ACTIVIDADES 3, 4, 9, 10 y 11.

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. Tenga en cuenta que:

- El tamaño de bloque es de 512 bytes
- El asterisco indica último bloque
- Todo lo que está en blanco en la figura está libre.

Rellene la figura para representar lo siguiente:

- Creación del archivo DATOS1 con fecha 1-3-90, y tamaño de 10 bytes.
- Creación del archivo DATOS2 con fecha 2-3-90, y tamaño 1200 bytes.
- El archivo DATOS aumenta de tamaño, necesitando 2 bloques más.
- Creación del directorio D, con fecha 3-3-90, y tamaño 1 bloque.
- Creación del archivo CARTAS con fecha 13-3-90 y tamaño 2 kBytes.

Los archivos DATOS1, DATOS2 y D ocuparán 1, 3 y 1 bloque respectivamente. El archivo DATOS pasará a ocupar 4 bloques, le añadiremos los dos extra a partir de su fin de archivo. Por último, como la FAT32 no guarda los punteros de una tabla a la siguiente sino que guarda una tabla aparte, los 512B de cada bloque son aprovechables por el archivo, por lo que el archivo CARTAS ocupará unicamente 4 bloques.

Nombre	Tipo	Fecha	Nº Bloque
DATOS	F	8-2-90	3
DATOS1	F	1-3-90	7
DATOS2	F	2-3-90	4
D	D	3-3-90	10
CARTAS	F	13-3-90	8

FAT			
1	-	10	*(D)
2	-	11	*(DATOS)
3	15	12	13
4	6	13	14
5	*(D2)	14	*(CARTAS)
6	5	15	9

FAT			
7	*(D1)	16	-
8	12	17	-
9	11	18	-

Nota: "-" indica que el bloque no está en uso.

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.

1 si está ocupado, 0 si no está ocupado

001111111111111000

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?

Direccionamos 10 bloques de datos de forma directa, 256 bloques con el primer nivel de indexación, 256^2 bloques con el segundo nivel de indexación y 256^3 bloques con el tercer nivel de indexación. Si cada bloque es de 1KB:

Tamaño máximo = (10 + 256 + 256^2 + 256^3) * 1KB = 16843018 KB = 16,06 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.

Sabemos que cada bloque tiene un tamaño de 512 B. Como cada bloque tiene 512 B y cada puntero se almacena en 4B, cada tabla de indexado tendrá 512/4 = 128 entradas.

En primer lugar tenemos que comprobar que estamos accediendo al programa y no fuera del mismo. Es decir, continuaremos con la conversicón si N < TAMAÑO. Si se diese el caso contrario produciríamos una violación de segmento.

En primer lugar accedemos a la primera tabla de indexado (cuya posición inicial es PRIMERBLOQUE). Para ello realizamos la operación N/(512*128²). El cociente de esta operación (C1) nos indica que posición de la tabla de indexado buscamos mientras que el resto (R1) nos servirá más adelante. La primera dirección a la que accedemos es entonces PRIMERBLOQUE + C1, cuyo valor almacenaremos en SEGUNDOBLOQUE, ya que tendrá la dirección de memoria donde comienza el segundo bloque de indexado.

Para acceder a la tabla de segundo nivel de indexación realizamos un proceso similar al anterior. Dividiremos esta vez R1/(512*128), almacenando el cociente en C2 y el resto en R2. La posición a la que accederemos a continuación será SEGUNDOBLOQUE + C2, cuyo valor almacenaremos en TERCERBLOQUE.

Repitimos el proceso con el tercer bloque: R2/512 almacenando el cociente en C3 y el resto en R3. Accedemos a la posición TERCERBLOQUE + C3 y guardamos su valor en BLOQUE. El valor R3 será el desplazamiento dentro del bloque. Finalmente, para obtener el carácter pedido accedemos a BLOQUE + R3.

Las operaciones realizadas han sido las siguientes:

Verificar N < TAMAÑO
C1 = N / (512 * 128²)
R1 = N % (512 * 128²)
SEGUNDOBLOQUE = M(PRIMERBLOQUE + C1)
C2 = R1 / (512 * 128)
R2 = R1 % (512 * 128)
TERCERBLOQUE = M(SEGUNDOBLOQUE + C2)
C3 = R2 / 512
R3 = R2 % 512
BLOQUE = M(SEGUNDOBLOQUE + C3)
CARACTER = M(BLOQUE + R3)

Donde los accesos a memoria están indicados con ***M(posición)***.

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:

- **Modificados infrecuentemente, y accedidos frecuentemente de forma aleatoria**

Métodos de Asignación de espacio: Contiguo.

Como se modifica infrecuentemente, podemos asumir que el archivo no va a crecer, pudiendo utilizar este método sin el peligro de eficiencia que supone la compactación. Aunque también podríamos utilizar el método "No contiguo indexado", utilizando éste ahorramos la tabla de asignación de archivos (la FAT), que no sería de especial utilidad en este caso particular.

- **Modificados con frecuencia, y accedidos en su totalidad con cierta frecuencia**

Métodos de Asignación de espacio: No Contiguo - Enlazado

Como se modifica de forma frecuente no puede ser "Contiguo", además si es "No Contiguo" puede crecer dinámicamente sin que sea necesario compactar. Por otro lado, como se pide acceder totalmente con frecuencia, sabemos que "No contiguo, Enlazado" permite acceder secuencialmente (perfecto en nuestro caso al querer acceder en la totalidad del archivo) de forma efectiva.

- **Modificados frecuentemente y accedidos aleatoriamente y frecuentemente**

Métodos de Asignación de espacio: No Contiguo - Indexado

Por lo explicado en el apartado anterior, aquí tampoco sería recomendable utilizar acceso contiguo. Por otro lado, el "No contiguo Indexado" permite un buen acceso directo. Frente al método "No contiguo Enlazado" cuyo acceso directo es tremendamente ineficiente, este es sin duda el óptimo.

Trabajo realizado por:
José Antonio Álvarez Ocete
Alberto Estepa Fernández
Julio José Prado Muñoz
Carlos Santiago Sánchez Muñoz