SO-Relacion-Tema-4.pdf

Frankie2

Sistemas Operativos

2º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Universidad de Granada

WUOLAH

El más PRO del lugar puedes ser Tú.

¿Quieres eliminar toda la publi de tus apuntes?

¡Fuera Publi!
Concéntrate al máximo

Apuntes a full.
Sin publi y sin gastar coins

Para los amantes de la inmediatez, para los que no desperdician ni un solo segundo de su tiempo o para los que dejan todo para el último día.

Quiero ser PRO

4,95 / mes

1. Sea un Sistema Operativo que sólo soporta un directorio (es decir, todos los archivos existentes estarán al mismo nivel), pero permite que los nombres de archivo sean de longitud variable. Apoyándonos únicamente en los servicios proporcionados por este Sistema Operativo, deseamos construir una "utilidad" que "simule" un sistema jerárquico de archivos. ¿Es esto posible? ¿Cómo?

Como solo nos podemos apoyar en el nombre del archivo, quizás la solución más simple sea indicar con una serie de caracteres al principio del nombre del archivo al usuario al que pertenece, a continuación usar un delimitador para indicar que ha terminado el identificador de usuario y por último, el nombre del fichero y su extensión. Algo así como:

u1#archivoUsuario1.txt
u2#archivoUsuario2.txt
u2#otroArchivo.c

2. En un entorno multiusuario, cada usuario tiene un directorio inicial al entrar en el sistema a partir del cual puede crear archivos y subdirectorios. Surge, entonces, la necesidad de limitar el tamaño de este directorio para impedir que el usuario consuma un espacio de disco excesivo. ¿De qué forma el Sistema Operativo podría implementar la limitación de tamaño de un directorio?

Podemos implementarlo con un sistema de cuotas de usuario. Así, podemos establecer un límite de espacio que puede ocupar un directorio concreto, y de la misma forma podemos establecer límites distintos para usuarios distintos.

Este límite se almacenaría en el i-nodo del directorio y cada vez que se modifica su contenido tendríamos que comprobar que no sobrepasa ese límite.

3. 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.

Nombre	Tipo	Fecha	NºBloque
DATOS	F	8-2-90	3
DATOS1	F	1-3-90	1
DATOS2	F	2-3-90	2
D	D	3-3-90	6
CARTAS	F	13-3-90	7

1	* (DATOS1)	**10**	*****
2	4 (DATOS2)	**11**	
3	15 (DATOS)	**12**	
4	5	**13**	
5	*	**14**	
6	* (D)	**15**	16
7	8 (CARTAS)	**16**	17
8	9	**17**	*
9	10	**18**	

Tenga en cuenta que:

- el tamaño de bloque es de 512 bytes
- el asterisco indica ultimo bloque
- todo lo que está en blanco en la figura está libre.

Rellene la figura para representar lo siguiente:

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.**

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

Sabiendo que un mapa de bits contiene un bit por bloque de datos y por inodo respectivamente para indicar si están libres o no.

Por lo cual, el resultado sería el siguiente:

Nº Bloque	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Bits	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	1

5. Si se pierde el primer puntero de la lista de espacio libre, ¿podría el Sistema Operativo reconstruirla? ¿Cómo?

Sí podría. Habría que recorrer todos los directorios y leer todos los i-nodos de los directorios y de los archivos. Es un proceso costoso, pero una vez acabes, obtendrás la lista de bloques libres y así podrás reconstruir la lista.

6. El espacio libre en disco puede ser implementado usando una lista encadenada con agrupación o un mapa de bits. La dirección en disco requiere D bits. Sea un disco con B bloques, en que F están libres. ¿En qué condición la lista usa menos espacio que el mapa de bits?

Tamaño lista enlazada = F*D
Tamaño bitmap = B

Por tanto la lista enlazada ocupa menos espacio que el mapa de bits si se cumple la siguiente condición: B > F*D lógicamente si F son bloques libres del disco B >= F y B > 0. El bitmap nunca cambiará su tamaño mientras que la lista enlazada crecerá conforme haya más bloques libres.

En el caso de de que todos los bloques estén libres (F = B) la lista enlazada tendrá un tamaño de B*D, por otra parte en el caso de que estén todos los bloques ocupados la lista no ocupará nada (F = 0) ya que 0*D = 0. Por otra parte el bitmap en ambas situaciones ocupará el mismo tamaño B.

7. Entre los posibles atributos de un archivo, existe un bit que marca un archivo como temporal y por lo tanto está sujeto a destrucción automática cuando el proceso acaba ¿Cuál es la razón de esto? Después de todo un proceso siempre puede destruir sus archivos, si así lo decide.

Los archivos temporales son archivos que pueden ser creados por un programa cuando no se le puede asignar memoria suficiente. Algunos programas, cuando crean esos archivos temporales, es probable que no los eliminen por dos motivos principales: o bien el programador se ha olvidado añadir el código necesario para eliminar estos archivos, o bien porque el programa crasheó. Por tanto, para que estos archivos no consuman mucha memoria cuando se dejan olvidados, se incluye el uso de este bit para que se eliminen cuando el proceso acabe sea de la forma que sea.

8. Algunos SO proporcionan una llamada al sistema (RENAME) para dar un nombre nuevo a un archivo existente ¿Existe alguna diferencia entre utilizar esta llamada para renombrar un archivo y copiar el archivo a uno nuevo, con el nuevo nombre y destruyendo el antiguo?

La principal diferencia es el coste de la operación. Es mucho más costoso copiar manualmente el archivo a uno nuevo con el nuevo nombre y posteriormente eliminar el antiguo. Adicionalmente, serían ficheros distintos a la vista del sistema operativo, ya que aunque tienen los mismos metadatos son archivos *distintos*.

9. Un i-nodo de UNIX tiene 10 direcciones de disco para los diez primeros bloques de datos, y tres direcciones más para realizar una indexación a uno, dos y tres niveles. Si cada bloque índice tiene 256 direcciones de bloques de disco, cuál es el tamaño del mayor archivo que puede ser manejado, suponiendo que 1 bloque de disco es de 1KByte?

Primeramente, tenemos 10 bloques en el i-nodo que representan las direcciones de disco para los diez primeros bloques de datos, que cada uno ocupa 1KB. A continuación tenemos 3 bloques, donde el primer bloque se utiliza para realizar la indexación a un nivel, con 256 direcciones, por tanto ocupa 256*1KB. El siguiente bloque se utiliza para realizar la indexación a dos niveles, con 256 direcciones junto a las 256 direcciones del nivel anterior, por tanto, ocupa 256*256*1KB. Siguiendo el mismo razonamiento, el bloque para el último nivel de indexación ocupa 256*256*256*1KB.

Sumamos a continuación cada tamaño y obtenemos que:

(10*1KB) + (256*1KB) + (256*256*1KB) + (256*256*256*1KB) = 16.843.018 KB = 16,06GB

El tamaño máximo del archivo que puede ser manejado por el i-nodo es de algo más de 16 GB.

10. Sobre conversión de direcciones lógicas dentro de un archivo a direcciones físicas de disco. Estamos utilizando la estrategia de indexación a tres niveles para asignar espacio en disco. Tenemos que el tamaño de bloque es igual a 512 bytes, y el tamaño de puntero es de 4 bytes. Se recibe la solicitud por parte de un proceso de usuario de leer el carácter número N de determinado archivo. Suponemos que ya hemos leído la entrada del directorio asociada a este archivo, es decir, tenemos en memoria los datos PRIMERBLOQUE y TAMAÑO. Calcule la sucesión de direcciones de bloque que se leen hasta llegar al bloque de datos que posee el citado carácter.

Tamaño de Bloque: 512 Bytes
Tamaño de Puntero: 4 Bytes

512 Bytes / 4 Bytes = 128 direcciones

i = n/128*128*512 -> n' = n%128*128*512
j = n'/128*512 -> n'' = n'%128*512
k = n''/512 -> n''' = n'''%512

11. ¿Qué método de asignación de espacio en un sistema de archivos elegiría para maximizar la eficiencia en términos de velocidad de acceso, uso del espacio de almacenamiento y facilidad de modificación (añadir/borrar /modificar), cuando los datos son:

a) **modificados infrecuentemente, y accedidos frecuentemente de forma aleatoria**
Asignación contigua, debido a que es buena para accesos aleatorios, además de no necesitar que crezca el archivo. A su vez, podemos hacer uso también de la asignación no contigua indexada, debido a que beneficia el acceso aleatorio y no tendría problema en que un fichero no se modificara.

b) **modificados con frecuencia, y accedidos en su totalidad con cierta frecuencia**
Asignación no contigua enlazada, debido a que el archivo puede crecer de forma dinámica mientras haya espacios libres y el acceso secuencial se beneficia con este tipo de asignación.

c) **modificados frecuentemente y accedidos aleatoriamente y frecuentemente**
Asignación no contigua indexada, debido a que no produce fragmentación externa, beneficiando los archivos que se modifican frecuentemente, además de favorecer los accesos directos, por lo que se podrá acceder de forma aleatoria sin ningún problema.

12. ¿Cuál es el tamaño que ocupan todas las entradas de una tabla FAT32 que son necesarias para referenciar los cluster de datos, cuyo tamaño es de 16 KB, de una partición de 20 GB ocupada exclusivamente por la propia FAT32 y dichos cluster de datos?

20 GB * (1024 MB/GB) * (1024 KB/MB) = 20.971.520 KB
20.971.520 KB / 16 KB = 1.310.720 entradas
1.310.720 entradas * 32 bits = 41.943.040 bits
41.943.040 bits / (8 bits * 1024 B) = 5.120 KB
5.120 KB / 1.024 KB = 5 MB -> El tamaño de la tabla FAT32 será de 5MB

13. Cuando en un sistema Unix/Linux se abre el archivo /usr/ast/work/f, se necesitan varios accesos a disco. Calcule el número de accesos a disco requeridos (como máximo) bajo la suposición de que el i-nodo raíz ya se encuentra en memoria y que todos los directorios necesitan como máximo 1 bloque para almacenar los datos de sus archivos.

1. Cargar el bloque de directorio de la raíz
2. Buscamos el i-nodo de /usr
3. Cargamos el i-nodo de /usr y cargamos el bloque de directorio de /usr
4. Localizar y acceder al i-nodo de /usr/ast
5. Cargar el bloque de directorio de /usr/ast
6. Localizar y acceder al i-nodo de /usr/ast/work
7. Cargar el bloque de directorio de /usr/ast/work
8. Localizar y acceder al i-nodo /usr/ast/work/f

14. Suponiendo una ejecución correcta de las siguientes órdenes en el sistema operativo Linux: /home/jgarcia/prog

```
$ ls -i          //lista los archivos y sus números de i-nodos del
                 directorio prog
18020        fich1.c
18071        fich2.c
18001        pract1.c

/home/jgarcia/prog > cd ../tmp
/home/jgarcia/tmp > ln -s ../prog/pract1.c p1.c  //crea enlace simbólico
/home/jgarcia/tmp > ln ../prog/pract1.c p2.c     //crea enlace absoluto
```

Represente gráficamente cómo y dónde quedaría reflejada y almacenada toda la información referente a la creación anterior de un enlace simbólico y absoluto ("hard") a un mismo archivo, pract1.c.

Al crear un enlace simbólico y un enlace duro ocurre lo siguiente, con el enlace simbólico se crea un nodo con un puntero al fichero pract1.c. Mientras que el enlace “hard” copia el fichero pract1.c pues dos ficheros que comparten el mismo i-nodo son copias exactas tanto en contenido, permisos...

Las tablas de direcciones de ambos directorios son:

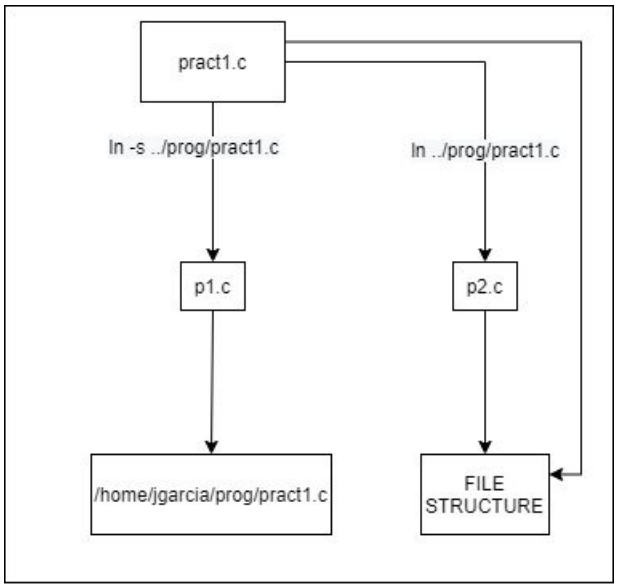
/home/jgarcia/prog

Nombre Archivo	I-Nodo	Contador Enlaces
pract1.c	455454	2

/home/jgarcia/tmp

Nombre Archivo	I-Nodo	Contador de Enlaces
p1.c	775857 (enlace simbólico)	1
p2.c	455454	2

Es necesario especificar en el i-nodo de p1.c que se trata de un enlace simbólico.



15. En un sistema de archivos ext2 (Linux), ¿qué espacio total (en bytes) se requiere para almacenar la información sobre la localización física de un archivo que ocupa 3 Mbytes?. Suponga que el tamaño de un bloque lógico es de 1 Kbytes y se utilizan direcciones de 4 bytes. Justifique la solución detalladamente.

Tamaño de Bloque = 1 Kbytes
Tamaño de Direcciones = 4 bytes

(2^10 bytes/bloque) / (2^2 bytes/direcciones) = 256 direcciones

Tamaño del archivo = 3 Mbytes * (1024 Kbytes/Mbytes) = 3072 Kbytes
Número de bloques = 3072Kbytes / (1Kbytes/bloque) = 3072 bloques

En ext2 poseemos 12 bloques que apuntan a información, el bloque trece tiene indexación a un nivel, el bloque 14 indexación a dos niveles y el bloque 15 a tres niveles.

Los 12 primeros bloques:
3072 Kbytes - (12 * 1 Kbytes) = 3060 Kbytes restantes, no podemos almacenarlo en su totalidad

Bloque 13, indexación a un nivel:
256 direcciones -> 3060 Kbytes - (256 bloques * 1 Kbytes/bloques) = 2804 Kbytes restantes, no podemos almacenarlo en su totalidad

Bloque 14, indexación a dos niveles:
256 dir * 256 dir = 65536 direcciones -> 2804 Kbytes < 65536 direcciones , por tanto se podría almacenar en el bloque 14.

2804 Kbytes/256 Kbytes = 11 bloques

En total después de la indexación tendremos un total de 13 bloques para el almacenamiento del fichero. Ya que tenemos que sumar los 11 bloques del nivel 2, el bloque de nivel 1 y el bloque de nivel 0. Por tanto, basta con multiplicar los bloques por el tamaño de bloque.

16. En la mayoría de los sistemas operativos, el modelo para manejar un archivo es el siguiente:

- **Abrir el archivo, que nos devuelve un descriptor de archivo asociado a él.**
- **Acceder al archivo a través de ese descriptor devuelto por el sistema.**

¿Cuáles son las razones de hacerlo así? ¿Por qué no, por ejemplo, se especifica el archivo a manipular en cada operación que se realice sobre él?

Nos traemos el i-nodo del archivo (los metadatos) a memoria principal, por tanto, cuando queramos hacer una modificación del archivo lo haremos sobre la copia de los metadatos del archivo que tenemos en memoria principal. Se realiza así por una cuestión de eficiencia y aprovechamiento de recursos.

17. Sea un directorio cualquiera en un sistema de archivos ext2 de Linux, por ejemplo, DirB. De él cuelgan algunos archivos que están en uso por uno o más procesos. ¿Es posible usar este directorio como punto de montaje? Justifíquelo.

La respuesta es sí porque el hecho de que haya uno o más procesos que tengan abiertos archivos de DirB, no tenemos problemas ya que tenemos los metadatos de esos archivos abiertos en memoria principal. Si el propio DirB está siendo utilizado por el proceso, entonces no se puede usar como punto de montaje.