Tema 10

1) En un sistema UNIX en el que el tamaño de bloque es de 1024 bytes y una dirección de bloque ocupa 4 bytes, ¿cuál es el tamaño máximo de fichero?

2) Diseñe un algoritmo para el sistema del ejercicio anterior que permita calcular la dirección física asociada a la dirección lógica de un fichero. Utilice el algoritmo para determinar las direcciones físicas asociadas a las direcciones lógicas 9125 y 1000000.

3) El sistema operativo MS-DOS utiliza una tabla de asignación de ficheros (FAT) para llevar la cuenta de los bloques de disco asignados a ficheros. UNIX guarda esta información en los nodos i. Compare ambos métodos, en cuanto a rapidez, en el caso de accesos aleatorios a ficheros muy grandes suponiendo que:

a) La FAT está siempre en memoria principal.
b) La FAT reside siempre en el disco.

Nota: Suponga que ningún sistema utiliza caché de disco

4) ¿Cuántos accesos a disco hacen falta para abrir la ruta */usr/juegos/asesino* en UNIX? (Suponga que todos los directorios se almacenan a lo sumo en un bloque y que no se utiliza caché de disco).

5) Un sistema de ficheros UNIX utiliza bloques de 1024 bytes y direcciones de bloques de 16 bits. La implementación de los nodos i es especial, pues sólo contienen 8 punteros directos a bloques de datos, un puntero indirecto simple y un puntero indirecto doble. ¿Cuál es el tamaño máximo de fichero en dicho sistema?

6) Determine el espacio máximo que puede llegar a ocupar el registro de los bloques libres en un disco de 40MB, usando un tamaño de bloque de 2KB y direcciones de bloque de 16 bits si se utiliza

a) un mapa de bits
b) una lista enlazada de bloques

7) ¿Por qué hay que mantener el mapa de bits que registra si los bloques de disco están libres en la memoria secundaria y no en la memoria principal?

8) Comprobando la coherencia interna del sistema de ficheros se construyen estos contadores:

En Uso	1	0	1	0	0	1	0	1	1	0	1	0	0	1	0
Libres	0	0	0	1	1	1	0	0	0	1	0	1	1	0	1

¿Qué tipo de comprobación estamos realizando? ¿Existen errores? En caso afirmativo, ¿son errores serios? ¿Por qué?

9) Dado un fichero de 267K en un sistema de ficheros UNIX en el que el tamaño de bloque es de 1K, y las direcciones de bloque son de 4 bytes, ¿ qué gasto en recursos (nodos i, bloques directos, bloques indirectos y entradas de directorio) suponen las siguientes operaciones ?

a) realización de un enlace duro al fichero.

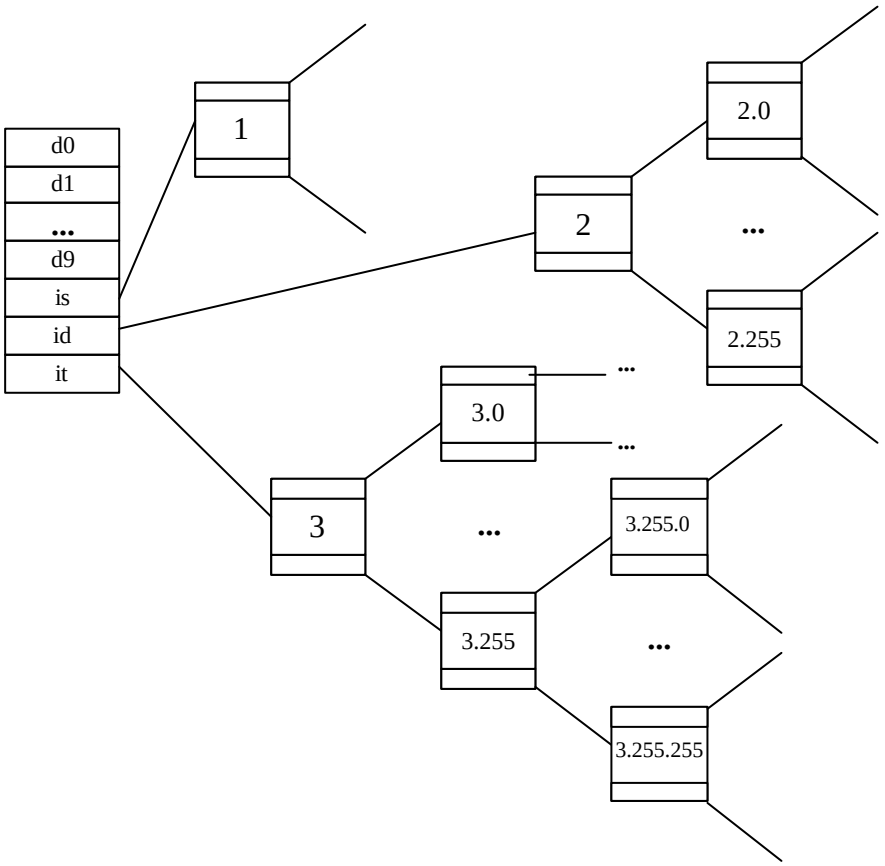
b) realización de un enlace simbólico al fichero.
c) copia del fichero.
d) traslado (operación *move*) del fichero.

Soluciones

1) En UNIX la información sobre los bloques asignados a un fichero se guarda en una pequeña estructura de datos llamada nodo i. Con los parámetros especificados en el enunciado hay que calcular a cuántos bloques puede hacer referencia (directa e indirectamente) un nodo i. Un nodo i contiene 10 punteros directos a bloques de datos del fichero, además, contiene un puntero indirecto simple con el que se apunta a un bloque que contiene punteros a bloques de datos del fichero. Este bloque puede albergar (1024 bytes : 4 bytes/dirección) direcciones, es decir, 256 direcciones a bloques de datos del fichero. Además, en el nodo i existe un puntero indirecto doble que apunta a un bloque que contiene punteros a bloques que contienen punteros a bloques de datos del fichero. Esto representa un total de 256 x 256 punteros a bloques de datos. Por último existe un puntero indirecto triple que, siguiendo un razonamiento análogo, permite guardar referencias a 256 x 256 x 256 bloques de datos.

Luego un fichero puede tener asociados $10 + 256 + 256^2 + 256^3$ bloques, o sea (16843018 bloques x 1024 bytes/bloque) bytes. El tamaño máximo de fichero en este sistema es, pues, cercano a 17 GB.

2) En el siguiente dibujo se representa la nomenclatura que utilizaremos para hacer referencia a los punteros directos a bloques de datos del nodo i (d0, ..., d9), al puntero indirecto simple (is), al puntero indirecto doble (id), al puntero indirecto triple (it) y a los distintos bloques utilizados para guardar punteros.



```
int dir_física( int dir_log) {
    int b;      /* bloque lógico en el que se encuentra la dirección lógica dir_log */
    int ord;    /* es el ordinal que ocupa b dentro del conjunto de bloques accesibles a través del
                   puntero indirecto del nodo i a partir del cual se accede a b */

    b = dir_log div 1024;

    if (b < 10) then {    /* la dir. física está en uno de los bloques apuntados por los punteros directos */
        El bloque físico es el apuntado por el puntero db; /* en db b es el valor de la variable b */
    } else if (b < 10 + 256) { /* la dir. física se debe obtener a partir de is */
        ord = b –10;
        Acceder a 1 mediante is; el bloque físico es el apuntado por la entrada ord de 1;
    } else if (b < 10 + 256 + 256^2) { /* la dir. física se debe obtener a partir de id */
        ord = b – (10 + 256);
        p1 = ord div 256;
        p2 = ord % 256;          /* % es la operación módulo en C */
        Acceder a 2 mediante id y a 2.p1 mediante la entrada p1 de 2; el bloque físico es el apuntado
        por la entrada p2 de 2.p1;
    } else { /* la dir. física se debe obtener a partir de it */
        ord = b – (10 + 256 + 256^2)
        p1 = ord div 256^2;
        p2 = ord % 256^2;
        p3 = p2 div 256;
        p4 = p2 % 256;
        Acceder a 3 mediante it, a 3.p1 mediante la entrada p1 de 3; a 3.p1.p3 mediante la entrada p3
        de 3p1; el bloque físico es el apuntado por la entrada p4 de 3.p1.p3;
    }
    return (dirección de inicio del bloque físico seleccionado en la sentencia if + (dir_log % 1024));
}
```

Apliquemos el algoritmo a la dirección lógica 9125. La dirección se encuentra en el bloque lógico 8 (9125 div 1024), al ejecutar la sentencia *if* anidada se determina que el bloque de datos se obtiene a partir de los punteros directos, concretamente a partir del noveno (*d8*). El dato se encuentra en la posición 9125 % 1024, es decir la 933, del bloque apuntado por *d8*.

Aplicando el algoritmo a la dirección 1000000 se obtiene: La dirección se encuentra en el bloque lógico 976 (1000000 div 1024), la sentencia *if* anidada determina que el bloque de datos se obtiene a partir del puntero indirecto doble. El bloque físico es apuntado por la entrada 198 del bloque 2.2, y el dato se encuentra en la posición 1000000 % 1024, es decir la 576.

3) En UNIX, cuando un proceso abre un fichero la información de su nodo i, que incluye los punteros directos e indirectos a sus bloques de datos, es copiada en su PCB. Esta copia se realiza para no tener que acceder al nodo i (lo que implica una lectura de disco) para obtener las direcciones de sus bloques cada vez que se lee o escribe en el fichero (el PCB está en memoria principal).

Si se realizan accesos aleatorios se estará accediendo a posiciones distribuidas aleatoriamente en el fichero. Como el fichero es de gran tamaño se puede suponer que en algunos accesos será preciso utilizar hasta el tercer nivel de indexación. Si el acceso se sitúa en los 10 bloques primeros del fichero, sólo hay que consultar uno de los 10 punteros directos y realizar un sólo acceso a disco para leer el bloque de datos correspondiente. Si la lectura se sitúa en un bloque posterior, lo cual es probable, pues es aleatoria y el fichero grande, se tendrán que consultar los bloques de direcciones, lo que implica 2,

3 o 4 accesos a disco para leer un bloque de datos dependiendo de que el bloque se encuentre indirectamente apuntado de manera simple, doble o triple.

El caso de MS DOS es distinto. Se trata de recorrer la FAT para obtener la dirección del bloque que se quiere leer. Si la FAT está en memoria principal el tiempo empleado en su recorrido es despreciable frente a un acceso a disco. Por lo tanto, en promedio se obtiene un rendimiento mejor que el de UNIX. En este último sólo lecturas de los 10 primeros bloques implicaban un acceso a disco, con la FAT en memoria principal siempre se requiere un único acceso a disco.

Si la FAT está en disco, el recorrido implica accesos a disco. El acceso al bloque i precisará de i accesos a disco (se supone que no hay caché) para localizar su dirección en disco. Por lo tanto, el esquema de UNIX es más eficiente si la FAT está en disco.

4) En UNIX, abrir una ruta consiste en almacenar su nodo i en el PCB del proceso que realiza la apertura. En el caso de la ruta */usr/juegos/asesino* primero hay que consultar el directorio */* para encontrar la entrada *usr*, después hay que consultar el directorio */usr* para encontrar la entrada *juegos*, y por último hay que consultar */usr/juegos* para encontrar la entrada *asesino*. La consulta de cada uno de los directorios implica dos accesos a disco, uno para leer su nodo i y otro para leer su primer bloque de datos, donde se encuentran sus entradas (recuerde que se supone que un directorio se almacena en un único bloque, cuya dirección se almacenará en el primer puntero directo a bloques de datos del nodo i). La consulta de los directorios */*, */usr* y */usr/juegos* implica, por tanto, 6 accesos. Además, se precisa de otro acceso para leer el nodo i del fichero o directorio *asesino*.

6) El disco tiene un tamaño de 40MB y cada bloque tiene un tamaño de 2KB, por lo tanto el disco contiene (40MB : 2KB/bloque) bloques, es decir 20K bloques.

a) Se necesitan tantos bits como bloques haya en el sistema para registrar la situación (ocupado o libre) de cada uno de los bloques. Luego se necesitarán 20K bits. Se trata ahora de calcular cuántos bloques (unidad lógica de acceso mínima al disco) ocupan 20K bits. Calculémoslo:

20K bits : 16K bits/bloque = 1.25 bloques, es decir, se precisan **2 bloques**

b) Se anotan las direcciones de los bloques libres en una lista enlazada de bloques. Cada dirección es de 16 bits (2 bytes). En un bloque podemos hacer referencia a un total de (2KB : 2B/dirección) direcciones, es decir 1024 direcciones, de las cuales 1023 apuntan a bloques libres y una al siguiente bloque de la lista enlazada. Poniéndose en el peor caso, es decir, que los 20K bloques del disco estén libres se necesitan:

20K : 1023 direcciones/bloque = 20.02 bloques. Luego se precisan a lo sumo **21 bloques**

7) Por su gran tamaño. Normalmente los discos dispondrán de una gran capacidad de almacenamiento secundario, y por tanto, de un gran número de bloques.

8) Se está realizando una comprobación de coherencia de bloques. En las columnas 2, 6 y 7 se detectan errores. El tipo de error de las columnas 2 y 7 no es grave, indica que hay dos bloques que ni están siendo utilizados por ningún fichero ni aparecen como libres en la lista de bloques libres, si persiste esta situación los dos bloques nunca se utilizarán, es decir, se desaprovecharán. El error de la columna 6 sí es potencialmente grave en cuanto a la coherencia del sistema. El bloque está siendo utilizado por un fichero, pero también aparece en la lista de bloques libres. Por lo tanto, este bloque podría ser asignado a otro fichero, lo que llevaría a la situación inconsistente de que fuera utilizado por dos ficheros distintos.