Sistemas Operativos UCM 2012/2013

Módulo 2 Gestión de Archivos y Directorios

1. Considerar un sistema donde el espacio libre se especifica en una lista de espacios libres.

a) Suponer que se pierde el puntero a la lista. ¿Puede el sistema reconstruirla?
b) Sugerir un esquema que asegure que el puntero nunca se pierda como resultado de un fallo de memoria.

SOLUCIÓN

a) Habría que recorrer todos los ficheros del sistema y guardar un registro de sus posiciones, para seguidamente asignar los que faltan a la lista enlazada (válido para 1 solo error).
b) Lista doblemente enlazada? Duplicidad? En general, redundancia de información.

2. Calcular el número de accesos a disco necesarios (para el caso peor y para el caso mejor) para leer 20 bloques lógicos consecutivos (no necesariamente los 20 primeros) de un fichero en un sistema con:

a) Asignación contigua
b) Asignación no contigua mediante índice enlazado (*FAT*)
c) Asignación no contigua indexada directa

Nota: Asumir que no hay ningún dato relacionado con el sistema de ficheros en memoria *RAM* y que el fichero se encuentra en el directorio actual.

SOLUCIÓN

a) 20 (datos) + encontrar el fichero en el directorio actual (entre 1 y N accesos dependiendo del tamaño del directorio y la posición del fichero)
b) 20 (datos) + 1-N (directorio actual) + seguir la FAT (entre 1 (mínimo) e Y+20 (maximo))
c) 20 (datos) + 1-N (directorio actual) + 1 (i-nodo) + buscar las tablas de índices adecuadas (hasta 3 niveles)

3. Un dispositivo de memoria flash de 64 MB de capacidad y bloques de 1KB, contiene un sistema de ficheros FAT. ¿Cuántos bytes son necesarios para almacenar la tabla FAT?

a. 64 KB
b. 128 KB
c. 1 MB
d. 512 KB
e. Ninguna de las respuestas anteriores es correcta

¿Es posible realizar enlaces rígidos en un sistema de ficheros tipo FAT? ¿Por qué no se puede establecer enlaces rígidos a ficheros de volúmenes distintos en sistemas de ficheros tipo EXT2?

SOLUCIÓN

Número de bloques de datos direccionables: 64 MB/1KB = 64K bloques
La FAT debe poder contener 64K direcciones
Cada dirección ocupa como mínimo: $\log_2$ 64K bits = 16 bits = 16/8 bytes = 2 bytes
Luego, la FAT ocupa: 64K direcciones * 2 bytes/dirección = 128KB (respuesta b)

Apartado b)

No es posible – porque en FAT cada nombre de fichero dentro de un volumen tiene una entrada de atributos distinta y mantenerlas exactamente iguales representaría un problema de coherencia desorbitado PORQUE los enlaces rígidos están asociados a un inodo de un volumen específico, y los inodos no se pueden compartir entre volúmenes (cada volumen es un "espacio" de inodos independiente.

4. En la siguiente figura se representa una tabla *FAT*. Al borde de sus entradas se ha escrito, como ayuda de referencia, el número correspondiente al bloque en cuestión. También se ha representado la entrada de cierto directorio. Como simplificación del ejemplo, suponemos que en cada entrada del directorio se almacena: Nombre de archivo/directorio, el tipo (*F*=archivo, *D*=directorio), la fecha de creación y el número del bloque inicial.

#Bloque	Índice	#Bloque	Índice
1		10	
2		11	
3	15	12	
4		13	
5		14	
6		15	<EOF>
7		16	
8		17	
9		18	

Nombre	Tipo	Fecha	Nº Bloque
DATOS	F	8-2-05	3

Rellene la figura para representar lo siguiente (**Nota**: supóngase tamaño de bloque 512 Bytes y que siempre se elige el primer bloque vacío como política de asignación):

a) Creación del archivo *DATOS1* con fecha 1-3-90, y tamaño de 10 Bytes.
b) Creación del archivo *DATOS2* con fecha 2-3-90, y tamaño 1200 Bytes.
c) El archivo *DATOS* aumenta de tamaño, necesitando 2 bloques más.
d) Creación del directorio *D*, con fecha 3-3-90, y tamaño 1 bloque.
e) Creación del archivo *CARTAS* con fecha 13-3-90 y tamaño 2 KBytes.

Si usamos un *Mapa de Bits* para la gestión del espacio libre, especifique la sucesión de bits que contendría respecto a los 18 bloques de la tabla anterior.

SOLUCIÓN

#Bloque	Índice	#Bloque	Índice
1	<EOF>	10	11
2	4	11	12
3	15	12	<EOF>
4	5	13	
5	<EOF>	14	
6	7	15	<EOF>/6
7	<EOF>	16	
8	<EOF>	17	
9	10	18	

Nombre	Tipo	Fecha	Nº Bloque
DATOS	F	8-2-05	3
DATOS1	F	1-3-90	1
DATOS2	F	2-3-90	2
D	D	3-3-90	8
CARTAS	F	13-3-90	9

5. Considerar un fichero que consta de 100 bloques. ¿Cuántas operaciones de disco son necesarias para cada una de las tres estrategias de asignación (contigua, enlazada e indexada) al realizar las siguientes operaciones?:

a) Añadir un bloque de información al comienzo
b) Añadirlo a la mitad
c) Añadirlo al final
d) Suprimirlo del principio
e) Suprimirlo de la mitad
f) Suprimirlo del final.

SOLUCIÓN

Añadir → Mirar si hay hueco al principio o final. Si lo hay, marcarlo como ocupado y mover la porción de datos necesaria (modificar la posición inicial del fichero si hace falta). Si no hay hueco, buscar uno lo suficientemente grande y copiar el contenido completo, actualizando la posición inicial del fichero (liberar y reservar huecos). Actualizar siempre el tamaño del fichero y atributos en el directorio.

Eliminar → Mover la porción de datos necesaria (modificar la posición inicial del fichero si hace falta), marcar el espacio como libre. Actualizar siempre el tamaño del fichero y atributos en el directorio.

Añadir → Buscar un hueco en la FAT, enlazarlo en la posición deseada (modificando si es necesario el bloque inicial en el directorio). Actualizar el tamaño y atributos en el directorio.
Eliminar → Marcar el bloque correspondiente como libre, quitarlo de la lista enlazada (modificando si es necesario el bloque inicial en el directorio). Actualizar el tamaño y atributos en el directorio.

Añadir → Buscar un hueco en el mapa de bits y marcarlo como ocupado. Indexar el nuevo hueco en la posición correcta, desplazando los siguientes índices. Actualizar los atributos en el i-nodo.
Eliminar → Buscar un hueco en el mapa de bits y marcarlo como libre. Eliminar el índice y desplazar los siguientes índices. Actualizar los atributos en el i-nodo.

6. Un sistema de ficheros *UNIX* utiliza bloques de 1024 bytes y direcciones de disco de 16 bits. Los *i-nodos* (entradas en una tabla que contiene la información descriptiva de los ficheros) contienen 8 direcciones de disco para bloques de datos, una dirección de bloque índice indirecto simple y una dirección de bloque índice indirecto doble. ¿Cuál es el tamaño máximo de un fichero en este sistema? ¿Y el de la partición?
SOLUCIÓN
El tamaño máximo de un fichero será el número de bloques accesible, directa o indirectamente, desde un i-nodo:
Directamente se puede acceder a 8 bloques
Indirectamente se puede acceder a:
Simple: 1024[Bytes]/2[Bytes/Bloque] → 512 [Bloques].
Doble: 512 [Índices/Bloque] • 512 [Bloque/Índice] = 512^2 [Bloques]
Recuento final:
8 + 512 + 512•512 = 262 664 bloques = 268.967.936Bytes = 256'5MB
Pero el tamaño del disco es como máximo 2^{16} = 64 MB!!!!

7. Dar 5 nombres de ruta diferentes para el fichero */etc/passwd*. (Sugerencia: considerar las entradas de directorio **.** y **..**)
SOLUCIÓN
Si consideramos que un fichero en UNIX sólo está representado unívocamente por su número de inodo y no por su nombre simbólico, podría haber muchos nombres distintos que designaran al mismo fichero: todos los que hubieran sido declarados enlaces al mismo inodo mediante la orden.

Pero si se nos pide acceder al mismo nombre simbólico expresando con distintos nombres de rutas una solución podría ser:
/etc/passwd , nombre de ruta absoluto
passwd , suponiendo que el directorio actual es /etc
./passwd
./../passwd
../etc/passwd , suponiendo que el directorio actual es /usr

8. ¿Cuántos accesos a disco son necesarios para abrir el fichero *games/chess* en *UNIX*?
SOLUCIÓN
Para abrir al fichero games/chess son necesarias como mínimo 4 referencias a disco (entendiendo que algunas de las referencias a disco pueden ser evitadas porque la información precisada ya se encuentra en la cache de buffers):
1. Acceso al directorio actual para buscar el nombre games
2. Acceso al inodo correspondiente al directorio games
3. Acceso al directorio games para buscar el nombre chess
4. Acceso al inodo correspondiente al fichero chess
5. (No es preciso acceder a los datos del fichero chess durante la apertura)

9. Un programa *UNIX* crea un fichero e inmediatamente se posiciona (con *lseek*) en el byte 55 millones. Luego escribe un byte. ¿Cuántos bloques de disco ocupa ahora el fichero (incluyendo bloques indirectos)? Asúmase que los i-nodos contienen 10 índices directos, 1 índice indirecto simple, 1 índice indirecto doble y

1 índice indirecto triple, los bloques tienen un tamaño de 2 Kbytes y el tamaño de los índices (direcciones de bloques de disco) es de 32 bits. ¿Qué sucesión de índices lógicos nos lleva al byte posicionado por *lseek*?

SOLUCIÓN

Cada bloque contiene 2048 bytes y que cada índice ocupa 4 bytes → un bloque de índices apunta a 512 bloques.

Con los 10 índices directos se accede a los primeros 10 x 2048 = 20.480 bytes
Con el índice indirecto simple se accede a los siguientes (2048 / 4) x 2048 = 1.048.576 bytes
Con el índice indirecto doble se accede a los siguientes $(2048 / 4)^2$ x 2048 = 536.870.912 bytes

Luego el bloque de datos que contiene el byte 55 millones se alcanza mediante el índice indirecto doble, y en consecuencia el número de bloques que ocupa un fichero que sólo tiene escrito el byte 55 millones es de 3:

1. El bloque de índices al que apunta el índice indirecto doble (índices indirectos). (Tiene escrito el índice 51) → floor((55e6-10*2048-1024^2) / 1024^2) = 51
2. El bloque de índices al que apunta uno de los índices del bloque anterior (índices directos). (Tiene escrito el índice 221) floor((55e6-10*2048-1024^2-51*1024^2) / 2048) = 221
3. El bloque de datos que contiene el byte 55 millones. (Tiene escrito el byte 960)
4. 55e6-10*2048-1024^2-51*1024^2-221*2048 = 960

La cuenta total es: 20.480 + 1.048.576 + 51 x 1.048576 + 221 x 2048 + 960 = 55.000.000

10. Juan crea un fichero de nombre *JFichero*. María crea un enlace físico a *JFichero* y le da el nombre *MFichero*. Juan elimina *JFichero*. Luego Juan crea un nuevo fichero y también le llama *JFichero*. ¿Cuántos ficheros diferentes existen después de las acciones anteriores? ¿Sería diferente la respuesta si el enlace que ha creado María fuese un enlace simbólico de nombre *MFichero* que apuntara a *JFichero*?

SOLUCIÓN

a. Dos ficheros diferentes
b. Un fichero y un enlace simbólico

11. Sugerir una razón por la cuál alguien pudiera desear construir un directorio sobre los cuales los demás usuarios tuvieran permiso de ejecución pero no de lectura, en un sistema de ficheros de tipo *UNIX*.

SOLUCIÓN

The execute permission, which grants the ability to execute a file. This permission must be set for executable binaries or shell scripts in order to allow the operating system to run them. When set for a directory, this permission grants the ability to traverse its tree in order to access files or subdirectories, but not see the content of files inside the directory (unless read is set).
Por ejemplo, en /home para poder entrar en tu directorio pero no saber el nombre de los demás

12. Dos estudiantes de informática, *Estudiante1* y *Estudiante2*, sostienen una discusión respecto a los *i-nodos*. *Estudiante1* argumenta que, dado que las memorias son cada vez más grandes y baratas, cuando se abre un fichero es más sencillo y rápido obtener una nueva copia del *i-nodo* para llevarla a la tabla de *i-nodos* en memoria, que buscar en la tabla entera para comprobar si ya está allí. *Estudiante2* discrepa. ¿Quién tiene razón?

SOLUCIÓN

El estudiante 2. Sólo se debe de tener una copia válida del inodo, ya que contiene absolutamente toda la información del fichero (menos los datos en si), que deberá de ser actualizada para hacerla visible a otras tareas por consistencia.

13. Un sistema de ficheros basado en i-nodos y mapa de bits contiene la siguiente información:

Mapa de bits: 1 0 0 1 0 1 1 1 0 0 0 0 1 0 0 1 0 0 0

i-nodo 2

Tamaño	1
#Enlaces	NA
Tipo F/D	D
Directo	3

i-nodo 3

Tamaño	2
#Enlaces	1
Tipo F/D	F
Directo	6

i-nodo 4

Tamaño	1
#Enlaces	
Tipo F/D	F
Directo	12

i-nodo 5

Tamaño	
#Enlaces	NA
Tipo F/D	D
Directo	0

i-nodo 9

Tamaño	1
#Enlaces	NA
Tipo F/D	D
Directo	

Indirecto	Null

Indirecto	7

Indirecto	Null

Indirecto	Null

Indirecto	Null

Bloque 0

.	5
..	2
C	9
D	4

Bloque 3

.	2
..	
A	3
B	5
E	4

Bloque 5

.	9
..	5

Bloque 6

Datos sin formato

Bloque 7

Bloque 12

Bloque 15

Datos sin formato

a) Rellene los huecos para que el sistema sea consistente. Asuma para ello que el tamaño se expresa en bloques.

b) Dibuje el árbol del directorio empleando óvalos para los directorios, rectángulos para los ficheros y triángulos para los datos

SOLUCIÓN

Apartado a)

Mapa de bits: 1 0 0 1 0 1 1 1 0 0 0 0 1 0 0 1 0 0 …… 0

i-nodo 2

Tamaño	1
#Enlaces	NA
Tipo F/D	D
Directo	3
Indirecto	Null

i-nodo 3

Tamaño	2
#Enlaces	1
Tipo F/D	F
Directo	6
Indirecto	7

i-nodo 4

Tamaño	1
#Enlaces	2
Tipo F/D	F
Directo	12
Indirecto	Null

i-nodo 5

Tamaño	1
#Enlaces	NA
Tipo F/D	D
Directo	0
Indirecto	Null

i-nodo 9

Tamaño	1
#Enlaces	NA
Tipo F/D	D
Directo	5
Indirecto	Null

Bloque 0

.	5
..	2
C	9
D	4

Bloque 3

.	2
..	2
A	3
B	5
E	4

Bloque 5

.	9
..	5

Bloque 6

Datos sin formato

Bloque 7

15

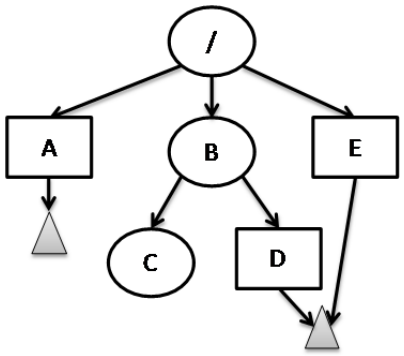
Bloque 12

Datos sin formato

Bloque 15

Datos sin formato

Apartado b)



14. Un sistema de ficheros UNIX utiliza bloques de 512 bytes y direcciones de disco de 16 bits. Los **nodos-i** contienen 10 direcciones de disco para bloques de datos, una dirección de bloque índice indirecto simple y una dirección de bloque índice indirecto doble. Conteste de manera razonada a las siguientes cuestiones:

a) ¿Cuál es el tamaño máximo de un fichero en este sistema?

b) Un programa UNIX crea un fichero en este sistema e inmediatamente después escribe un byte de datos en la posición 1.000 y otro en la posición 10.000. ¿Cuántos bloques de datos ocupa este nuevo fichero en disco?

SOLUCIÓN

Bloques 512B, direcciones 2B -> 252 direcciones por bloque

a) Tam. máximo:

Por organización: 10+256+(256)^2 -> 10+2^8+2^16 Bloques ~ 32’13MB

Por tamaño de índice: 2^16Bloques -> **32MB**

b) Pos 1000 -> 1000/512=1'95 -> está en el bloque directo 1 (empezando en 0)

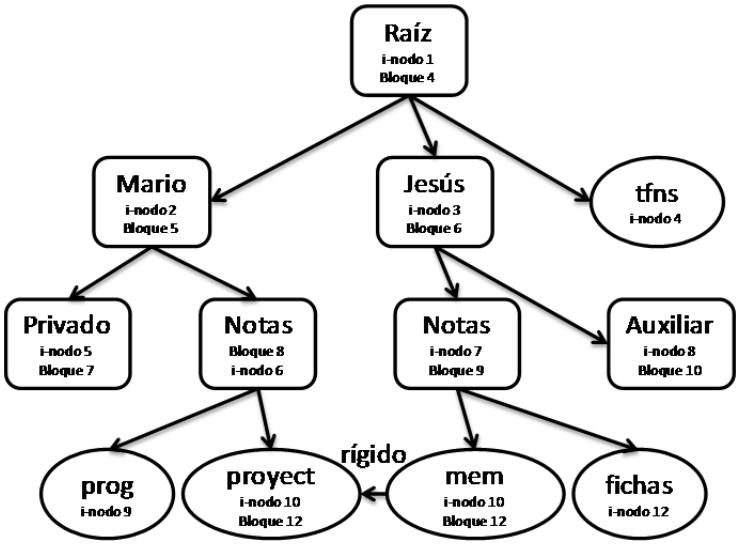
Pos 10000 -> 10000/512=19'5 -> está en el bloque directo simple 9

15. El sistema de ficheros de un SO diseñado a partir de UNIX utiliza bloques de disco de 1024 bytes de capacidad. Para el direccionamiento de estos bloques se utilizan punteros de 16 bits. Para indicar el tamaño del fichero y el desplazamiento (*offset*) de la posición en bytes en las operaciones *read* y *write*, se utilizan números de 64 bits. Cada i-nodo tiene 8 punteros de direccionamiento directo, 1 puntero indirecto simple y 1 puntero indirecto doble.

a) ¿Cuál será el tamaño máximo de un fichero suponiendo despreciable el espacio ocupado por el superbloque y la tabla de i-nodos?

b) Si se modifica el tamaño de puntero pasándolo a 32 bits, ¿cuál será el nuevo tamaño máximo?

c) Dada la siguiente estructura de directorio, en el que *Jesús* comparte el fichero *proyect* de *Mario* mediante un enlace rígido de nombre *mem*, indicar los contenidos de los directorios e *i-nodos* que se encontrará el sistema (y el orden en que se los encontrará) al hacer la búsqueda del fichero *memoria* desde el directorio raíz.



Nota: los directorios se muestran como rectángulos y los ficheros como óvalos.

SOLUCIÓN

a) Tamaño

por offset : 2^{64} bytes = 2^{54} bloques →16Peta Bloques

por organización: 8 + 1024/2 + $(1024/2)^2$ = 262644 bloques → ~256K5 Bloques

por longitud de puntero: 2^{16} bloques → 64K Bloques

Resultado: 2^{16} bloques → ~64MB

b) Tamaño

por offset : 2^{64} bytes = 2^{54} bloques

por organización: 8 + 1024/4 + $(1024/4)^2$ = 65800 bloques → ~64K256 Bloques

por longitud de puntero: 2^{32} bloques

Resultado: 65800 bloques→ ~64MB256

Sólo cálculo a través de i-nodo: 0.5

c)

Inodo1 (raíz)	Bloque 4:	
	"."	1
	" .."	1
	"mario"	2
	"jesus"	3
	"tfno"	4

Inodo 3 (jesus) Bloque 6
“.” 3
“ ..” 1
“notas” 7
“auxiliar” 8
Inodo7 (notas) Bloque 9
“.” 7
“ ..” 3
“memoria” 10
“fichas” 12
Inodo 10 (memoria) Bloque 12 -- datos del fichero memoria

Sin “.” y “..” -0.25
Sin contenido de bloques -0.25

16. Dado un sistema de ficheros tipo UNIX, en el que los directorios son relativamente pequeños (caben en un bloque de disco) y únicamente tiene una partición; razona qué operaciones de disco se necesitan para abrir el archivo “`/usr/curso3/SO/alumnos.txt`”. Suponer que el i-nodo del directorio raíz está ya en memoria y que no se ha cargado en memoria anteriormente ningún otro elemento de la ruta.

SOLUCIÓN

1. Lectura del bloque correspondiente al directorio raíz para encontrar la entrada usr y su número de i-nodo.
2. Lectura del i-nodo correspondiente a la entrada usr para encontrar el bloque correspondiente al directorio usr.
3. Lectura del bloque correspondiente al directorio usr para encontrar la entrada curso3 y su número de i-nodo.
4. Lectura del i-nodo correspondiente a la entrada curso3 para encontrar el bloque correspondiente al directorio curso3.
5. Lectura del bloque correspondiente al directorio curso3 para encontrar la entrada SO y su número de i-nodo.
6. Lectura del i-nodo correspondiente a la entrada SO para encontrar el bloque correspondiente al directorio SO.
7. Lectura del bloque correspondiente al directorio SO para encontrar la entrada alumnos.txt y su número de i-nodo.
8. Lectura del i-nodo correspondiente a la entrada alumnos.txt