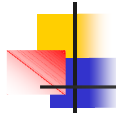


Tema 6

Sistema de Ficheros

Índice

- Ficheros
- Directorios
- Diseño del Sistema de Ficheros



Ficheros

- Almacenan grandes cantidades de información
- Persistencia
- Compartir información
- SISTEMA DE FICHEROS:
 - parte del sistema operativo que se encarga de la gestión de ficheros
- **Abstracción** del sistema que nos permite guardar información sin preocuparnos de los detalles del manejo de los discos o dispositivos de almacenamiento secundario

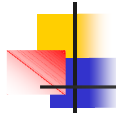
3



Ficheros: Nombres

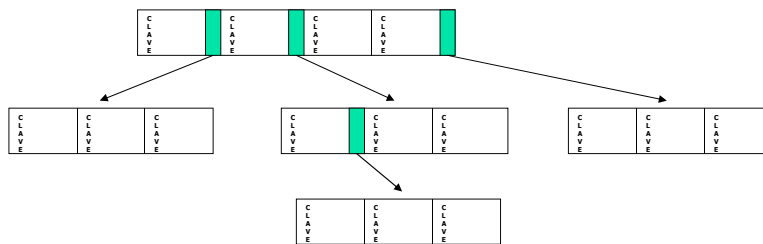
- Asignación de nombre al fichero
 - Un fichero es creado por un proceso que le asigna un nombre
 - Se puede acceder al contenido del fichero usando este nombre
- Reglas para la asignación de nombres
 - Depende del SO
 - Longitud máxima
 - Mayúsculas/Minúsculas
 - Extensión

4

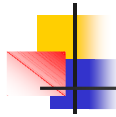


Ficheros: Estructura

- Secuencia de bytes
- Secuencia de registros
- Árbol de bloques de registros



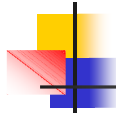
5



Ficheros: Tipos

- Ficheros regulares
 - Información de usuarios (ASCII o binarios)
- Directorios
 - Ficheros del sistema que contienen información necesaria para mantener la estructura del sistema de ficheros
- Ficheros especiales
 - De caracteres y de bloques
 - Para poder hacer referencia a los dispositivos
 - Para modelar el acceso a dispositivos de E/S y discos

6



Ficheros: Acceso

- Acceso secuencial
 - Se lee el fichero en orden, byte a byte o registro a registro
- Acceso directo
 - Se puede leer en cualquier orden
 - Hay llamadas al sistema para posicionarnos en el fichero

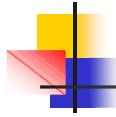
7



Ficheros: Atributos

- Información extra que le asocia el SO
- La lista de atributos depende del SO
 - Tamaño
 - Propietario
 - Protección
 - Flag de sólo lectura
 - Flag de fichero oculto
 - Fecha y hora
 - De creación
 - Último acceso
 - Última modificación

8



Ficheros: Operaciones

- Crear/Borrar
- Abrir/Cerrar
- Leer/Escribir/Añadir
- Posicionarse
- Obtener/Establecer atributos
- Renombrar
- Proyectar ficheros en memoria (ficheros mapeados en memoria)
 - MAP/UNMAP
 - Problemas con el acceso simultáneo:
 - Los cambios podrían no quedar reflejados
 - El fichero podría ser más grande que el espacio de direcciones del proceso completo
 - Se puede proyectar sólo un trozo

9



Directorios

- Permiten organizar los ficheros
- Son en sí mismos ficheros
- Contienen una entrada por fichero perteneciente al directorio
- Toda la información en el mismo directorio o una referencia a una estructura con toda la información
- Cuando un fichero se abre se busca en el directorio y se carga en memoria toda la información

Directorio

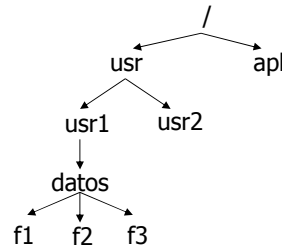
Fichero 1	Atributos + Direcciones De disco
Fichero 2	
Fichero 3	

10



Directorios: Organización

- Se podría tener
 - un único directorio
 - Inmanejable
 - un directorio por usuario
 - Puede haber usuarios con muchos ficheros
- Se agrupan los ficheros formando una estructura jerárquica:
 - Árbol de directorios
- Nombres de ruta de acceso
 - Nombre de ruta absoluta
 - /usr/usr1/datos/f1
 - Nombre de ruta relativa
 - datos/f1
 - 2 entradas especiales:
 - . => Directorio actual
 - .. => Directorio padre



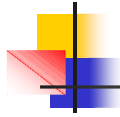
11



Directorios: Operaciones

- Crear: Se ponen las entradas . y ..
- Borrar: Tiene que estar vacío
- Abrir /cerrar
 - Para listar todos los ficheros de un directorio
- Leer
 - El programador no tiene que conocer la estructura interna del directorio
- Renombrar
- Crear/destruir enlaces (entradas de directorios)
 - Para que un fichero pueda aparecer en más de un directorio o varias veces en un mismo directorio
 - Al destruir el último enlace a un fichero se elimina el fichero (ya no se puede llegar a él)

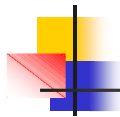
12



Diseño del sistema de ficheros

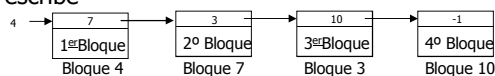
- Desde el punto de vista interno
 - Almacenamiento de ficheros
 - Estructura de los directorios
 - Gestión del espacio en disco
 - Fiabilidad ...

13



Almacenamiento de ficheros

- Un fichero consta de una secuencia de bloques
- El SO debe llevar la cuenta de los bloques asignados a cada fichero
- Bloques contiguos
 - Se almacena el fichero en bloques consecutivos
 - El SO debe saber en qué bloque empieza y cuantos bloques ocupa
 - Problemas:
 - El fichero no puede crecer a menos que se conozca el tamaño máximo
 - Desperdicio del disco por fragmentación
- Listas enlazadas
 - Acceso directo lento
 - Normalmente se lee/escrbe en potencia de dos



14

Almacenamiento de ficheros: Tabla de Asignación de Ficheros (FAT)

- La información del fichero indica cuál es el primer bloque del fichero
- En la posición de la tabla correspondiente al nº del primer bloque se encuentra el segundo y así sucesivamente
- Hay una tabla asociada a cada sistema de ficheros (disco o partición del disco), con una entrada por cada bloque del disco o partición

Se almacena: Primer bloque A partir de la FAT se puede acceder a los bloques de cada fichero

FA	6	FA	6	8	4	2
FB	5	FB	5	9	12	
FC	10	FC	10	3	13	

0	X
1	X
2	EOF
3	13
4	2
5	9
6	8
7	FREE
8	4
9	12
10	3
11	FREE
12	EOF
13	EOF
14	FREE
15	BAD

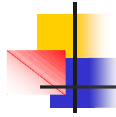
15

FAT 16

- Cada entrada en la tabla FAT es de 16 bits
⇒ nº máximo de bloques = $2^{16} = 64$ Kbloques
- Tam. máx disco = nº máximo de bloques * Tam. bloque
- (clúster = bloque)
- Tam. máx. tabla FAT = $2^{16} * 2B = 2^7 * 2^{10}B = 128KB$

Tamaño de bloque	Tamaño máximo de disco
1 KB	64 MB
2 KB	128 MB
4 KB	256 MB
8 KB	512 MB
16 KB	1 GB
32 KB	2 GB
64 KB	4 GB

16



FAT: Ejemplo 1 (FAT16)

- Disco de 1GB (2^{30} B) con bloques de 32KB (2^{15} B)
 - ¿Cuántos bloques tendrá el disco?
 - $2^{30}/2^{15}=2^{15}$ bloques
 - ¿Cuántos bits se necesitan para direccionar el bloque?
 - Se necesitan 15 bits para indicar el bloque
 - ¿Cuántas entradas en la tabla se necesitan?
 - Se necesitan $2^{15}=32K$ entradas de 16 bits cada una
 - ¿Cuánto ocupa la FAT?
 - La FAT ocupa $2^{15}*2B=2^{16}B=64KB$

17

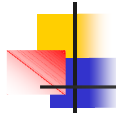


FAT 32

- Cada entrada en la tabla FAT es de 32 bits \Rightarrow n^o máximo de bloques = 2^{32} (en teoría)
- Hay 4 bits que no se usan \Rightarrow n^o máximo de bloques = $2^{28} = 256$ Mbloques
- Tam. máx. disco = n^o máximo de bloques * Tam. bloque
- Tam. máx. tabla FAT = $2^{28}*2^2B=2^{30}B=1GB$
- Por otras limitaciones el tam. máx. de disco es 2 TB
- Por limitación de tam. máx. FAT $\cong 16MB \Rightarrow$ tam. máx. de disco $\cong (2^{24}/2^2)*2^{15}B=128 GB$
- La herramienta format no permite >32GB (Limitaciones del sistema de archivos FAT32)

Tamaño de bloque	Tamaño máximo de disco (4 bits sin usar)
1 KB	256 GB
2 KB	512 GB
4 KB	1 TB
8 KB	2 TB
16 KB	4 TB
32 KB	8 TB
64 KB (NO)	

18



FAT: Ejemplo 2 (FAT 32)

- Disco de 64GB (2^{36}) en bloques de 512B (2^9)
 - ¿Cuántos bloques tiene el disco?
 - $2^{36}/2^9 = 2^{27}$ bloques
 - ¿Cuánto ocupa la FAT?
 - 2^{27} entradas en la FAT con 32 bits cada entrada (FAT32)
 - $2^{27} \times 2^2\text{B} = 2^{29}\text{B} = 512\text{MB}$
 - No permitido FAT > 16MB


19



FAT: Ejemplo 3 (FAT32)

- Disco de 64GB (2^{36}) en bloques de 16KB (2^{14})
 - ¿Cuántos bloques tiene el disco?
 - $2^{36}/2^{14} = 2^{22}$ bloques
 - ¿Cuánto ocupa la FAT?
 - 2^{22} entradas en la FAT con 32 bits cada entrada (FAT32)
 - $2^{22} \times 2^2\text{B} = 2^{24}\text{B} = 16\text{MB}$

20



FAT16 / FAT32

- FAT16
 - Ventaja: Puede residir en memoria y el acceso es rápido
 - Inconveniente: Se desperdicia mucho espacio (bloques grandes)
 - Ejemplo:
 - disco de 2GB (2^{31} B)
 - Nº de bloques máximo en FAT16 = 2^{16}
 - Tamaño mínimo de bloque: 32KB
 - Ya que $2^{31}/2^{16} = 2^{15} = 32\text{KB}$
- FAT32
 - Ventaja: Desperdicia menos espacio (bloques pueden ser más pequeños)
 - Inconveniente: la FAT se hace muy grande y ocupa más memoria (actualmente menos importancia)

21



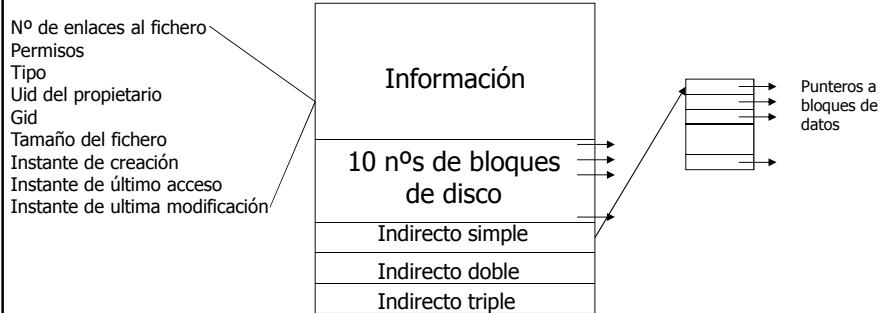
FAT: Problema

- Para buscar la posición 32K de un fichero pueden hacer falta de 1 a 33 accesos a la FAT (0K a 32K) para recorrer la cadena
- Esto se debe a que toda la información de todos los ficheros están mezclados de forma aleatoria en la tabla
- Puede que haga falta la FAT completa aunque se haya abierto un fichero

22

Almacenamiento de ficheros: Nodos-i

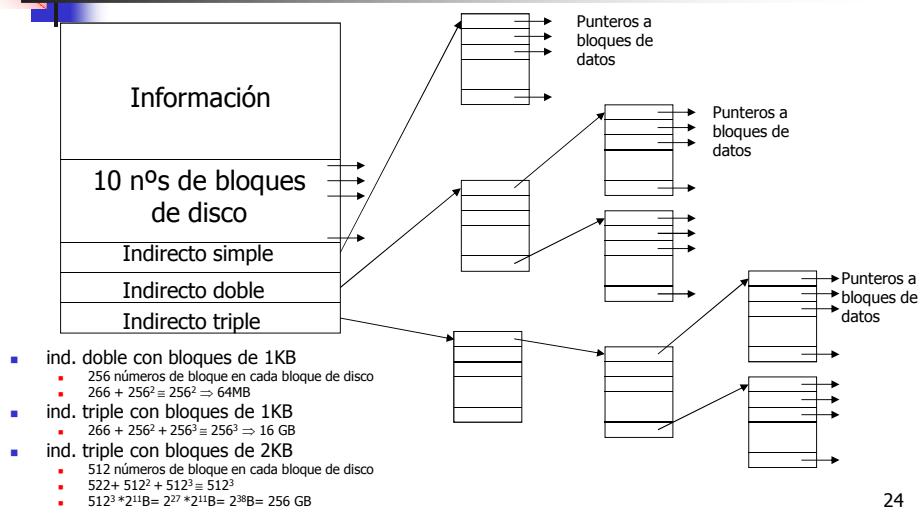
- Se reúne toda la información acerca de un determinado fichero en una estructura



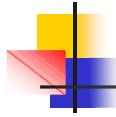
Para bloques de 1KB con direcciones de 32 bits (4B):
En un bloque se pueden almacenar 256 ($2^{10}/2^2$) direcciones de disco
Con esto bastaría para ficheros de 266 bloques (266KB)

23

Nodos-i



24



Nodos-i

- Los bloques indirectos sólo se utilizan cuando hacen falta
- Para conocer la ubicación de cualquier bloque sólo se necesitan 3 referencias a disco
- El nodo-i se lee al abrir el fichero
- Este es el método de almacenamiento de UNIX

25



Estructura de los directorios

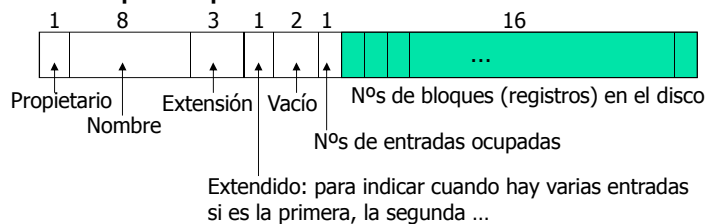
- El SO utiliza el nombre de la ruta de acceso (camino) para localizar la entrada de directorio correspondiente al fichero.
- Una vez localizada la entrada de directorio se puede saber los bloques de disco ocupados por el fichero.
- En algunos sistemas los atributos de los ficheros se almacenan en:
 - la entrada de directorio
 - el nodo-i
- La información almacenada en la entrada de directorio puede ser:
 - la lista de todos los bloques ocupados
 - el nº del primer bloque
 - el nº de nodo-i
- El sistema de directorios sirve para hacer la correspondencia entre el nombre del fichero y los bloques que ocupa en disco

26



Directorios en CP/M

- Sólo tiene un directorio
- Una vez encontrada la entrada de directorio para un determinado fichero se pueden ver los bloques que utiliza

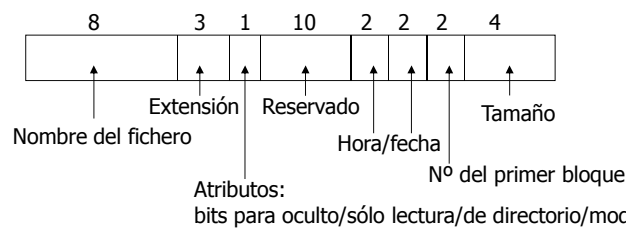


- Si no caben todos los n°s de bloques en una única entrada se utilizan varias

27



Directorios en MS-DOS



- Con el número de bloque se puede buscar en la FAT y encontrar el resto de bloques.

28



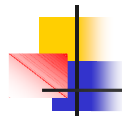
Directorios en UNIX

- Muy simple

2	14
Número de nodo-i	Nombre de fichero

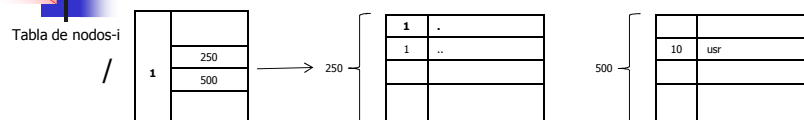
- Toda la información relativa al fichero está en el nodo-i

29



Nodos-i y directorios (I)

/usr/usr1/datos.txt



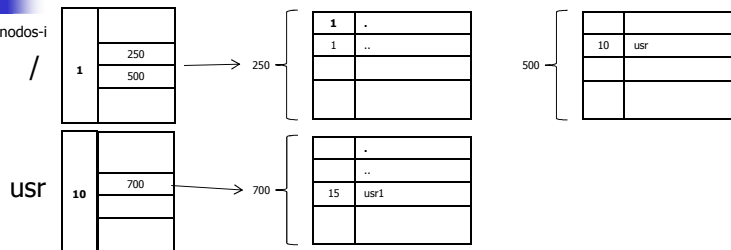
30



Nodos-i y directorios (II)

/usr/usr1/datos.txt

Tabla de nodos-i



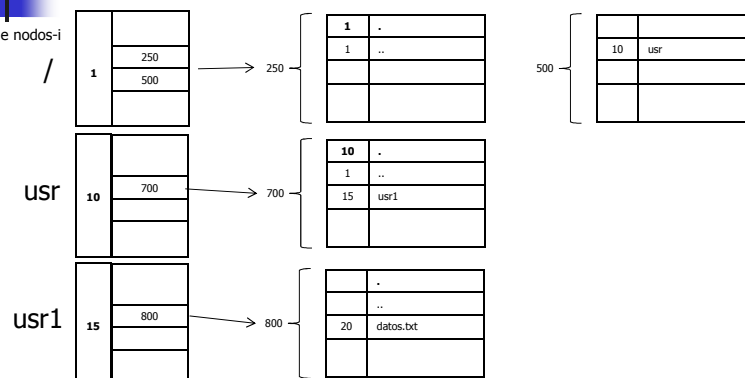
31



Nodos-i y directorios (III)

/usr/usr1/datos.txt

Tabla de nodos-i

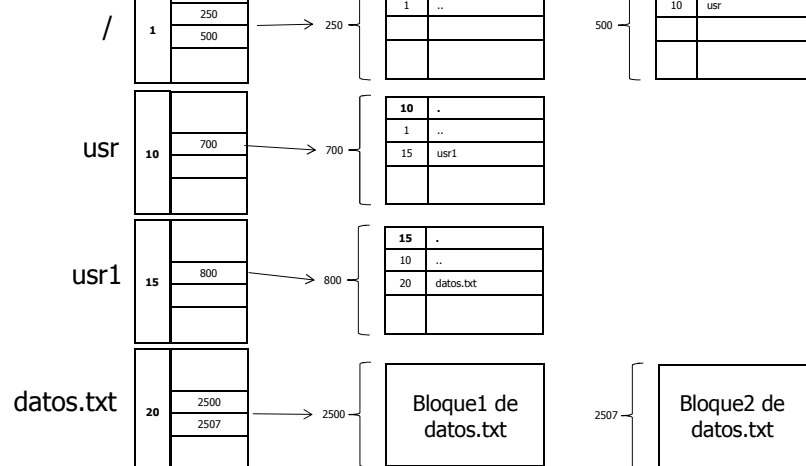


32

Nodos-i y directorios (y IV)

/usr/usr1/datos.txt

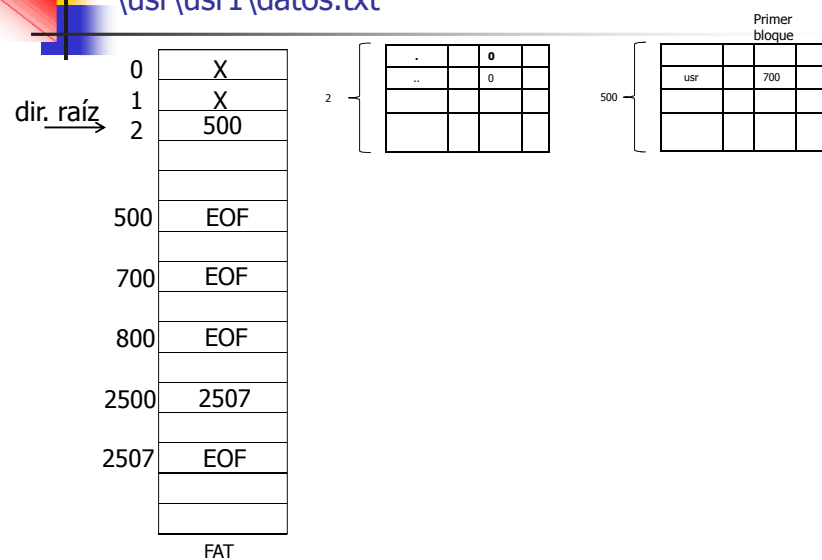
Tabla de nodos-i



33

FAT32 y directorios (I)

\usr\usr1\datos.txt

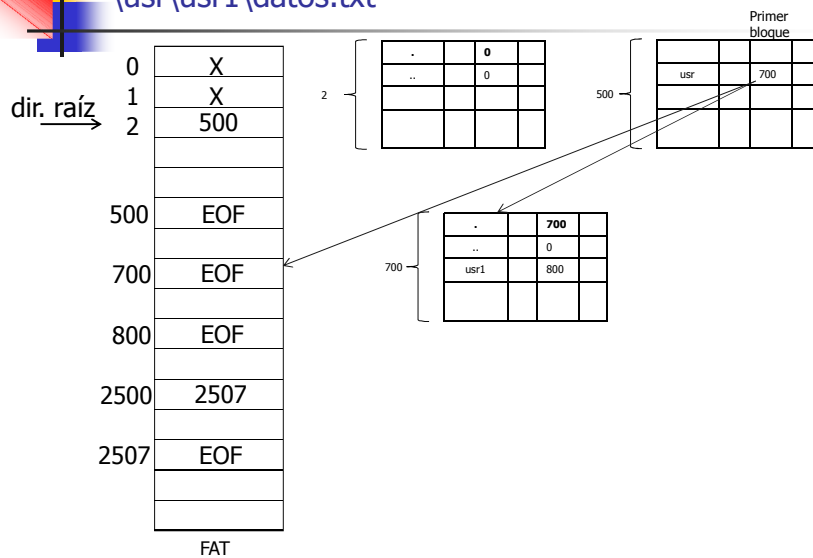


34



FAT32 y directorios (II)

\usr\usr1\datos.txt

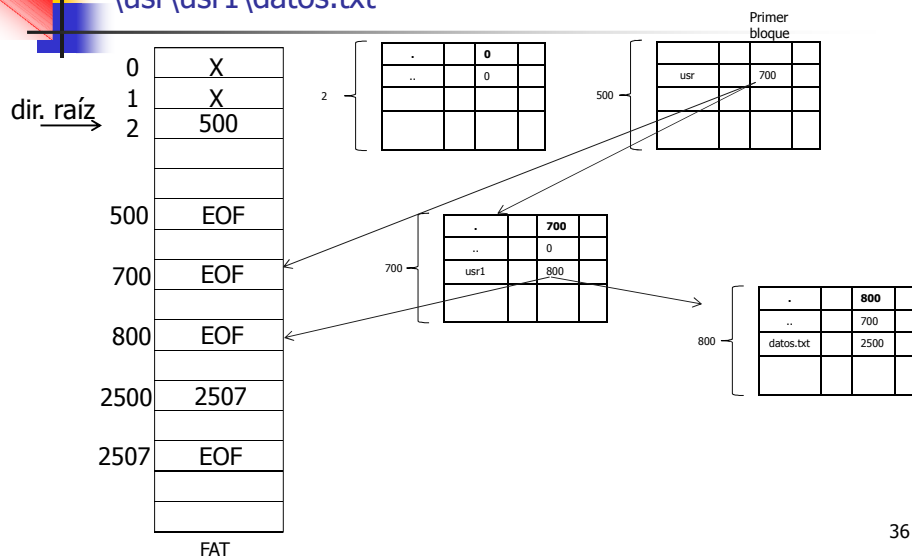


35



FAT32 y directorios (III)

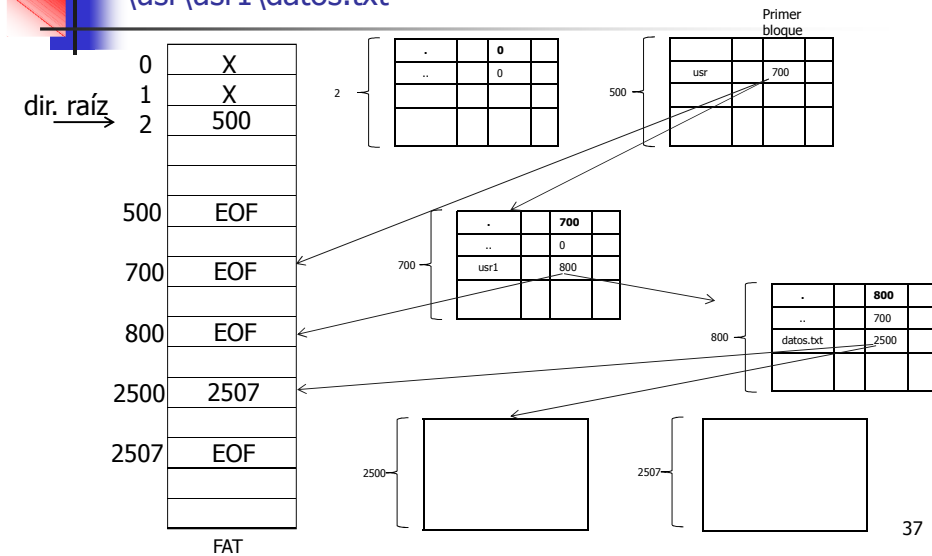
\usr\usr1\datos.txt



36

FAT32 y directorios (IV)

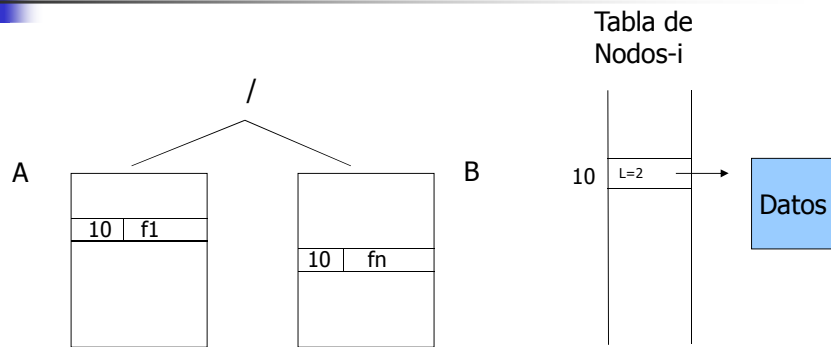
\usr\usr1\datos.txt



Ficheros compartidos

- Para facilitar la compartición de información, un fichero puede aparecer en dos directorios distintos
- Problema:
 - En los sistemas donde la entrada de directorio contiene los nºs de bloques del fichero
 - Si se hace una copia completa de la entrada, si ésta se modifica, no quedan reflejadas las modificaciones
- Solución:
 - En los sistemas con nodos-i: Enlace normal (hard link). Se crean varias entradas con el mismo nº de nodo-i.
 - En cualquier sistema: Enlace simbólico (symbolic link, acceso directo). Se crea un nuevo fichero de tipo enlace que contiene la ruta del fichero con el que se enlaza.

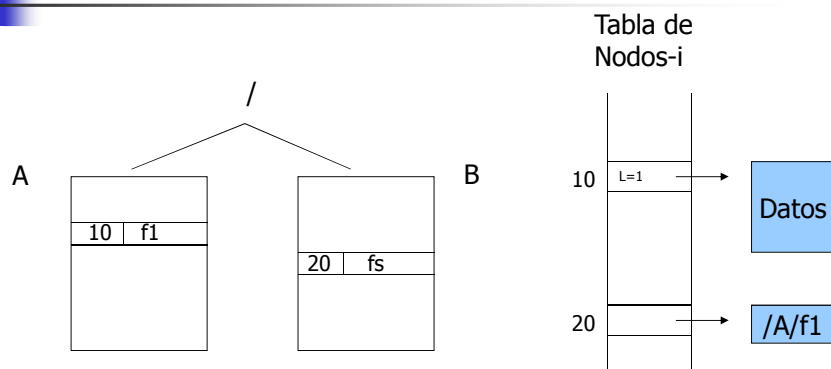
Enlace normal



- En el nodo-i 10, el nº de enlaces en el nodo-i es 2
- Las entradas /A/f1 y /B/fn contienen el mismo nº de nodo-i y son totalmente equivalentes

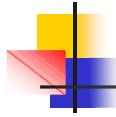
39

Enlace simbólico



- Con el enlace simbólico no se incrementa el nº de enlaces del nodo-i
- En el nodo-i 10, el nº de enlaces vale 1
- Las entradas /A/f1 y /B/fs llevan a los mismos datos, pero NO son totalmente equivalentes
- En el nodo-i 20 hay una indicación de que es de tipo enlace simbólico

40



Enlace normal /enlace simbólico

- En el enlace normal
 - ¿Cuándo se borra el fichero?
 - Cuando se intenta borrar un fichero:
 - Se decrementa el nº de enlaces del nodo-i
 - Se elimina el fichero cuando se llega a 0
 - El propietario sigue consumiendo espacio en disco aunque lo borre, si hay otros enlaces
 - Sólo se pueden enlazar ficheros en el mismo sistema de ficheros
- En el enlace simbólico
 - Si el fichero original se borra, los demás usuarios ya no pueden acceder a él
 - Inconvenientes: Consume tiempo, espacio en la tabla de nodos-i y un bloque por cada fichero enlazado
 - Ventaja: Se pueden enlazar ficheros que estén en cualquier sistema de ficheros

41



Enlaces

1	.
1	..
2	var
5	usuarios

2	.
1	..

5	.
1	..
500	d1
600	d2

600	.
5	..

500	.
5	..
700	problemas
800	ejercicios

700	.
500	..
900	probl

800	.
500	..
900	eje1

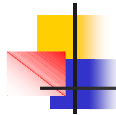
42



Gestión del espacio en disco

- Los bloques ocupados por los ficheros hay que gestionarlos.
- Diversos aspectos:
 - Tamaño de los bloques
 - Gestión de espacio libre
 - Cuota de disco

43



Tamaño de bloque

- Si bloques muy grandes
 - Se desperdicia mucho espacio por fragmentación interna
 - Fragmentación interna: Se refiere a la existencia de bloques no utilizados por completo. En promedio se desperdicia medio bloque por fichero.
 - (Fragmentación externa: Se refiere a la situación en la que los bloques de un mismo fichero no ocupan posiciones contiguas. Las operaciones de L/E tienen que realizar saltos con menor rendimiento del sistema)
- Si bloques muy pequeños
 - Los ficheros tendrán muchos bloques y leerlos será lento
 - Los datos acerca de los bloques ocupan más
- Tamaño de bloque: 512 Bytes, 1KByte, 2KBytes

44

Ejemplo de fragmentación interna (datos)

- Tam. de disco=
 $256\text{GB} = 2^8 * 2^{30}\text{B} = 2^{38}\text{B}$
- Tam. de bloque=
 $32\text{KB} = 2^5 * 2^{10}\text{B} = 2^{15}\text{B}$
- Tam. medio de los
ficheros= 4
bloques= $128\text{KB} = 2^7 * 2^{10}\text{B} = 2^{17}\text{B}$

45

Ejemplo de fragmentación interna (datos y solución)

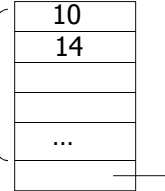
- Tam. de disco=
 $256\text{GB} = 2^8 * 2^{30}\text{B} = 2^{38}\text{B}$
- Tam. de bloque=
 $32\text{KB} = 2^5 * 2^{10}\text{B} = 2^{15}\text{B}$
- Tam. medio de los
ficheros= 4
bloques= $128\text{KB} = 2^7 * 2^{10}\text{B} = 2^{17}\text{B}$
- Nº de ficheros=
 $2^{38}\text{B} / 2^{17}\text{B} = 2^{21}$
- Desperdicia=
 $\frac{1}{2}$ bloque/fichero*
 2^{21} ficheros=
 2^{20} bloques=
 2^{20} bloques*
 $2^{15}\text{B}/\text{bloque} = 2^{35}\text{B} = 2^5 * 2^{30}\text{B} = 32\text{GB}$
- Desperdicia= $\frac{1}{2} * \frac{1}{4} = \frac{1}{8}$ del total

46

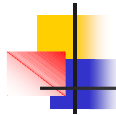


Gestión de espacio libre

- Lista enlazada de bloques
 - Números de bloques libres
 - Número del siguiente bloque de bloques libres
- Mapa de bits
 - Disco con n bloques: mapa de n bits
 - 1: bloque libre
 - 0: bloque asignado
 - Menos espacio (no si el disco está casi lleno)
 - Si el disco está casi lleno y no se puede mantener todo el mapa de bits en memoria, se tendrán que hacer más accesos a disco que con la lista enlazada de bloques



47



Cuotas de disco

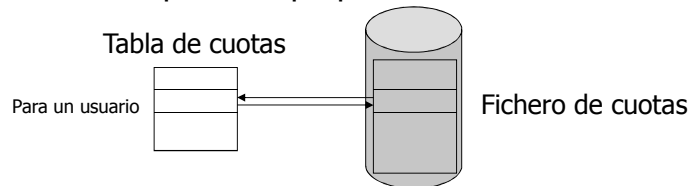
- Para evitar que el disco se llene rápidamente en sistemas multiusuarios
- El administrador establece unas cuotas de disco
 - Número máximo de ficheros
 - Número máximo de bloques
- El S.O se debe encargar de que se cumplan estas cuotas
- Al abrir un fichero: toda la información (atributos y direcciones de disco) se lleva a la [tabla de ficheros abiertos](#) (en memoria)
 - Propietario
- Si el fichero aumenta de tamaño se incrementa la utilización actual del usuario (del propietario del fichero)

48

Cuotas de disco

Tabla de cuotas

- Fichero de cuotas con cuotas máximas para los usuarios
- Tabla de cuotas con las cuotas máximas de aquellos usuarios que sean propietarios de ficheros abiertos

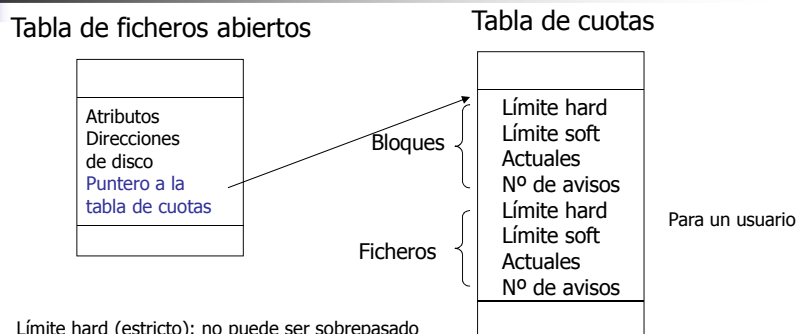


- Cuando un usuario cierra todos los ficheros se lleva al fichero de cuotas la información reflejada en la tabla de cuotas para el usuario propietario

49

Cuotas de disco

Entrada de la tabla de cuotas



- Límite hard (estricto): no puede ser sobrepasado
- Límite soft (flexible): puede ser sobrepasado temporalmente
- Cada vez que un bloque se añade a un fichero se incrementa el nº de bloques actualmente ocupados y se comprueba si se excede el límite
 - Si se excede el límite soft: el sistema avisa
 - Si se excede el límite hard: da un error
- Al entrar un usuario en la máquina se comprueba si se ha sobrepasado el límite soft, en este caso se decrementa el número de avisos, al llegar a 0 no se deja entrar en la máquina
- Variante: no nº de avisos sino tiempo

50



Fiabilidad en sistemas de ficheros

- Gestión de bloques defectuosos
- Copias preventivas
- Coherencia interna del sistema de ficheros

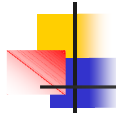
51



Gestión de bloques defectuosos

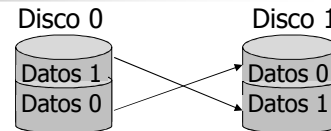
- Hardware:
 - El controlador reserva en disco espacio para almacenar la lista de bloques defectuosos
 - Se sustituye cada bloque defectuoso por un bloque que ha sido reservado para ello
 - El controlador hace la correspondencia entre bloques defectuosos y bloques de reserva
- Software:
 - El sistema de ficheros construye un fichero con todos los bloques defectuosos (ya no estarán disponibles y no se usarán)
 - El fichero de bloques defectuosos no se debe leer ni escribir

52

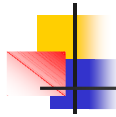


Copias preventivas

- Se copia toda la información
 - Reflejo
 - Se copian los datos
 - Duplicación
 - Hardware duplicado
 - RAID (Redundant array of inexpensive/independent disk)
 - Aumentan la integridad de los datos
 - Mejoran la tolerancia a fallos y errores
 - Mejoran el rendimiento
 - Se pueden tener discos virtuales mucho mayores que los discos comúnmente disponibles
- Copias incrementales
 - Cada cierto tiempo se hace una copia completa (ej: una vez a la semana)
 - Cada día se hace copia de aquellos datos que se han modificado desde la última copia.



53



Coherencia interna del Sistema de Ficheros

- Los sistemas de ficheros leen bloques, los modifican y luego los escriben en disco
- Si el sistema se para y no se copian los bloques modificados, el sistema de ficheros puede quedar en un estado inconsistente (bloques de directorios, de nodos-i, de listas de bloques libres)
- Al arrancar el sistema se hacen algunas comprobaciones de la consistencia del sistema de ficheros:
 - Comprobaciones de bloques
 - Comprobaciones de directorios
 - Otras comprobaciones

54

Comprobaciones de bloques: libre u ocupado

- Comprueba que un bloque aparezca en la lista de libres o como ocupado
- Tabla con una entrada por bloque
 - Contador de veces que aparece en la lista de bloques libres (mira la lista de bloques libres)
 - Contador de veces que aparece en algún fichero (mira la información acerca de los bloques ocupados por los ficheros)
 - Si el sistema de ficheros está en un estado consistente debe aparecer un 1 en el primer contador o en el segundo

55

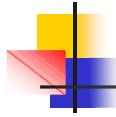
Estado consistente de bloques

Ocupado/Libre

✓	0	1
✓	1	0

- Si aparece algo distinto de 0 - 1 ó de 1 - 0 el estado es inconsistente y hay que reparar el sistema de ficheros
- Una vez reparado, el sistema de ficheros conserva su coherencia aunque probablemente no la información que había en los ficheros

56



Estado no consistente de bloques

Ocupado	Libre	Bien	Se añade a la lista de libres	Se borra de la lista de libres hasta dejarlo sólo una vez	Se borra totalmente de la lista de libres	Se copia en otro/s bloque/s y se asigna a otro/s fichero/s
0	1	x				
1	0	x				
0	0		x			
0	>1			x		
1	>=1				x	
>1	0					x
>1	>=1				x	x

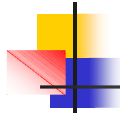
57



Comprobaciones de directorios

- Se comprueba el nº de veces que aparece un nodo-i en una entrada de directorio
- Se crea una tabla de contadores con un contador por nodo-i (tantos elementos como nodos-i haya en el sistema de ficheros)
- Mira recursivamente todos los directorios
- Para cada entrada incrementa el contador de la tabla de la posición del nodo-i correspondiente
- Se compara este contador con el nº de enlaces almacenado en el nodo-i
- En un sistema consistente deben coincidir

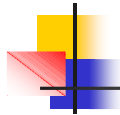
58



Inconsistencias de directorios

- **Nº de enlaces en nodo-i > contador**
 - Problema: si se borran todas las entradas de directorio el nº de enlaces seguirá siendo >0 y no se libera el nodo-i, esto desperdicia espacio en disco
 - Solución: nº de enlaces = contador
- **Nº de enlaces en nodo-i < contador**
 - Problema: si se borran las entradas de directorio llegará un momento en que el nº de enlaces sea 0 pero sigue habiendo entradas de directorio
 - Se borrarán todos los bloques del fichero y se libera el nodo-i correspondiente. Habrá entradas de directorio que indican un nodo-i no coherente
 - Solución: nº de enlaces = contador

59



Otras comprobaciones

- Si nodo-i en el directorio > nodos-i en disco: directorio dañado
- Circunstancias extrañas: protección 007
- Directorios con un nº de entradas muy elevado

60



Otras prestaciones del Sistema de Ficheros

- Protección del sistema contra errores humanos
- Técnicas para aumentar el rendimiento

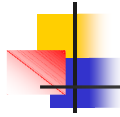
61



Protección del sistema contra errores humanos

- Ejemplo:
 - `rm *.c`
- Cuando los ficheros son borrados se pone un bit a 1 en el directorio o en el nodo-i indicando que el fichero se ha borrado pero no se liberan los bloques pertenecientes al fichero hasta que no haga falta
- Si el usuario se da cuenta del error, se pueden recuperar los ficheros.

62



Técnicas para aumentar el rendimiento

- Memoria caché
- Lectura adelantada
- Reducción del movimiento del brazo del disco

63



Memoria caché: justificación

- El acceso a disco es más lento que acceder a memoria
- Se usa para reducir el nº de accesos a disco
- Son bloques de disco que se mantienen en memoria para mejorar las prestaciones


64



Memoria caché: algoritmo de uso

- Cuando se pide un bloque se mira en la memoria caché
 - Si está no se accede a disco
 - Si no, se accede a disco y se copia el bloque en memoria caché para las siguientes peticiones
 - Si la memoria caché está llena hay que quitar un bloque, si ha sido modificado hay que reescribirlo en disco
 - El bloque que se elimina puede ser elegido con algunos de los algoritmos vistos para sustitución de página en memoria

65



Memoria caché: problema si el sistema cae

- Problema: Si hay una caída del sistema puede haber bloques en memoria caché que hayan sido modificados y no se hayan copiado a disco
 - Si son bloques críticos del sistema (nodos-i, bloques de directorio...), el sistema de ficheros puede quedar en un estado incoherente
- Solución: Los bloques críticos se copian directamente en disco una vez modificados
- De todas formas los bloques de datos tampoco es conveniente tenerlos mucho tiempo en la memoria caché
 - Llamada al sistema SYNC: manda a disco todos los bloques modificados.
 - Caché transparente a escritura: los bloques modificados se escriben inmediatamente en el disco (necesita más entrada/salida y es más lento)

66



Memoria caché: bloques con distinta probabilidad de uso

- Problema: Si se mantienen en memoria caché algunos bloques a los que es poco probable hacer referencia a ellos en breve, se está desperdiciando memoria caché
- Solución: Se dividen los bloques en distintas categorías (nodos-i, bloques indirectos, bloques de datos llenos, bloques de datos parcialmente llenos ...) y los que se van a necesitar no se sustituyen

67



Lectura adelantada de bloques

- Al leer el bloque k, el sistema mira si el bloque k+1 está en la memoria caché
 - Si no está lo lee de forma anticipada
- Mejora el rendimiento en ficheros de acceso secuencial
- Empeora en ficheros de acceso aleatorio
- El sistema puede asociar un bit a cada fichero
 - 1: acceso secuencial (por defecto)
 - 0: acceso aleatorio
 - Se pondría a 0 cuando se hace un desplazamiento en el fichero

68



Reducción del movimiento del brazo del disco

- Para reducir el movimiento del brazo del disco, se asignan bloques cercanos si es probable que se tengan que acceder en secuencia
 - Bloques de un mismo fichero
 - Nodos-i y los bloques del fichero

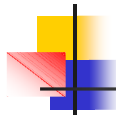
69



Reducción del movimiento : bloques de un mismo fichero

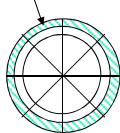
- Con mapas de bits es fácil
- Con listas enlazadas es más complicado
 - Se puede asignar el espacio en disco en unidades de 2 bloques, de esta forma sería más fácil la lectura de un fichero secuencial

70



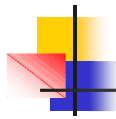
Reducción del movimiento : nodos-i y bloques de un fichero

Nodos-i



- Localizar el nodo-i cerca de los bloques del fichero
 - Normalmente los nodos-i se encuentran al principio del disco
 - Se podrían ubicar en el centro del disco
 - Reduce el movimiento del brazo del disco
 - Se podría dividir el disco en grupos de cilindros, cada uno con sus nodos-i
 - Ubicando los ficheros y sus nodos-i en el mismo grupo de cilindros para hacer el acceso más eficiente

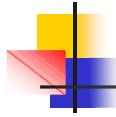
71



Mecanismos de protección

- Mecanismos de protección
 - Técnicas que utilizan los sistemas operativos para proteger los recursos del sistema
- Políticas de protección
 - Deciden qué hacer
 - Se aplican mediante los mecanismos de protección
 - Las políticas que se llevan a cabo se deciden por el administrador del sistema

72



Dominio de protección

- Un sistema de computación se puede modelar como un conjunto de objetos con identificador único y pueden ser:
 - Hardware: memoria, impresoras, discos ...
 - Software: ficheros, procesos, bases de datos ...
- Sobre cada objeto, los procesos pueden llevar a cabo un conjunto finito de operaciones:
 - Ejemplo para un fichero: crear, abrir, leer, escribir....
- Sólo se debe permitir que un proceso acceda a aquellos recursos para los que está autorizado
- Dominio:
 - Es un conjunto de pares <objeto, permisos>
 - Permisos: operaciones permitidas
 - Cada par define para cada objeto las operaciones que se pueden realizar sobre él
 - En cada instante, cada proceso se ejecuta dentro de un cierto dominio de protección

73



Matriz de acceso (o de protección)

- Filas: representan dominios
- Columnas: representan objetos
- Los elementos de la matriz son los derechos de acceso (las operaciones que los procesos en el dominio pueden realizar sobre el objeto)

Objeto Dominio	F1	F2	F3	Impresora
D1			lectura	
D2		Lectura Escritura Ejecución		Impresión
D3		Ejecución		
D4	Lectura Escritura		Lectura Escritura	

74



Formas de implantar la matriz de acceso

- Tabla global
- Listas de control de acceso
- Listas de capacidades

75



Tabla global

- Conjunto de tripletas:
 - $\langle \text{Dominio, Objeto, Conjunto de derechos} \rangle$
- Si se intenta ejecutar la operación M sobre el objeto O_j en el dominio D_i se busca la tripleta $\langle D_i, O_j, C_k \rangle$
 - Si M pertenece a C_k : se realiza la operación
 - Si M no pertenece a C_k : se produce un error
- Problemas:
 - La tabla global es muy grande (necesita mucha E/S)
 - Si todo el mundo tiene derechos de lectura sobre un determinado objeto hay que poner una tripleta para cada dominio con ese objeto

76



Listas de control de acceso (ACL's)

- Representan una columna de la matriz
- La lista para un objeto consiste en pares
 - <dominio, conjunto de derechos>
- Si hay un dominio con el conjunto de derechos vacío para ese objeto, no aparece en la lista
- Se puede definir un valor por omisión, y el conjunto de derechos puede ser restrictivo
- Esta forma se corresponde directamente con las necesidades de los usuarios, cada vez que se crea un objeto, se especifican los derechos de cada dominio
- Pero conociendo el dominio es difícil encontrar el conjunto de derechos

77



Listas de capacidades

- Representan una fila de la matriz
- La lista para un proceso consiste en pares
 - <objeto, conjunto de derechos>
- A los elementos de la lista se les conoce como capacidades

78



Matrices de acceso dinámicas

- Pueden cambiar:
 - Añadir o eliminar objetos
 - Añadir o eliminar dominios
 - Conceder o quitar un derecho en un dominio para un objeto
- La matriz se puede considerar como un objeto más con las operaciones anteriores y dar derecho a los dominios sobre este objeto

79

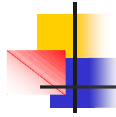


Matrices de acceso dinámicas

- La relación entre proceso y dominio también puede ser dinámica.
 - Un proceso puede cambiar de un dominio a otro
 - Operación de cambio sobre el dominio
 - Se permite cambiar de D_i a D_j si para el dominio D_i existe el derecho de cambio para el objeto D_j

Objeto Dominio	D1	D2	D3	D4
D1		Cambio		
D2			Cambio	Cambio
D3				
D4		Cambio		

80



Alcance de los dominios

- Se puede definir un dominio
 - Para cada usuario
 - Para cada proceso
 - Para cada procedimiento

81

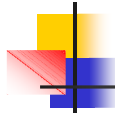


En UNIX

- La protección se hace básicamente en el sistema de ficheros
- Un dominio está asociado a un usuario y a un grupo
- Los objetos son los ficheros
 - Pero como los dispositivos están representados por ficheros especiales de dispositivos, se controlan implícitamente los distintos dispositivos
- Se utilizan
 - Listas de control de acceso
 - Listas de capacidades

82

UNIX:

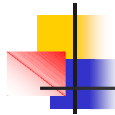


Procesos, ficheros y permisos

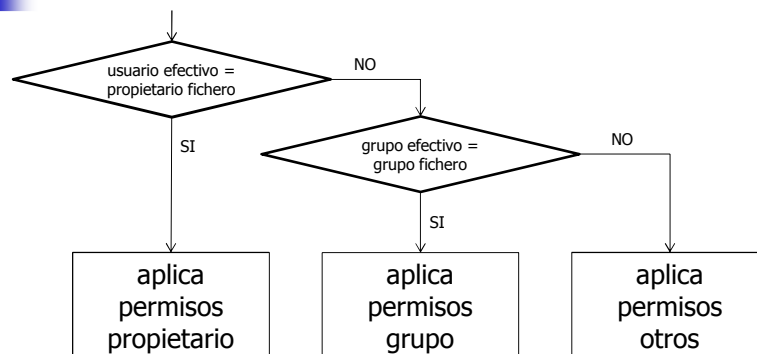
- Cada proceso tiene (heredados del padre al crearse con fork):
 - Usuario real
 - Usuario efectivo (por defecto igual al real)
 - Grupo real
 - Grupo efectivo (por defecto igual al real)
- Cada fichero tiene (heredado del proceso que lo crea):
 - Propietario (usuario efectivo)
 - Grupo (grupo efectivo)
- También tiene permisos (rwx) (lectura, escritura y ejecución) independientes para:
 - Propietario
 - Grupo
 - Otros
- Los permisos se representan con 9 caracteres. Por ejemplo:
 - rwxr_xr__
 - Propietario: rwx
 - Grupo: r_x
 - Otros: r__
- Los permisos se representan con 3 dígitos en octal. Por ejemplo:
 - 754
 - Propietario: 7=111
 - Grupo: 5=101
 - Otros: 4=100

83

UNIX:



Proceso que accede a fichero



84

UNIX:



Cambio de permisos

- Los permisos de un fichero los puede cambiar un proceso cuyo usuario efectivo coincida con el propietario del fichero (y un proceso de root)
- El propietario de un fichero no se puede cambiar (excepto un proceso de root)

85

UNIX:

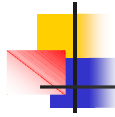


Listas de control de acceso

- Se utiliza una ACL para cada objeto (fichero)
- Con los bits rwx para los permisos:
 - r lectura
 - w escritura
 - x ejecución
- Para cada fichero, los dominios se agrupan en tres tipos de procesos:
 - Usuario efectivo del proceso igual al propietario del fichero
 - Usuario efectivo distinto al propietario, pero grupo efectivo del proceso igual al grupo del fichero
 - Resto de procesos

86

UNIX:



Listas de capacidades

- Cuando un proceso abre un fichero, se busca en la estructura de directorios y se comprueba que tiene acceso
- Si se puede abrir, copia toda la información referente al fichero en memoria en la tabla de ficheros abiertos para el proceso
 - Tipo de acceso que se puede realizar sobre el objeto
- Dominio: proceso
- Objetos: ficheros abiertos del proceso

87

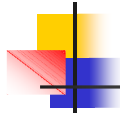
UNIX:



Cambio de dominio

- Es posible cambiar el usuario efectivo (y el grupo efectivo) de un proceso
- Los ficheros, además de los nueve bits de permisos, tienen dos bits adicionales:
 - Bit set-uid: Bit de cambio de usuario efectivo
 - Bit set-gid: Bit de cambio de grupo efectivo
- Si un proceso del usuario A (A es el usuario real y efectivo) ejecuta (exec) el programa del fichero FB (cuyo propietario es el usuario B)
 - Si set-uid no activo:
 - A es el usuario real y efectivo
 - Si set-uid está activo:
 - A es el usuario real y B el usuario efectivo

88



Ejemplo de cambio de dominio

- La orden passwd cambia el password del usuario
 - El fichero passwd contiene los passwords de los usuarios y sólo lo puede modificar el root
 - El programa que se ejecuta con la orden passwd está en un fichero con el bit set-uid activado y cuyo propietario es root
 - Cuando se ejecuta passwd se cambia el usuario efectivo al root y se puede cambiar el fichero passwd

89