DISEÑO DE SISTEMAS OPERATIVOS. 21-6-2006. Examen de la parte teórica

Ejercicio 2 (4,5 puntos)

Se dispone de un dispositivo grabador y reproductor de audio digital comprimido (mp3) o sin comprimir (wav), que se puede conectar a un ordenador vía un puerto USB 2.0 (8MB/seg.), con una capacidad de 512 MB (tiene un disco duro interno de este tamaño basado en tecnología de memoria flash no volátil), suficientes para contener unas diez horas de música, o unas 150 canciones.

Cuando no está conectado por el USB a un ordenador, es un grabador y reproductor de sencillo manejo a través de unos botones y de un pequeño display. El pequeño sistema empotrado del dispositivo sabe reconocer y reproducir los archivos con extensión mp3 ó wav así como grabar en un nuevo archivo wav lo que registra con un pequeño micrófono incorporado.

Cuando se conecta a un sistema Linux vía el USB, se comporta como un *pen-drive*. El disco interno del reproductor se hace presente y accesible en el sistema Linux a través del dispositivo "/dev/sda1". Podremos entonces montarlo y transferir archivos en uno u otro sentido.

Existe la restricción de que el disco interno debe tener formato FAT.

a) Dé razones que justifiquen esta restricción o argumentos en contra de tal decisión si la considera arbitraria o existe otra claramente mejor o más adecuada.

b) Dibuje y dimensione el sistema de archivos de este dispositivo, indicando sus diferentes partes, y el tamaño de cada una de ellas y su contenido inicial.
Razone y justifique adecuadamente las decisiones que tome.
Considere de igual tamaño las unidades: agrupación (en inglés *cluster*) y bloque.

Suponga que realizamos el experimento de, a partir del disco vacío, hacer una única grabación hasta llenar el disco con un único archivo de nombre `REC00001.wav`.

c) Dibuje y explique cuál será el contenido de la FAT en este momento.
Explique y calcule qué cantidad de información del disco debe ser leída para conseguir acceder directa y exclusivamente al último byte de este fichero.
¿Cuánto tiempo tardaría la operación descrita dada la tasa de transferencia de un USB 2.0?

d) Una vez convenientemente montado el dispositivo /dev/sda1 sobre el directorio /mnt/mp3, se ejecuta el mandato

```
cp /mnt/mp3/REC00001.wav /dev/null
```

que copia el mencionado archivo sobre el dispositivo sumidero (la papelera).
Se pide que estime razonadamente cuento tiempo llevará la ejecución del mandato anterior, en los dos supuestos siguientes:

- **1)** El sistema posee una cache de bloques suficientemente grande con políticas LRU y de escritura retardada (*delayed write*).
- **2)** La cache de boques del sistema está desactivada en lo que respecta a este dispositivo.

A la vista de los resultados del apartado anterior parece que un sistema operativo de propósito general como Linux puede llegar a ser muy ineficiente sin cache. Sin embargo, el pequeño sistema empotrado del dispositivo (obviamente sin cache de bloques) es capaz de leer y escribir los archivos que manipula, con una latencia de acceso independiente del tamaño del archivo, esto es, tarda lo mismo cuando está reproduciendo la primera parte de un archivo que cuando reproduce la última. Dicho de otro modo, este dispositivo funciona bajo restricciones de tiempo real para poder reproducir audio a velocidad constante así como para poder grabarlo a tasa constante.

e) Esto es sin duda paradójico, pero tiene su explicación en la "naturaleza secuencial" de los archivos manipulados y en la diferente forma de manipular los archivos abiertos. Imagine y explique esta diferencia.

SOLUCIÓN

a) Requisito FAT.

Aunque el formato FAT parezca técnicamente obsoleto, sigue siendo perfectamente apto para muchos usos, y este es uno de ellos. Las ventajas fundamentales de que el disco interno tenga formato FAT son:

- La sencillez. Frente a sistemas de archivos más modernos, este resulta sencillo de implementar y esto es muy importante para que el sistema empotrado del dispositivo lo sepa manipular. Si le diésemos cualquier otro formato, el sistema empotrado del dispositivo no sería capaz de manipularlo.
- La portabilidad entre equipos. Si queremos poder conectar nuestro reproductor a prácticamente cualquier equipo (con cualquier sistema operativo) y poder ver los archivos que contiene, necesitamos que el sistema de archivos sea uno que todos esos sistemas operativos implementen.
- La compatibilidad entre equipos. por ser ya antiguo, no hay discrepancias a la hora de implementar su utilización. todo sistema operativo que lo manipula lo hace de igual forma, y los cambios que se hagan estando conectado a un equipo serán correctamente interpretados al conectarlo a cualquier otro equipo.

b) El formato FAT, ofrece el siguiente dibujo: [SB][FAT1][FAT2][DROOT][Resto...]

[SB] es el Sector de boot, NO el superbloque. Es un sector (512B) reservado para contener código de arranque del sistema junto con un espacio de datos con parámetros que configuran el tipo de sistema de ficheros FAT. El sistema de archivos FAT **no** tiene superbloque.

[FAT] cada copia de la FAT, Tabla de Ubicación de Archivos. Tiene una entrada por *cluster* o unidad de espacio de disco asignable a un archivo. El ancho (tamaño) de estas entradas debe ser suficiente para poderlas numerar todas. Hablamos de FAT12, FAT16 o FAT32 indicando el ancho de las mismas en bits. En este caso, con FAT16 es suficiente. 2^29 B/disco / 2^16 cluster/disco = 2^13 B/cluster = 8 KB/cluster. Es una cifra perfectamente razonable cuando se indica que se prevén pocos archivos de gran tamaño. Se mantienen varias copias de la FAT para luchar contra la posible corrupción del sistema. El cluster nº 0 no se utiliza. El nº 1 se refiere al ocupado por el directorio raíz.

[DROOT] es el directorio raíz y viene a continuación de la última copia de la FAT. Ocupa un *cluster*.

[Resto...] es el espacio asignable a archivos y directorios.

Inicialmente lo único ocupado será el directorio raíz. El cluster correspondiente (Ej. el número 1) será el primero y el último y así estará marcado en la FAT. La entrada número 0 contiene el información que caracteriza el tipo de FAT. El directorio raíz estará inicializado y tendrá ocupadas dos entradas de nombre "." y ".." asociadas ambas al cluster raíz (el número 1).

Cada copia de la FAT ocupará 2*2^16 Bytes = 2^17 B luego 16 clusters.

c) Tras grabar REC00001.wav.

Al haber grabado el archivo de un golpe, se le habrán asignado todos los clusters del disco de forma secuencial. La FAT, contendrán una lista con los cluster asignados al nuevo archivo, del primero (el nº 2) al último, ofreciendo el siguiente dibujo: [XX][EOF][3][4][5]...[EOF] correspondiendo con las entradas 0, 1, 2, etc de la FAT.
Consideraremos que el último cluster asignable será el número 2^16 para que los cálculos sean más redondos:
(2^29 B/disco – 2^9 B(SB) – 2*2^17 B(FATs)) / 2^13 B/cluster ~= 2^16 – 2*16. clusters/disco ~= 2^16.
Habrá ocupada una entrada más del directorio raíz, con el nombre "REC00001.wav" apuntando al cluster nº 2.

Para acceder directamente al último byte del archivo, habrá que averiguar dónde está almacenado su último bloque (sinónimo de cluster). Para ello, recorreremos las entradas de la FAT, desde la primera asignada al archivo he indicada en su entrada de directorio. Se trata pues de leer toda la FAT (2^17 B) más el último cluster asignado al archivo.
(2^17 B + 2^13 B) / 2^23 B/seg ~= 2^-5 segundos.

d) Tiempo estimado para: `cp /mnt/mp3/REC00001.wav /dev/null`

Del enunciado ha de deducirse que el único tiempo no despreciable, para la realización de la mencionada copia es el tiempo que llevará la lectura completa (secuencial) del archivo de origen. El sistema de archivos se encuentra montado en el directorio /mnt/mp3. Esto, dependiendo de la implementación del Sistema Operativo, puede significar:

- que se mantenga una copia de la FAT en memoria durante todo el tiempo que el dispositivo esté montado
- o que no se haga esto y que la FAT se acceda por demanda como cualquier otro bloque del dispositivo.

Las dos opciones son posibles. La primera ofrecería iguales prestaciones en el acceso con o sin cache. Por ello interesa estudiar la segunda.

1) Si disponemos de una cache de bloques lo suficientemente grande, cada bloque deberá ser leído una única vez. La información que debe leerse es, básicamente, toda la FAT y todo el espacio para datos. En total y redondeando, los 512 MB del dispositivo. Luego: 512 MB / 8 MB/s = 2^29 / 2^23 = 2^6 = 64 segundos.
2) Considerando ahora que el sistema no realiza cache de los bloques de este dispositivo, pero entendiendo que los bloques del fichero fueron asignados de manera estrictamente secuencial, cada bloque de la FAT que se acceda nos servirá para avanzar hasta 2^12 posiciones de la lista enlazada de bloques del archivo. Esto significa que el último bloque de la FAT será leído 2^12 veces (1 para cada uno de los (aprox.) 2^12 bloques de datos a los que apunta). El penúltimo será accedido el doble de veces, el antepenúltimo el triple, y así hasta el primer bloque de la FAT que será accedido 16 veces (la FAT ocupa 16 clusters) dicha cantidad. El número total de accesos a bloque serán:
   2^16 blks(datos) + 2^12 * Sum{1,16}(i) blks(FAT) = 2^16 + 2^12 * (17*16/2) blks = 2^16 + 2^15 + 2^19 blks.
   Luego el tiempo empleado será:
   (2^16 + 2^15 + 2^19) blks * 8 KB/blk / 8 MB/s = (2^6 + 2^5 + 2^9) s = 64+32+512 = 608 s ~= 10 minutos.

e) Aparente ineficiencia

Como hemos hecho notar en el apartado anterior, la ineficiencia ocurriría sólo si se desactiva la cache y si la implementación del manejo del sistema de archivos FAT montado no incorpora la optimización mínima de mantener una copia de la FAT en memoria.

Lo primero que debemos constatar es que Linux es un Sistema Operativo de propósito general, y por lo tanto, para ganar en generalidad, no siempre se puede utilizar la implementación más óptima. El sistema empotrado de un dispositivo como este, no tiene esta restricción y sí la de ser eficiente, a la vez que sencillo y fiable, para un uso no general sino muy específico.

Concretamente, el mandato `cp` del apartado anterior, en Linux, se implementaría como un bucle con llamadas `read` y `write` sobre descriptores asociados a una posición actual sobre el fichero que iría incrementándose. En otras palabras, el concepto de posición actual sobre el fichero (dada como desplazamiento desde el principio del archivo) es lo que permite el recorrido secuencial del mismo. A cada acceso secuencial (dado el desplazamiento desde el principio del archivo) se ha de volver a determinar cual es el bloque o bloques involucrados en la operación. Esto es algo muy costoso si el direccionamiento de bloques subyacente es intrínsecamente secuencial, como es el caso de la FAT.

Frente a esta generalidad-ineficiente, el sistema empotrado puede realizar la lectura y escritura secuencial de los archivos, conociendo, no el desplazamiento en bytes desde el origen, sino simplemente el número de cluster que en cada momento se esté procesando (y quizás el desplazamiento dentro del cluster). Así, sencillamente se evita volver a posicionarnos desde el principio a cada acceso y obtenemos una latencia constante (independiente) de la posición y del tamaño del archivo, ya que nos limitamos a ir realizando la operación “ir al siguiente cluster” para avanzar secuencialmente sobre el archivo.