialmente el index.html de bienvenida).
- Archivos no disponibles para ser compartidos: los contenidos en /boot/grub/ (Subdirectorio donde se ubican los archivos del gestor de arranque GRUB).
- Archivos estáticos y variables.

Esto es distingue entre los archivos que no cambian sin la interacción de un administrador

del sistema y, archivos que cambian sin la interacción de un administrador del sistema.

Un ejemplo representativo pueden ser los archivos log (archivos de bitácora) del sistema. Estos cambian sin la intervención del administrador, es decir, son archivos variables. Existen además archivos estáticos. No cambian su contenido ni tamaño a menos que lo autorice el administrador del sistema (o sea el propio quien lo modifique):

- Archivos estáticos: /etc/password, /etc/shadow.
- Archivos variables: /var/log/messages (log de mensajes generados por el kernel del sistema).

7.3 Archivos en Linux

Al igual que en UNIX en Linux “todo es una archivo”, esto es, tanto el software como el hardware, es decir, un dispositivo físico como una impresora, un reproductor de DVD un directorio, un subdirectorio y un archivo de texto, se generalizan en una abstracción denominada archivo. Los dispositivos de hardware son son archivos binarios.

Al igual que en UNIX es posible que Linux permita “montar” y “desmontar” volúmenes, como por ejemplo un CDROM, esto es el dispositivo se visualiza como un subdirectorio en el sistema de archivos. En ese subdirectorio se ubicará el contenido del disco compacto cuando esté montado y, nada cuando esté desmontado. El comando *mount* permite realizar las operaciones de montar y desmontar en un distribución GNU/Linux.

Organización del sistema de archivos de acuerdo a FHS
Al igual que en UNIX se tiene el directorio raíz (/), que debe que ser el único directorio en el nivel superior del árbol jerárquico de archivos y del cuál se deriva el sistema de archivos (Figura 29); el contenido de este directorio debe ser el adecuado para reiniciar, restaurar, recuperar y/o reparar el sistema, es decir, debe proporcionar métodos, herramientas y utilidades necesarias para cumplir estas especificaciones.

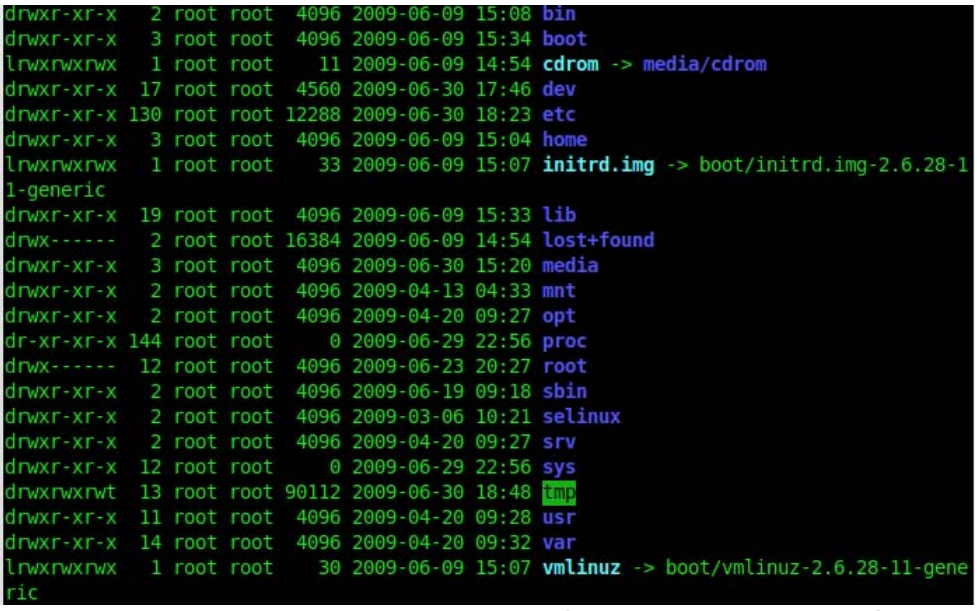

Figura 29: Contenido del directorio raíz (/) de una distribución de Linux (Kubuntu).

Algunos de los directorios contenidos en el directorio raíz se muestran en la Tabla 7:

Tabla 7: Subdirectorios del directorio raíz /).

Subdirectorio	Descripción
/lib	El directorio /lib contiene librerías compartidas (similar a las dll's para los usuarios de Windows) necesarias para arrancar el sistema y para los archivos ejecutables contenidos por ejemplo en, /bin. Normalmente las librerías son archivos binarios escritos en lenguaje C. También contiene módulos del kernel esenciales que permiten el funcionamiento de muchos elementos Hardware. Se ubican normalmente en /lib/modules/versión-del-kernel/.
/media	Contiene los subdirectorios que se utilizan como puntos del montaje para los medios de almacenamiento, tales como disquetes, CD-ROM y memorias USB.
/mnt	Este directorio contiene sistemas de archivos externos que hayan sido montados. Las entidades que aparecen dentro de /mnt representan recursos externos a los que se puede acceder a través de este directorio.
/opt	En este directorio (/opt de OPTIONS) se suelen instalar complementos o add-ons de los programas. Las aplicaciones crean un subdirectorio dentro de /opt denominado con el mismo nombre del programa.
/root	Este directorio es el directorio /home del administrador del sistema (root).
/sbin	Los programas y comandos que se utilizan para la administración del sistema se almacenan en /sbin, /usr/sbin y /usr/local/sbin. /sbin únicamente contiene los ejecutables esenciales para el arranque, recuperación y reparación del sistema. Todos estos directorios (/sbin, /usr/sbin y /usr/local/sbin) se utilizan con fines administrativos, por tanto, sólo puede ejecutar su contenido el administrador.
/srv	Contiene los archivos de datos específicos para cada servicio instalado en el sistema.
/tmp	En este directorio se guardan los archivos temporales.

Subdirectorio	Descripción
/bin	En este directorio se ubica el código binario o compilado de los programas y comandos que pueden utilizar todos los usuarios del sistema. La denominación bin proviene de BINARY.
/boot	Este directorio contiene todo lo necesario para que funcione el proceso de arranque del sistema. /boot almacena los datos que se utilizan antes de que el kernel comience a ejecutar programas en modo usuario. El núcleo del sistema operativo suele situarse en este directorio o en el directorio raíz.
/dev	Este directorio almacena las definiciones de todos los dispositivos. Cada dispositivo tiene asociado un archivo especial. Por ejemplo, el contenido de una sexta partición del disco duro sería /dev/hda5. El archivo asociado a un dispositivo como el ratón (mouse) tipo PS/2 será /dev/psaux. Además, es importante saber que los dispositivos pueden ser de bloque o de carácter. Normalmente los dispositivos de bloque son los que almacenan datos y, los de carácter los que transfieren datos.
/etc	El directorio /etc contiene archivos necesarios para configuración del sistema. Archivos que son propios de la computadora y que se utilizan para controlar el funcionamiento diversos programas. Deben ser ficheros estáticos y nunca pueden ser archivos binarios y/o ejecutables.
/home	Directorio que contiene los subdirectorios que son directorios origen para cada uno de los usuarios del sistema. Cada subdirectorio /home/user de cada usuario proporciona el lugar para almacenar sus archivos, así como los archivos de configuración propios de cada uno. Algunos servicios, y no solo usuarios, tienen aquí su directorio origen, por ejemplo: el servicio de transferencia de ficheros (FTP). El administrador tiene su propio directorio home, que es /root.

El núcleo tiene la capacidad de crear dos entornos o modos de ejecución totalmente separados. Uno de ellos está reservado para el propio kernel, denominado el “modo núcleo”; y el otro está reservado para el resto de programas, llamado el “modo usuario”. Realmente se crean dos entornos totalmente separados, es decir, cada uno tiene su propia zona de memoria y procesos independientes.

Esta técnica ofrece seguridad y estabilidad al sistema. Cuando un proceso del “modo usuario” necesita recursos del “modo kernel” (por ejemplo, acceder a la memoria USB) se hace una llamada al sistema (interfaz que ofrece el núcleo para la comunicación del modo usuario con el modo kernel).

7.4 Estructuras de Almacenamiento i-nodes de UNIX

Un i-node es el lugar en el cuál el sistema de archivos almacena los metadatos (Figura 22) que caracterizan el archivo y contiene los datos de conexión al contenido del archivo. También son conocidos como *FCB* (File Control Block) o *file recor*d.

Los inode hacen referencia al contenido de los archivos por medio de las referencias de la lista de bloques en el disco que pertenecen a cada archivo.

Aunque a alto nivel un archivo se percibe como un flujo continuo de bytes, en realidad los bloques que contienen los datos del archivo, pueden no estar en bloques continuos en el disco. Los inode contienen la información que el sistema de archivos utiliza para relacionar la posición lógica de un archivo con la posición física que ocupa en el disco.

7.5 Sistemas de archivos soportados por Linux

Linux soporta una gran cantidad de tipos diferentes de sistemas de archivos (Tabla 8). Algunos de los más importantes son:

Tabla 8: Sistemas de archivos soportados por Linux.

Sistema de Archivos	Descripción
minix	El más antiguo y supuestamente el más fiable, pero muy limitado en características (algunas marcas de tiempo se pierden, 30 caracteres de longitud máxima para los nombres de los archivos) y restringido en capacidad (como mucho 64 MB de tamaño por sistema de archivos).
xia	Una versión modificada del sistema de archivos minix que eleva los límites de nombres de archivos y tamaño del sistema de archivos, pero por otro lado no introduce características nuevas. No es muy popular, pero se ha verificado que funciona muy bien.
ext3	El sistema de archivos ext3 posee todas las propiedades del sistema de archivos ext2. La diferencia es que se ha añadido una bitácora (journaling). Esto mejora el rendimiento y el tiempo de recuperación en el caso de una caída del sistema. Se ha vuelto más popular que el ext2.
ext2	El más sistema de archivos nativo Linux que posee la mayor cantidad de características. Está diseñado para ser compatible con diseños futuros, así que las nuevas versiones del código del sistema de archivos no necesitará rehacer los sistemas de archivos existentes.
ext	Una versión antigua de ext2 que no es compatible en el futuro. Casi nunca se utiliza en instalaciones nuevas, y la mayoría de la gente que lo utilizaba han migrado sus sistemas de archivos al tipo ext2.

Adicionalmente, existe soporte para sistemas de archivos adicionales ajenos (Tabla 9), para facilitar el intercambio de archivos con otros sistemas operativos. Estos sistemas de archivos ajenos funcionan exactamente como los propios, excepto que pueden carecer de características usuales UNIX , o tienen curiosas limitaciones, u otros inconvenientes.

7.6 Sistema de archivos virtual de Linux

Es una capa de indirección que maneja las llamadas al sistema orientadas a archivos e invoca las funciones correspondientes del sistema de archivos activo para realizar las operaciones de entrada/salida (Figura 30).

El sistema de archivos VFS define un conjunto de funciones que cada sistema tiene que implementar. El sistema de archivos virtual conoce los sistemas de archivos que soporta el kernel, para ello utiliza una tabla de sistemas de archivos registrados, los cuáles se definen durante la configuración del kernel.

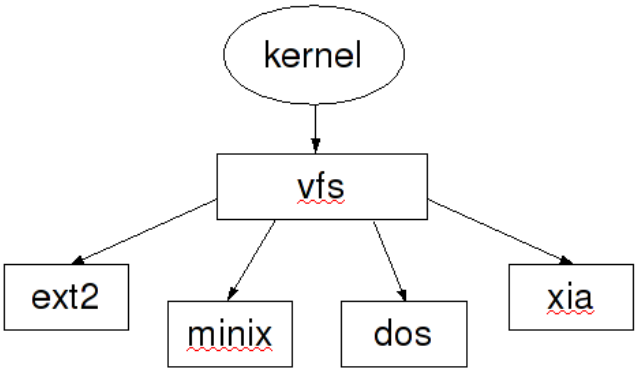

Figura 30: Capa del sistema de archivos virtual.

Tabla 9: Otros sistemas de archivos soportados por Linux.

Sistema de Archivos	Descripción
msdos	Compatibilidad con el sistema de archivos FAT de MS-DOS (y OS/2 y Windows NT).
umsdos	Extiende el dispositivo de sistema de archivos msdos en Linux para obtener nombres de archivo largos, propietarios, permisos, enlaces, y archivos de dispositivo. Esto permite que un sistema de archivos msdos normal pueda utilizarse como si fuera de Linux, eliminando por tanto la necesidad de una partición independiente para Linux.
vfat	Esta es una extensión del sistema de archivos FAT conocida como FAT32. Soporta tamaños de discos mayores que FAT. La mayoría de discos con MS Windows son vfat.
iso9660	El sistema de archivos estándar del CD-ROM; la extensión popular Rock Ridge del estándar del CD-ROM que permite nombres de archivo más largos se soporta de forma automática.
nfs	Un sistema de archivos de red que permite compartir un sistema de archivos entre varios ordenadores para permitir fácil acceso a los archivos de todos ellos.
smbfs	Un sistema de archivos que permite compartir un sistema de archivos con un ordenador MS Windows. Es compatible con los protocolos para compartir archivos de Windows.
hpfs	El sistema de archivos de OS/2.

7.7 Montar un sistema de archivos en Linux

La montura del sistema de archivos raíz es el último paso de la iniciación del sistema de archivos. La llamada al sistema **setup** invoca a la función **mount_root** después de que fueron registradas todas las implementaciones de sistemas de archivos que pueden soportar el kernel. Los sistemas de archivos se registran con estructuras de tipo **file_system_type** (Figura 31).

```
struct file_system_type {
        const char *name;
        int fs_flags;
        struct super_block *(*get_sb)(struct file_system_type *,
                int, char *, void *, struct vfsmount *);
        void (*kill_sb) (struct super_block *);
        struct module *owner;
        struct file_system_type *next;
        struct list_head fs_supers;
        struct lock_class_key s_lock_key;
        struct lock_class_key s_umount_key;
};
```

Figura 31: Estructura tipo file_system_type.

La función mount_root invoca a una función llamada **read_super** que se implementa por cada sistema de archivos registrado, ésta llamada se encarga de establecer el superbloque de su sistema de archivos. La función mount_root lo recibe y lo almacena en un arreglo de superbloques que mantiene en memoria y procede a darlo de alta en el VFS por medio de la llamada **add_vfsmnt.** Esta última llamada se encarga de llenar los datos de la estructura **vfsmount** (Figura 32) y de agregarla a una lista ligada, que contiene los sistemas de archivos montados que mantiene el VFS.

Cuando se monta un sistema de archivos diferente al raíz, se utiliza la llamada mount, que revisa que ese sistema de archivos no esté montado, recupera sus datos de la lista de sistemas de archivos registrados, invoca a la implementación read_super apropiada, llena una estructura vfsmount y la agrega a la lista.

```
struct vfsmount {
        struct list_head mnt_hash;
        struct vfsmount *mnt_parent;    /* fs we are mounted on */
        struct dentry *mnt_mountpoint;  /* dentry of mountpoint */
        struct dentry *mnt_root;        /* root of the mounted tree */
        struct super_block *mnt_sb;     /* pointer to superblock */
        struct list_head mnt_mounts;    /* list of children, anchored here */
        struct list_head mnt_child;     /* and going through their mnt_child */
        atomic_t mnt_count;
        int mnt_flags;
        char *mnt_devname;              /* Name of device e.g. /dev/dsk/hda1 */
        struct list_head mnt_list;
};
```

Figura 32: Estructura tipo vfsmount.

7.8 Sistema de archivos ext2

El sistema de archivos ext2 (second extended filesystem o “segundo sistema de archivos extendido”) permite que los datos de los archivos se almacenen en bloques de datos. Todos los bloques de datos son de la misma longitud; el tamaño de los bloques de datos particular para cada sistema de archivos ext2 se decide cuando se crea (usando mke2fs). El tamaño de cada archivo se redondea hasta un numero entero de bloques. Si el tamaño de bloque es 1024 bytes, entonces un archivo de 1025 bytes ocupará dos bloques de 1024 bytes.

No todos los bloques del sistema de archivos contienen datos, algunos se utilizan para mantener la información que describe la propia estructura del sistema de archivos. El sistema de archivos ext2 define su topología describiendo cada uno de ellos con la estructura de datos inode.

Como se ha mencionado un inode describe que bloques ocupan los datos de un archivo y también sus permisos de acceso, las fechas de modificación y su tipo. Cada archivo en el sistema ext2 se describe por un único inode y cada uno tiene un número que lo identifica. Los inodes del sistema de archivo se almacenan juntos en la tabla de inodes.

Los directorios ext2 son simplemente archivos especiales (ellos mismos descritos por inodes) que contienen punteros a los i-nodes de sus entradas de directorio.

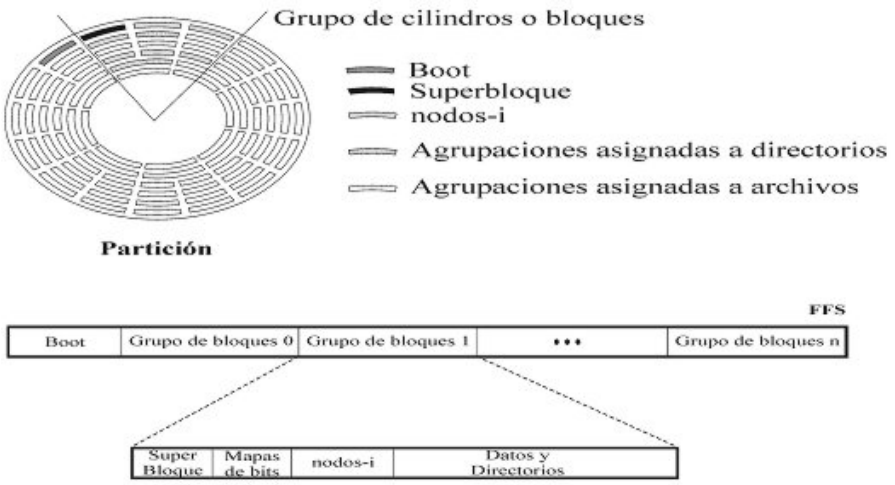

Figura 33: Sistema de archivos ext2.

En la Figura 33 se muestra la disposición del sistema de archivos ext2 ocupando una serie de bloques en un dispositivo estructurado. Para cada sistema de archivos los dispositivos de bloque son sólo una serie de bloques que se pueden leer y escribir. El sistema de archivos no se debe preocupar de donde poner un bloque en el medio físico, eso es trabajo del controlador del dispositivo. Siempre que un sistema de archivos necesita leer información o datos del dispositivo de bloque que los contiene, pide al controlador de dispositivo lea un número entero de bloques.

El sistema de archivos ext2 divide las particiones lógicas que ocupa en grupos de bloque (Block Groups), en los que cada grupo duplica información crítica para la integridad del sistema de archivos, ya sea valiéndose de archivos y directorios como de bloques de información y datos. El duplicar permite al sistema de archivos recuperarse de un desastre.

Algunas de las principales características del sistema de archivos ext2 son:

- Al crear el sistema de archivos, el administrador puede elegir el tamaño de bloque (desde 1KB hasta 4KB), dependiendo de la longitud media esperada de los archivos. Por ejemplo, un bloque de 1KB es preferible cuando la longitud media es menor de uno miles de bytes debido a que produce menos fragmentación interna. Por otro lado, el tamaño grande de bloque es preferible para archivos mayores de miles de bytes dado que producen menos transferencias de disco.
- El administrador puede elegir al crear el sistema cuantos inodes por partición, dependiendo del número de archivos almacenados en él. Esto maximiza el uso efectivo del espacio utilizable de disco.

- El sistema de archivos particiona los bloques de disco en grupos. Cada grupo incluye bloques de datos e inodes almacenados en pistas adyacentes. Gracias a esta estructura, los archivos en un único grupo de bloques pueden ser accedidos con un tiempo de búsqueda medio menor.
- El sistema de archivos preasigna bloques de datos de disco a archivos regulares antes de que estos se utilicen. Así, cuando un archivo incrementa su tamaño, varios bloques ya están reservados en posiciones físicas adyacentes, reduciendo la fragmentación del archivo.
- Soporta enlaces simbólicos rápidos. Si el nombre de camino del enlace simbólico tiene 60 bytes o menos, se almacena en el inode y puede así traducirse sin leer un bloque de datos.
- Una implementación cuidadosa de la estrategia de actualización de archivos que minimiza el impacto de las caídas del sistema.
- Soporte para comprobaciones automáticas de consistencia sobre el estado del sistema de archivos en el arranque del sistema. Estas comprobaciones se realiza mediante el programa /sbin/e2fsck, que puede activarse no solo tras caídas del sistema, sino después de un número predefinido de montajes (se incrementa un contador después de cada operación de montaje), o después de cierta cantidad de tiempo transcurrida desde la comprobación más reciente.
- Soporte de archivos inmutables (no pueden ser modificados) y para archivos solo-añadir (solo podemos añadir datos al final de archivo). Ni el superusuario puede sobrepasar estas clases de protección.
- Compatibilidad con las semánticas de los ID de grupo de un nuevo archivo de los sistemas de archivos de System V y BSD. En System V un nuevo archivo asume el ID de Grupo del proceso que lo crea; en BSD, un nuevo archivo hereda el ID de Grupo del directorio que lo contiene. Ext2 incluye una opción de montaje que especifica qué semántica utilizar.

7.9 Sistema de archivos ext3

Es una versión mejorada de ext2, que incorpora el concepto de bitácora (**journal**), es decir, es una extensión con journaling del sistema de archivos ext2. Con el journaling se obtiene una enorme reducción en el tiempo necesario para recuperar un sistema de archivos después de un cierre no limpio. Es un sistema de archivos recomendable en entornos donde la alta disponibilidad es importante.

Las mejoras introducidas proporcionan las siguientes ventajas:

- **Disponibilidad:** Tras un corte eléctrico o una caída inesperada del sistema (también se denomina *cierre no limpio del sistema*), se debe comprobar con el programa e2fsck cada sistema de archivos ext2 montado en la máquina para ver si es consistente. El proceso de comprobación lleva mucho tiempo y puede prolongar el tiempo de arranque del sistema de un modo significativo, especialmente si hay grandes volúmenes que contienen un elevado número de archivos. Durante este proceso, no se puede acceder a los datos de los volúmenes. Con la característica ***journaling*** del sistema de archivos ext3 ya no es necesario realizar este tipo de comprobación en el sistema de archivos después de un cierre no limpio del sistema. En el sistema ext3, únicamente se realiza una comprobación de consistencia en los casos puntuales en los que se producen determinados errores de hardware, como, por ejemplo, fallos en el disco duro. El tiempo empleado para recuperar un sistema de archivos ext3 tras un cierre no limpio del sistema no depende del tamaño del sistema de archivos ni del número de archivos, sino del

tamaño del *journal* (diario), utilizado para mantener la consistencia en el sistema. Por defecto, la recuperación del tamaño del "journal" tarda alrededor de un segundo, según la velocidad del hardware.

- **Integridad de los datos:** El sistema de archivos ext3 proporciona una integridad superior de los datos si se produce un cierre no limpio del sistema. El sistema de archivos ext3 permite seleccionar el tipo y el nivel de protección de los datos. Por defecto, los volúmenes ext3 son configurados para mantener un nivel de consistencia de los datos elevado en relación con el estado del sistema de archivos.
- **Velocidad:** El sistema de archivos ext3, aparte de permitir escribir datos más de una vez, en la mayoría de los casos tiene un rendimiento superior al que proporciona ext2 porque los "journals" de ext3 optimizan el movimiento de los cabezales de los discos duros. Se pueden seleccionar tres modos de journaling para optimizar la velocidad, pero, como contrapartida, la integridad de los datos se verá afectada.
- **Fácil transición:** La migración de ext2 a ext3 es muy sencilla y se pueden aprovechar las ventajas de un sólido sistema de archivos con journaling sin tener que volver a dar formato al sistema.

Si realiza una instalación nueva, el sistema de archivos que se asigna por defecto a las particiones Linux del sistema es ext3. Si realiza una actualización a una versión que usa particiones ext2, el programa de instalación le permitirá convertir estas particiones a ext3 sin perder los datos.

Es posible pasar de un sistema de archivos ext2 a uno ext3. El programa tune2fs permite añadir un journal a un sistema de archivos ext2 existente sin modificar los datos en la partición. Si el sistema de archivos ya está montado mientras se realiza la migración, el journal estará visible como .journal en el directorio raíz del sistema de archivos. Si el sistema de archivos no está montado, el journal se ocultará y no aparecerá en el sistema de archivos.

7.10 NFS (Network File System)

El sistema de archivos de red NFS de Sun, utilizado por Linux que se encarga del acceso a archivos en una red. Su origen es de alrededor de 1985 y su objetivo es ofrecer acceso eficiente y transparente a sistemas de archivos remotos en una red heterogénea, que da la impresión de trabajar sobre un único disco duro local. Algunas distribuciones de Linux pueden trabajar como servidor o como cliente de NFS, lo que implica que pueden exportar sistemas de archivos a otros sistemas, así como montar los sistemas de archivos que otras máquinas exportan.

NFS resulta útil para compartir directorios de archivos entre múltiples usuarios de la misma red. Por ejemplo, un grupo de usuarios que trabajan en un mismo proyecto pueden tener acceso a los archivos de ese proyecto usando una porción compartida del sistema de archivos NFS (conocido como NFS share), que se ha montado en un directorio determinado, como puede ser **/myproject.** Para acceder a los archivos compartidos; el usuario accede al directorio **/myproject** de su máquina local. No tendrá que introducir contraseñas o memorizar comandos especiales. El usuario podrá trabajar como si el directorio estuviese en su máquina local (Figura 34).

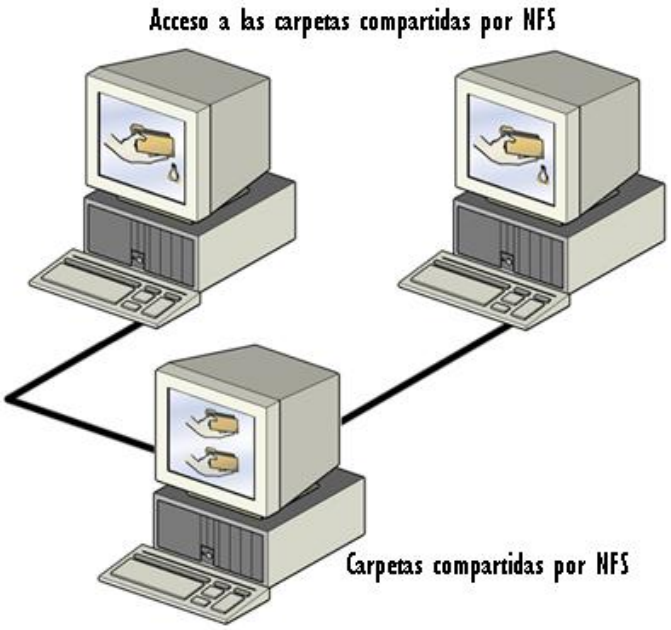

Figura 34: Acceso por medio de NFS.

NFS es un servicio de red que permite a una computadora cliente montar y acceder a un sistema de archivos (en concreto, un *directorio* remoto), exportado por otra computadora de tipo servidor. NFS es soportado por Mac OS X de Apple, y los clientes NFS (y servidores de NFS) se encuentran disponibles incluso para sistemas Windows.

Los sistemas de archivos en red ofrecen varias ventajas sobre los sistemas de archivos locales:

- Los datos e información pueden estar distribuidos en los equipos de la red reduciendo el riesgo de que el fallo en la máquina local impida acceder a los datos. Los sistemas de archivos en red permiten ingresar en múltiples máquinas y acceder a los datos exactamente de la misma forma en todas ellas.
- Proporcionan ubicaciones centralizadas para los datos, que están compartidas entre los usuarios.
- Simplifican el acceso a los datos existentes en sistemas más veloces. Las aplicaciones se pueden ejecutar en una máquina más rápida y potente con solo ingresar como usuario de ese equipo y ejecutar la aplicación desde el sistema de archivos en red.
- Proporcionan la oportunidad de centralizar operaciones administrativas, tales como la copia de seguridad de los datos (backup).
- Proporcionan interoperabilidad y flexibilidad. Normalmente es posible acceder a sistemas de archivos en red desde equipos que ejecuten Linux, Windows, Mac OS X, BeOS, BSD entre otros.

Para compartir directorios sobre NFS debe montar el sistema de archivos de una máquina remota en el sistema de archivos local por medio del comando mount. El directorio local debe existir previamente para que el montaje pueda realizarse. La opción -t nfs indica a mount el tipo de sistema de archivos que va a montar.

Una vez el montaje se ha realizado, cualquier acceso a archivos o directorios (lectura,

escritura, cambio de directorio, etc.) se traduce de forma transparente a peticiones al equipo servidor, el cuál las atenderá y devolverá su respuesta al cliente, todo a través de la red. Es decir, el montaje de directorios mediante NFS permite trabajar con archivos remotos exactamente igual que si fueran locales, aunque lógicamente con una menor velocidad de respuesta.

En general, NFS es flexible, y trabaja adecuadamente en los escenarios como:

- Un servidor NFS puede exportar más de un directorio y atender simultáneamente a varios clientes.
- Un cliente NFS puede montar directorios remotos exportados por diferentes servidores.
- Cualquier sistema UNIX puede ser a la vez cliente y servidor NFS.

En el sistema cliente, el funcionamiento de NFS está basado en la capacidad de traducir los accesos de las aplicaciones a un sistema de archivos en peticiones al servidor correspondiente a través de la red. Esta funcionalidad del cliente se encuentra normalmente programada en el núcleo de Linux, por lo que no necesita ningún tipo de configuración.

Respecto al servidor, NFS se implementa mediante dos servicios de red, denominados **mountd** y **nfsd:**

El servicio **mountd** se encarga de atender las peticiones remotas de montaje, realizadas por la orden mount del cliente. Entre otras cosas, este servicio se encarga de comprobar si la petición de montaje es válida y de controlar bajo qué condiciones se va a acceder al directorio exportado (sólo lectura, lectura/escritura, etc.). Una petición se considera válida cuando el directorio solicitado ha sido explícitamente exportado y el cliente tiene permisos suficientes para montar dicho directorio.

Por su parte, y una vez un directorio remoto ha sido montado con éxito, el servicio **nfsd** se dedica a atender y resolver las peticiones de acceso del cliente a archivos situados en el directorio.

NFS asume un sistema de archivos jerárquico(directorios). Los archivos son secuencias de bytes sin estructura y carentes de significado inherente: es decir, todos lo archivos se ven como una secuencia contigua de bytes, sin ninguna estructura de registros.

Con NFS, todas las operaciones sobre archivos son *síncronas*. Esto significa que la operación sólo retorna cuando el servidor ha completado todo el trabajo asociado para esa operación. En caso de una solicitud de escritura, el servidor escribirá físicamente los datos en el disco, y si es necesario, actualizará a estructura de directorios, antes de devolver una respuesta al cliente. Esto garantiza la integridad de los archivos.

Un servidor NFS no necesita mantener información extra de ninguno de sus clientes para funcionar correctamente. En caso de un fallo del servidor, los clientes sólo tienen que reintentar la solicitud hasta que el servidor responda, sin tener que repetir la operación de mount.

8. CODA

Sistema de archivos distribuido, desarrollado en la Universidad Carnegie Mellon por M.

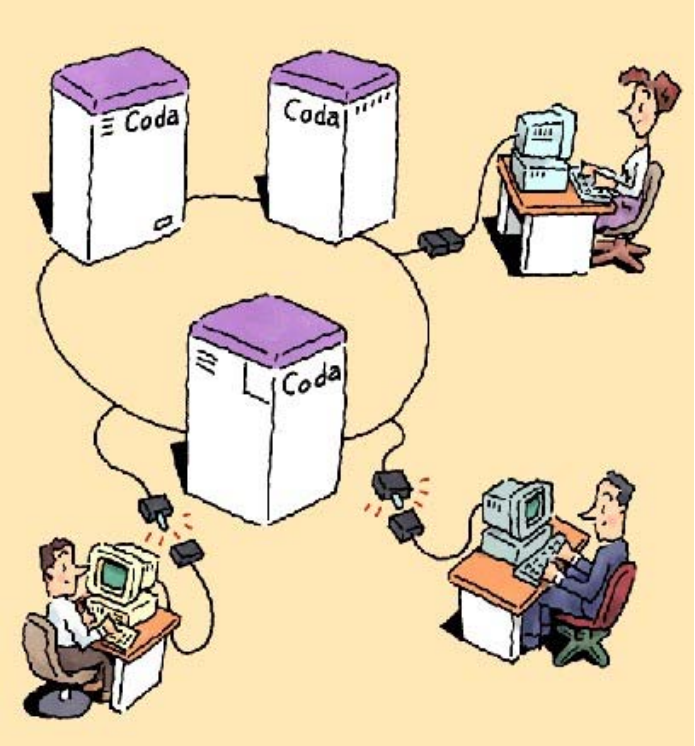

Figura 35: Logo de CODA (ilustración de Gaich Muramatsu).

Satyanarayanan y su grupo de colaboradores. Está basado en una arquitectura cliente/servidor, y ha sido diseñado para proporcionar funcionalidades no existentes en otros sistemas de archivos distribuidos – por ejemplo NFS – como es el soporte para la movilidad de los clientes.

Algunas de las características relevantes de este sistema de archivos son:

- Permite operar en modo de desconexión, es decir, desconectados del sistema, de manera que pueden realizar cambios locales en los archivos que se propagan a todo el sistema de archivos una vez que el cliente en cuestión se vuelve a conectar.
- Facilita características para mejorar la resistencia a fallos en el sistema, mediante la posibilidad de incluir servidores con réplicas de los datos del sistema, y mecanismos para manejar conflictos entre servidores, para manejar fallos en la red, y para controlar la desconexión de los clientes y los posibles conflictos e inconsistencias que ello puede acarrear. También incorpora mecanismos de respaldo (backup) de los datos del sistema.
- Proporciona mejoras en el rendimiento, con respecto a otros sistemas de archivos distribuidos, mediante la utilización de caches locales en los clientes y para la escritura de datos en los servidores.
- Incorpora características de seguridad basadas fundamentalmente en la autenticación de usuarios, mediante la utilización de listas de control de acceso (ACL).
- El código está disponible de forma libre.

El objetivo principal de este sistema de archivos mejor rendimiento, menos sensible a la

no disponibilidad en los servidores o a sobrecargas de la red. Relacionada con esta meta está la provisión de cierto grado de movilidad a los clientes del sistema.
Actualmente existen versiones de CODA para diferentes variantes de Unix y en distribuciones de Linux (ie. Debian y RedHat) y Windows 95/98 y XP.

Si un cliente CODA se ejecutara en una estación de trabajo con Linux como sistema operativo, mediante el comando /mount nos mostraría un sistema de archivos de nombre Coda montado en:

/coda

Todos los archivos que el servidor ponga a disposición del cliente estarán bajo este directorio. El cliente se conecta a “Coda” y no a servidor individual, lo que hace que la implementación sea invisible al usuario. Esto es una diferencia notable a la de montar un NFS que se hace por servidor.

Cuando se hace una operación sobre un archivo se hacen las llamadas correspondientes al sistema para atender dicha operación. Las llamadas al sistema son operaciones por medio de las cuales un programa solicita servicios al kernel, como por ejemplo, cuando se abre un archivo, el kernel realizará una operación de búsqueda para localizar el inode de un archivo y regresará el dato correspondiente al programa que solicitó la operación. Como ya se ha expuesto, el inode contiene la información para acceder a los datos en el archivo y es utilizada por el kernel. La llamado open entra al VFS del kernel, y cuando se percata de que la solicitud es por un archivo en el sistema de archivos /coda es manejada por el módulo del kernel correspondiente para el sistema de archivos CODA. CODA se considera un módulo de sistema de archivos minimalísta: mantiene un cache de las peticiones más recientes que ha resuelto VFS, o de otro modo envía las solicitudes a administrador de cache de CODA (Coda cache manager), conocido como **Venus**.

Venus revisará el cache del por ejemplo, tmp/foo, y en caso de que dicho cache no se encuentre, contactará al servidor para preguntar por tmp/foo. Cuando el archivo sea localizado, Venus responde al kernel, que a cambio, regresa la llamada al programa que hizo la llamada al sistema (Figura 36).

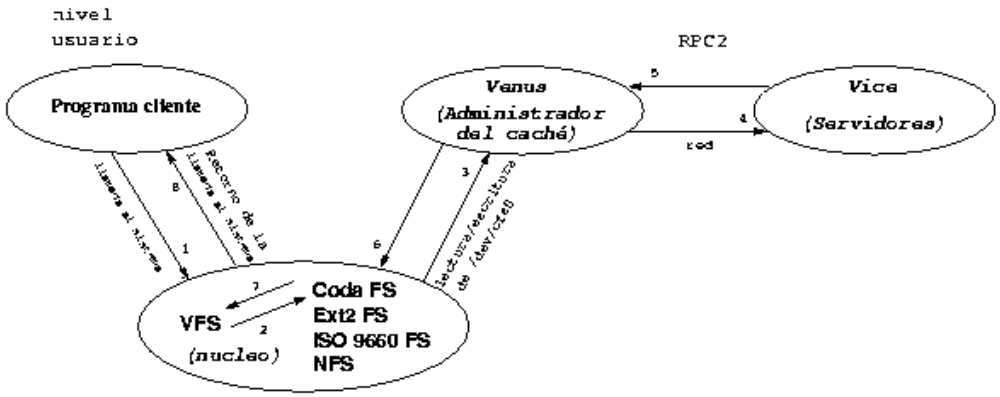

Figura 36. Operación del Administrador de Cache de Coda

La Figura anterior muestra cómo un programa usuario pide un servicio del kernel a través de una llamada al sistema. El kernel la pasa a Venus, permitiéndole leer la petición desde el dispositivo de caracteres /dev/cfs0. Venus trata de resolver la petición buscando en su caché, preguntando a los servidores o posiblemente declarando desconexión y sirviendo en modo de desconexión. El modo de desconexión entra en funcionamiento cuando no hay conexión de red a cualquier servidor que tenga archivos. Esto ocurre típicamente cuando los clientes se desconectan de la red o cuando existen fallos de red. Si la operación falla por una desconexión de los servidores también entraría en funcionamiento.

Cuando el kernel pasa la petición de apertura a Venus por primera vez, Venus obtiene el archivo entero desde los servidores, usando llamadas a procedimientos remotos para contactar con ellos. Entonces almacena el archivo como un contenedor en la zona de caché (actualmente /usr/coda/venus.chache). El archivo es ahora un archivo normal en el disco local, y las operaciones de lectura y escritura no llegarían a Venus y serían manejadas de manera completa por el sistema de archivos local (ie. ext2 en *Linux*). Las operaciones de lectura/escritura de CODA funcionarán a la misma velocidad que en los archivos locales. Si el archivo se abre por segunda vez, no se traerá de los servidores otra vez y la copia local estará disponible para su uso inmediatamente.

Los archivos de directorio (recordemos que un directorio es únicamente un archivo) así como todos los atributos (dueños, permisos y tamaños) son todos cacheados por Venus y Venus permite que las operaciones puedan continuar sin contactar con el servidor si los archivos están presentes en la caché. Si el archivo se ha modificado y ha sido cerrado, entonces Venus actualiza los servidores enviándoles el nuevo archivo. Otras operaciones que pueden modificar el sistema de archivos, como la creación de directorios, el borrado de archivos o directorios y la creación o borrado (simbólico) de enlaces, también son propagadas al servidor.

Como se puede apreciar, CODA cachea toda la información que necesita en el cliente y sólo informa al servidor de los cambios que deben realizarse al sistema de archivos. Los estudios han demostrado que las modificaciones son bastante menos habituales que el acceso s*ólo de lectura* a los archivos. Estos mecanismos para cachear datos de una manera *"agresiva"* fueron implementados en AFS y DFS, pero muchos otros sistemas tienen un sistema de caché más rudimentario.

9. Glosario

Agendado (“Scheduling”)
Procedimiento que realiza el núcleo del sistema operativo y particularmente el módulo conocido como el agendador (“scheduler”) que consiste en darle tiempo y horario de ejecución a cada uno de los procesos que están corriendo en un sistema computacional, en un espacio de tiempo dado.

Agendador (“Scheduler”)
Módulo del núcleo del sistema operativo que se encarga de administrar los tiempos y horarios de ejecución de los hilos y procesos corriendo en el sistema computacional.

Alternancia (“swapping”)
Es la acción de intercambiar aplicaciones cargadas en la memoria principal de trabajo del

sistema de cómputo o memoria RAM a la memoria temporal de alternancia (“memoria de swapping”) y viceversa; lo anterior con el propósito de darle turno de ejecución a la aplicación que se encontraba lista para correr pero en espera de memoria de trabajo.

Archivo
Datos estructurados que pueden recuperarse fácilmente y usarse en una aplicación determinada. Se utiliza archivo como sinónimo. El archivo no contiene elementos de la aplicación que lo crea, sólo los datos o información con los que trabaja el usuario.

Conjunto de bytes almacenados como una entidad individual. Todos los datos en disco se almacenan como un archivo con un nombre de archivo asignado que es único dentro del directorio en que reside.

Para la computadora, un archivo no es más que una serie de bytes. La estructura de archivo es conocida para el software que lo maneja. Por ejemplo, los archivos de bases de datos están compuestos por una serie de registros. Los archivos de procesamiento de texto, también llamados documentos, contienen un flujo continuo de texto.

Para la familia de sistemas operativos Unix – Linux, los periféricos como teclados, monitores, escáneres, impresoras, etcétera, son manejados como si fueran archivos.

Bit
Bit es el acrónimo de Binary digit. (Dígito binario). Un bit es un dígito del sistema de numeración binario. El bit es la unidad mínima de información empleada en informática, en cualquier dispositivo digital, o en la teoría de la información. Con él, podemos representar dos valores cualesquiera, como verdadero o falso, abierto o cerrado, blanco o negro, norte o sur, masculino o femenino, rojo o azul, etc. Basta con asignar uno de los dos valores (0,1) a cada uno de las dos opciones.

Bloque
El sistema de archivos se compone de una secuencia de bloques lógicos, cada uno de los cuáles tiene un tamaño fijo, que es el mismo para todo el sistema de archivos y suele ser un múltiplo de 512 bytes; algunos valores típicos en bytes de los bloques son: 512, 1024 y 2048. El tamaño elegido para el bloque influye en las prestaciones globales del sistema.

Bloque de Arranque
Un bloque de arranque (a veces llamado sector de arranque o MBR) es un sector en un disco duro, disquete, o cualquier otro dispositivo de almacenamiento de datos que contiene código de arranque, por lo general (pero no necesariamente), de un sistema operativo almacenado en otros sectores del disco. El término sector de arranque es usado para los equipos compatibles con la IBM PC, mientras que el término bloque de arranque es usado cuando se refiere a otros tipos de computadoras, como los sistemas de Sun Microsystems.

El BIOS selecciona el dispositivo de arranque, y luego copia el primer sector de disco de ese dispositivo (el cual puede ser un MBR, VBR o un código ejecutable), a la ubicación a partir de la dirección de disco 0x7C00.

Bloque de Datos
Los bloques de datos están situados a partir de la lista de i-nodes. Cada i-node tiene entradas – bloques de direcciones – para localizar dónde están los datos de un archivo en el disco. Como cada bloque del disco tiene asociada una dirección, las entradas de

direcciones consisten en un conjunto de direcciones de bloques de disco.

Byte, Octeto
Conjunto de ocho bits que se ha utilizado como unidad de capacidad de almacenamiento de la información en la memoria principal de la computadora o en las unidades de almacenamiento periféricas.

Cantidades de Bits

Cantidades de bits		
Prefijo Sistema Internacional		
Nombre (Símbolo)	**Estándar** SI	Uso Binario
kilobit (kbit)	10^3	2^{10}
megabit (Mbit)	10^6	2^{20}
gigabit (Gbit)	10^9	2^{30}
terabit (Tbit)	10^{12}	2^{40}
petabit (Pbit)	10^{15}	2^{50}
exabit (Ebit)	10^{18}	2^{60}
zettabit (Zbit)	10^{21}	2^{70}
yottabit (Ybit)	10^{24}	2^{80}

Control de Acceso
Mecanismo que hace posible limitar el acceso de usuarios no autorizados a un archivo; para evitar que lo altere, elimine o pueda obtener información confidencial.

CP/M (“Control Program for Microcomputers”)
Sistema operativo desarrollado en 1973 por Gary Kildall que permitía guardar archivos y ejecutar programas desde un disco Shugart de 8 pulgadas, el cual podía llevarse de la computadora de casa a la del trabajo. Tenía un sistema de archivos plano, sin directorios, que permitía guardar archivos cuyos nombres estaban limitados a 8 caracteres, mas 3 de la extensión, está extensión determinaba el tipo de archivo (bas, exe, …)

Cúmulos (“Clusters”)
En el caso de un disco magnético de memoria de almacenamiento, un cúmulo o clúster es un número fijo de sectores de 512 octetos (bytes) al que puede hacer referencia directamente un sistema de archivos.

Hablando de sistemas de cómputo un cúmulo (o “cluster” en inglés) es una agrupación de procesadores que el sistema operativo puede organizar y sincronizar para ejecutar tareas de cómputo paralelo. Estos procesadores están interconectados por medio de un mecanismo de red de comunicaciones que puede ser un conmutador electrónico (sistema

crossbar) o una red de área local; por ejemplo Gigaethernet.

En reconocimiento de patrones, un cúmulo o cluster es una agrupación de elementos de un conjunto, cuyas características son muy similares; de tal manera que pueden clasificarse como elementos que reúnen las características esenciales del cúmulo.

DEC (“Digital Equipment Corporation”)
Empresa que desarrolló una importante familia de minicomputadoras que dominaron el mercado de la computación desde finales de la década de los 70 hasta finales de la de los 80. Sus principales marcas fueron PDP y VAX. También desarrollaron los sistemas operativos, software utilitario y periféricos que permitieron constituir sistemas altamente productivos.

Demonio (“demon”)
Programa del sistema que se está ejecutando en segundo plano (background) y realiza tareas periódicas relacionadas con la administración del sistema.

Desborde de archivo de área contigua (“extents overflow file”)
Situación en la cual un archivo tiene una extensión que excede el tamaño de el área contigua de memoria que le fue asignada; por lo cual algunos fragmentos del archiva deben desbordarse para almacenarse en áreas de almacenamiento discontinuas.

Extent
Es un área contigua del almacenamiento de un sistema de archivos de computadora

I-node
Es una estructura de datos de un sistema tradicional de un sistema de archivos tipo UNIX tal como UFS o ext3. Un i-node almacena la información básica sobre archivos regulares, directorio u otro objeto del sistema de archivos.

Manejador (“Drive”)
Software que se encarga de implementar el intercambio de información y comandos entre el hardware de dispositivos como el teclado, el monitor, impresoras, escáneres, etcétera y el sistema operativo. Es el software que se encarga de controlar los dispositivos de entrada salida de la computadora.

Memoria de Alternancia (“Memory Swapping”)
Área de memoria secundaria o memoria en disco que se utiliza como almacenamiento temporal de una aplicación que se encuentra en espera del recurso de memoria principal para estar lista para correr.

Multitarea (“multitasking”)
Funcionalidad del sistema operativo que permite ejecutar concurrentemente dos o más tareas, compartiendo así el procesador entre dichas tareas y haciendo más eficiente su uso.

Multihilo (“Multitread”)
Funcionalidad del sistema operativo que permite ejecutar concurrentemente dos o más instancias del mismo software. El control de secuencia de ejecución del programa forma parte del contexto y por ende de la integridad del hilo de ejecución o instancia de ejecución del software. Los hilos de ejecución comparten el código del programa mientras están ejecutándose.

Partición
Es el conjunto de cúmulos o clústeres de memoria que tiene a bajo su administración un sistema operativo; para proporcionar a los procesos e hilos la memoria que requieren para poderse ejecutar.

PC IBM
Computadora personal de escritorio lanzada al mercado en 1984 por la IBM Corporation.

PDP ("Programmed Data Processor")
Minicomputadora de amplia penetración en el mercado en la década de los 80, fabricada por DEC. Se utilizó ampliamente en sistemas administrativos y aplicaciones industriales como sistemas de posproducción de televisión, sistemas de edición de prensa, controles industriales, etc.

Privilegios
Derechos que tiene un usuario sobre un determinado archivo o conjunto de archivos. Estos derechos son: lectura, escritura, modificación y ejecución.

Sector
Área de memoria, típicamente de 512 octetos o bytes que constituye usualmente el conjunto unitario de memoria que administra un sistema operativo, tanto en la memoria principal o memoria RAM, como en las unidades de almacenamiento magnético (discos).

Sector Malo
Sector dañado. Segmento de almacenamiento en disco que no puede ser leído ni grabado, debido a un problema físico en el disco. Los sectores dañados en los discos duros son marcados por el sistema operativo y luego ignorados. Si se graban datos en un sector que luego se daña, para recuperarlos debe utilizarse un software especial de recuperación de archivos, y algunas veces un hardware especial.

Sistema de archivos de Red CODA
Coda es un sistema de archivos distribuido desarrollado como un proyecto de investigación en la Universidad Carnegie Mellon desde 1987 bajo la dirección de M. Satyanarayanan. Se deriva directamente de una antigua versión de AFS (AFS-2) y ofrece muchas características similares. El sistema de archivos InterMezzo está, a su vez, inspirado en Coda. Coda todavía está en desarrollo, aunque el interés se ha desplazado desde la investigación hacia la creación de un producto robusto para uso comercial.

Sistema de archivos de Red NFS
Un Sistema de archivos de red (NFS) permite a los hosts remotos montar sistemas de archivos sobre la red e interactuar con esos sistemas de archivos como si estuvieran montados localmente. Esto permite a los administradores de sistemas, consolidar los recursos en servidores centralizados en la red. Hay dos versiones de NFS actualmente en uso. La versión 2 de NFS (NFSv2), que tiene varios años, es ampliamente soportada por muchos sistemas operativos. La versión 3 de NFS (NFSv3) tiene mejores características, incluyendo manejo de archivos de tamaño variable y facilidades de informes de errores mejoradas, pero no es completamente compatible con los clientes NFSv2. Red Hat Enterprise Linux soporta clientes tanto NFSv2 como NFSv3, y cuando monta un sistema de archivos a través de NFS, Red Hat Enterprise Linux usa NFSv3.

Sistema de archivos de Red NFSV4
Las características más sobresalientes de la versión cuatro de NFS son: el empleo de un

único puerto de red, el 2049 usando TCP, por lo que facilita mucho la configuración del cortafuegos; portmap ya no es necesario; se refuerza la seguridad mediante mecanismos como kerberos, aunque se puede seguir empleando la seguridad básica del sistema; la exportación de directorios está basada en un sistema de archivos virtual, al estilo de los que emplean los servidores de páginas Web ó los servidores de archivos FTP, lo que facilita su administración y uso.

Sistema de archivos FAT
El sistema de archivos FAT (“File Allocation Table”) fue una versión de sistema de archivos que permitía grabar archivos en el disco flexible utilizando el método de organización constituido por una tabla de distribución de archivos. Fue el sistema de archivos con el que operó el sistema operativo DOS con el cual inicialmente trabajaron las microcomputadoras personales PC.

Sistema de archivos FAT 16
En 1987 apareció lo que hoy se conoce como el formato FAT16. El tamaño de la partición manejada por el sistema estaba limitado por el número de sectores por clúster, que era de 8 bits. Esto obligaba a usar clústeres de 32 Kbytes con los usuales 512 bytes por sector. Así que el límite definitivo de FAT16 se situó en los 2 gigabytes.

Esta mejora estuvo disponible en 1988 con el sistema operativo MS-DOS 4.0. Más tarde, Windows NT aumentó el tamaño máximo del clúster a 64 kilobytes gracias al "truco" de considerar el número de clusters como un entero sin signo. No obstante, el formato resultante no era compatible con otras implementaciones de la época, y además, generaba mucha fragmentación interna (se ocupaban clústeres enteros aunque solamente se precisaran unos pocos octetos). Windows 98 fue compatible con esta extensión en lo referente a lectura y escritura. Sin embargo, sus manejadores de disco no eran capaces de trabajar con ella.

Sistema de archivos FAT 32
Con el lanzamiento de Windows 95 OSR2 se introduce el sistema de archivos FAT32, para solucionar en buena parte las deficiencias que presentaba FAT16; FAT32 admite un tamaño de clúster mínimo predeterminado de 4 KB y es compatible con los discos duros de más de 2GB de tamaño.

Sistema de archivos HFS (Hierarchical File System)
Este sistema de archivos fue el reemplazo del MFS de MacOS. HFS tenía soporte para discos MFS. HFS tenía una estructura de directorios jerárquica, que permitía la creación de subdirectorios anidados.

Sistema de archivos HPFS (High Performance File System)
HPFS es un sistema de archivos desarrollado conjuntamente por Microsoft e IBM para el OS/2 1.2. Se desarrolló, como muchos otros sistemas de archivos, para solventar problemas que representaba FAT, aunque HPFS comenzó sin basarse en FAT y con la intención aprovechar los ambientes multitarea.

Sistema de archivos MFS (Macintosh File System)
Es un formato de volumen (o sistema de archivos) creado por Apple Computer para almacenar archivos en disquetes de 400K. MFS fue introducido con el Macintosh 128K en enero de 1984. MFS era notable tanto por introducir los fork de recurso para permitir el almacenamiento de datos estructurados, así como por almacenar metadatos necesitados para el funcionamiento de la interfaz gráfica de usuario de Mac OS. MFS permite que los

nombres de archivo tengan una longitud de hasta 255 caracteres, aunque su buscador no permite que los usuarios creen nombres de más de 63 caracteres de longitud. A MFS se le denomina como sistema de archivo plano porque no admite carpetas.

Sistema de archivos NTFS (New Technology File System)
Fue creado para el sistema operativo de red y multitarea Windows NT de Microsoft. Maneja un sistema de archivos jerárquico similar al del sistema operativo UNIX. A diferencia del separador de nombres de archivos que maneja UNIX /, NT utiliza \, maneja también el concepto de directorio de trabajo vigente, y los nombres de ruta pueden ser relativos o absolutos. A diferencia del sistema de archivos de UNIX que permite montar sistemas de archivos que estén en diferentes máquinas y discos cómo si fueran parte del mismo árbol, lo que oculta la estructura de discos del resto del software, la versión NT 4.0 no contemplaba esta propiedad, por lo que los nombres de archivos absolutos debían comenzar con la letra de la unidad que indicaba de que disco lógico se trataba.

Sistema de archivos UFS (Unix File System)
Es un sistema de archivos utilizado por varios sistemas operativos UNIX y POSIX. Es un derivado del FFS (Berkeley Fast File System), el cual fue desarrollado desde el FS UNIX (este último desarrollado en los Laboratorios Bell). Casi todos los derivativos de BSD incluyendo a FreeBSD, NetBSD, OpenBSD, NeXTStep, y Solaris utilizan una variante de UFS. En Mac OS X está disponible como una alternativa al HFS. En Linux, existe soporte parcial al sistema de archivos UFS, de solo lectura, y utiliza un sistema de archivos nativo de tipo ext3, con un diseño inspirado en UFS.

Sistema de archivos VFAT (Virtual File Allocation Table)
Se deriva del sistema de archivos FAT; fue utilizado por primera vez en Windows 95 funcionando como interfaz entre las aplicaciones y la FAT. Emplea un sistema de 32 bits compatible con el sistema FAT pero tiene diferencias como la longitud de nombres de archivos, que puede ser de hasta 255 caracteres incluyendo espacios.

Sistema Operativo DOS
Es una familia de sistemas operativos para PC. El nombre representa las siglas de “Disk Operating System” (sistema operativo de disco). Fue creado originalmente para computadoras de la familia IBM PC, que utilizaban los procesadores Intel 8086 y 8088, de 16 bits y 8 bits, respectivamente, siendo el primer sistema operativo popular para esta plataforma. Contaba con una interfaz de línea de comandos en modo texto ó alfanumérico, vía su propio intérprete de comandos (command.com). La variante MS-DOS de Microsoft, era suministrada con las computadoras IBM PC, como sistema operativo nativo, hasta bien entrados los años 90 (versión 6.22), frecuentemente junto con una versión de la interfaz gráfica MS Windows de 16 bits, como es la 3.1x.

Sistema Operativo Linux
El núcleo de Linux está vinculado al proyecto GNU iniciado en 1983 por Richard Stallman cuyo propósito era crear un sistema operativo Unix completo integrado enteramente por software libre. Cuando el núcleo de Linux fue liberada en 1991, el proyecto GNU ya había producido varios de los componentes del sistema, incluyendo un intérprete de comandos, una biblioteca C y un compilador. Entonces, el núcleo creado por Linus Torvalds, permitió completar el sistema operativo que GNU se había planteado.

Sistema Operativo MacOS
MacOS, es una abreviatura de Macintosh Operating System (Sistema Operativo de Macintosh), es el nombre del primer sistema operativo de Apple para los equipos de

cómputo Macintosh. El Mac OS original fue el primer sistema operativo exitoso, equipado con una interfaz gráfica de usuario. El grupo de desarrollo de este sistema operativo incluyó a Bill Atkinson, Jef Raskin y Andy Hertzfeld.

El proyecto Macintosh y el proyecto Alto en Xerox PARC tienen una relación sustentada en documentos históricos; y las contribuciones del Sketchpad de Ivan Sutherland y el On-Line System de Doug Engelbart fueron muy significativas.

Apple restó importancia de forma deliberada a la existencia del sistema operativo en los primeros años de Macintosh para ayudar a hacer que la máquina pareciera más agradable al usuario y a distanciarla de otros sistemas como MS-DOS, que eran un desafío técnico. Apple quería que Macintosh fuera visto como un sistema que trabajara nada más con encenderlo.

Sistema Operativo RSX-11
Sistema operativo de tiempo compartido de tercera generación propietario, con el que operaron las computadoras PDP de la extinta compañía Digital Equipment Corporation que tenían procesadores de 16 bits.

Sistema Operativo Unix
Unix (registrado oficialmente como UNIX®) es un sistema operativo portable, multitarea y multiusuario; desarrollado, en principio, en 1969 por un grupo de empleados de los laboratorios Bell de la AT&T, entre los que figuran Ken Thompson, Dennis Ritchie y Douglas McIlroy.

Sistema Operativo Windows NT
Windows NT (Nueva Tecnología) es una familia de sistemas operativos producidos por Microsoft, de la cual la primera versión fue publicada en julio de 1993. Al principio fue diseñado para ser un sistema multiusuario, basado en lenguaje de alto nivel, independiente del procesador, con rasgos comparables a los de Unix. NT fue la primera versión para procesadores de 32 bits de Windows.

Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows Home Server y Windows 7 son versiones basadas en el sistema Windows NT.

Superbloque
El superbloque de un sistema de archivos (FS) es una estructura de metadatos que contiene la información que describe a dicho sistema. Su función principal consiste en indicar al FS el tamaño de las distintas partes del propio sistema de archivos. Los campos de un superbloque, entre otras cosas indican: el tamaño del sistema de archivos, la lista de bloques disponibles, el índice del siguiente bloque libre en la lista de bloques libres, el tamaño de la lista de los i-node, total de i-nodes disponibles, índice del siguiente i-node disponible en la lista de i-nodes disponibles, el indicador que informa si el superbloque ha sido o no modificado, etcétera.

10. Bibliografía

[1] Márquez, Francisco M., UNIX Programación Avanzada, Tercera edición, Alfaomega.
[2] Filesystem Hierarchy Standard, http://proton.pathname.com/fhs/
[3] Giampaolo, Dominic, Practical file systems design with Be file systems, Morgan Kaufmann Publishers 1999.

[4] Card, Rémy; Ts'o, Theodore; Tweedi, Stephene, Design and Implementation of the Second Extended Filesystem, http://web.mit.edu/tytso/www/linux/ext2intro.html
[5] Introducción a los sistemas de archivos FAT, HPFS y NTFS, http://support.microsoft.com/kb/100108/es
[6] From BFS to ZFS: past, present, and future of file systems, Reimer, Jeremy, http://arstechnica.com/hardware/news/2008/03/past-present-future-file-systems.ars
[7] The PDP-11 FAQ, http://www.village.org/pdp11/faq.html
[8] Tanenbaum, Andrew S., Modern Operating Systems, Third Edition, Pearson, Prentice Hall.
[9] OS data, Filesystems, http://www.osdata.com/holistic/connect/filesys.htm
[10] RSX-11, http://www.knowledgerush.com/kr/encyclopedia/RSX-11/
[11] Microsoft Ayuda y Soporte, Descripción del sistema de archivos FAT32, http://support.microsoft.com/kb/154997
[12] Macintosh: File System Specifications and Terms, http://support.apple.com/kb/TA27115
[13] The Mac Observer, Linus Torvalds Brands Mac File System "Utter Crap" http://www.macobserver.com/tmo/article/Linus_Torvalds_Brands_Mac_File_System_Utter_Crap/
[14] Kioskea.net, The file system, http://en.kioskea.net/contents/repar/filesys.php3
[15] Roy, Duncan, HPFS, http://pages.cs.wisc.edu/~bolo/shipyard/hpfs.html, http://cd.textfiles.com/megademo2/INFO/OS2_HPFS.TXT http://seds.org/~spider/spider/os2/HPFS/hpfs_vol.html
[16] Filesystems HOWTO, http://tldp.org/HOWTO/Filesystems-HOWTO.html#toc7
[17] Brecher Steve, HFS File Structure Explained, MACTECH, The journal of Macintosh technology, volumen 1, número 12, http://mactech.com/articles/mactech/Vol.01/01.12/HFSFileStructure/
[18] Inside Macintosh Files, http://developer.apple.com/documentation/mac/Files/Files-2.html
[19] Rivera Santos, Ivonne, Modificaciones al sistema de Archivos ext2 de Linux, CIC, 1999.
[20] Robbins, Kay A., Robbins, Steven, Unix Programación Práctica, Guía de la concurrencia, la comunicación y los multihilos, Prentice Hall.
[21] The MINIX 3 Operating System, http://www.minix3.org/
[22] Microsoft Ayuda y Soporte, El sistema jerárquico de archivos (HFS) de Macintosh, http://support.microsoft.com/kb/130437/es
[23] Coda File System, http://www.coda.cs.cmu.edu/doc/html/manual/x101.html
[24] J. Braam, The CODA Distributed File System, School of Computer Science, Carnegie Mellon University, http://www.coda.cs.cmu.edu/ljpaper/lj.html