Ejercicio 2 (4,5 puntos)

Un Ingeniero informático (el usuario `pepe`) vuelve de vacaciones de verano con la tarjeta de memoria (flash SD de 256MB) de su cámara digital prácticamente llena y decide colgar sus fotos de una página Web personal del servidor Apache que ejecuta permanente en su puesto de trabajo, un sistema GNU/Linux.
Esta máquina dispone de una bahía que acepta varios tipos de tarjetas de memoria, y permite su acceso a través del dispositivo de bloques `/dev/MF0a` sin latencia apreciable. El sistema está correctamente administrado para el montaje automático de este tipo de dispositivos extraíbles con sólo insertarlos en la ranura correspondiente. En este caso el contenido de la tarjeta aparecería bajo el directorio `/mnt/flash`. Así mismo, este sistema tiene un único disco duro de 400GB, con tres particiones, una raíz (`/`) otra de *swap* y otra para `/home` (de 300GB).

Para contestar a los dos siguientes apartados (A y B), razone sobre la naturaleza del dispositivo y el uso que se hace de él.

A) Para el montaje de las cuentas de usuario (`/home`), **conteste**:

- ¿Qué tipo de sistema de ficheros sería más adecuado y porqué?
- Justifique el uso de la cache de bloques.
- ¿Hay algún otro parámetro/opción que considere razonable?

B) Para el montaje de la tarjeta de memoria (`/dev/MF0a` sobre `/mnt/flash`), **conteste**:

- ¿Qué tipo de sistema de ficheros sería más adecuado y porqué?
- Justifique el uso de la cache de bloques.
- ¿Hay algún otro parámetro/opción que considere razonable?

Publicar las fotos en la Web implica colgarlas en algún directorio por debajo de `/home/pepe/public_html`, que es donde están las páginas Web del usuario `pepe`. se plantean tres alternativas (C, D y E):

C) Primera alternativa. Copiar las fotos con el mandato:

```
cp /mnt/flash/fotos/* /home/pepe/public_html/Ago07/fotos1/
```

En relación con la ejecución de este mandato, considere los siguientes eventos:

- Instante en que se devuelva el *prompt*.
- Instante en que los datos quedan almacenados en el dispositivo destino.

Conteste: En relación con las acciones del sistema de ficheros identifique justificadamente el instante en que sucede el primer y el segundo eventos. ¿Puede haber una diferencia apreciable de tiempo entre ellos?

D) Segunda alternativa. Realizar un enlace que haga accesible el directorio `/mnt/flash/fotos/` bajo la ruta `/home/pepe/public_html/Ago07/fotos2`.
Conteste: Escriba el mandato que permita esto.
Suponga que una visita web provoca que el proceso servidor Web haga la llamada

```
open("/home/pepe/public_html/Ago07/fotos2/DSCN2981.jpg", O_RDONLY);
```

Describa detalladamente:

- cómo se va realizando la decodificación de esta ruta
- qué sistemas de archivo se visitan, en qué orden y porqué.
- qué determina en cada caso el cambio de un sistema de ficheros a otro durante la decodificación.

E) Tercera alternativa. Montaje directo de `/dev/MF0a` sobre el directorio `/home/pepe/public_html/Ago07/fotos3`.
Esto supondría tener montado "dos veces" el mismo dispositivo sobre dos puntos de montaje distintos del árbol de nombres. Esto es algo que la mayoría de los sistemas operativos no soportan.
Conteste: Justifique la dificultad de permitirlo, razonando sobre cómo gestiona internamente el sistema los i-nodos involucrados en el montaje (los del los directorios punto de montaje y raíz del sistema montado).
Conteste: Si fuese usted el administrador de este sistema, ¿permitiría que cualquier usuario montase la tarjeta de memoria en cualquier directorio sobre el que tuviese permisos de escritura? Justifique la respuesta.

Solución

A) Para la partición montada sobre `/home`:

- De los varios sistemas de ficheros tipo UNIX soportados en Linux, alguno de los más modernos y que implemente *journaling*.
  El "uso general" que se va a realizar de esta partición destinada a dar soporte a las cuentas de usuario en un sistema GNU/Linux (punto de montaje `/home`), precisa un sistema de ficheros basado en nodos-i que soporte todas las características modernas de un sistema UNIX (multiusuario, cuotas de disco, etc.) y que sea lo más eficiente posible. Así mismo, dado el gran tamaño de la partición (300 GB), es imprescindible que el sistema tenga un buen comportamiento en recuperación ante posibles caídas que puedan corromper el sistema.
  Escogeríamos por lo tanto, un sistema de ficheros moderno que implemente *journaling*: ext3, JFS, Reiser, etc.
- La necesidad de la cache de bloques está plenamente justificada dado el "uso general" previsto y que el dispositivo subyacente es un disco duro convencional. Dicho de otro modo. No hay razones para no utilizarla, y usándola es seguro que el rendimiento general del sistema mejorará.
- Una vez más, dado el "uso general" previsto, es muy posible que la configuración por defecto sea la adecuada.

B) Para la tarjeta de memoria flash:

- Por compatibilidad/portabilidad, el sistema de ficheros adecuado es de tipo FAT.
  Hay que recordar que estas tarjetas vienen ya formateadas y que son/deben_ser reconocidas por todos los sistemas operativos, incluido el pequeño sistema empotrado que ejecutan, por ejemplo, las cámaras de fotos. Dado el tamaño que pueden llegar a tener estas tarjetas, lo recomendable (para no consumir agrupaciones demasiado grandes) sería una FAT32.
- La utilidad de la cache de bloques sobre un dispositivo como este es discutible.
  Por un lado, por homogeneidad de tratamiento dentro del sistema operativo, no vamos a hacer que el servidor de bloques utilice la cache de bloques para unos bloques sí y para otros no según del dispositivo al que se acceda. La cache de bloques se utilizará siempre, durante el acceso a cualquier bloque, no importa la naturaleza del dispositivo (orientado a bloques).
  No obstante, en el caso que nos atañe, el dispositivo en cuestión es básicamente "memoria" y ,según indica el enunciado, puede ser accedido "sin latencia aparente", lo cual viene a significar, a toda velocidad. Este razonamiento viene a constatar que el uso de la cache de bloques en este caso no va a suponer una mejora en el rendimiento de los accesos a este dispositivo.
- Por otro lado, se trata de un dispositivo "extraíble", como lo eran los disquetes o lo son los *pendrives*. Esto indica que el usuario puede estar tentado de sacar el dispositivo sin avisar al sistema operativo para que realice el desmontaje ordenado del mismo. Por esta razón, este tipo de dispositivos se suelen montar en modo de escritura síncrona. Esto significa básicamente que todo acceso de escritura sobre el dispositivo pasará por la cache de bloques, pero siempre en modalidad *write_through*, al objeto de minimizar la ventana de tiempo en que una extracción inesperada pueda dejar el sistema de ficheros inconsistente.
  También sería oportuno matizar en el montaje aspectos relacionados con la seguridad.

C) El mandato indicado, copia cada uno de los archivos encontrados en el directorio origen (`/mnt/flash/fotos`) al directorio destino (`/home/pepe/public_html/Ago07/fotos1`). Internamente la copia de cada archivo consiste en: apertura del archivo origen, creación del archivo destino y bucle de copia, leyendo del origen y escribiendo en el destino, todo el contenido de cada archivo.

```
fdi = open(origen, O_RDONLY);
fdo = creat(destino, 0666);
while((ret = read(fdi, buf, SIZE)) > 0)
            write(fdo, buf, ret);
close(fdo); close(fdi);
```

- El intérprete de mandatos o *shell*, devolverá el *prompt* cuando retorne la llamada `WAIT` que le hace esperar a que termine el proceso que ejecuta el mandato de copia.
  Curiosamente, esto sucederá antes de que los datos copiados queden almacenados en el dispositivo destino. La razón de este comportamiento reside en el uso de la cache de bloques.
  Las llamadas `WRITE` realizadas, dejan los datos en la cache de bloques, no en el dispositivo de destino.
- El instante en que, finalmente, los datos quedarán almacenados en el dispositivo de destino dependerá de la política

de escritura establecida para la cache de bloques. Para un sistema de ficheros de "propósito general" como es el caso, la política muy probablemente sea *delayed-write*, lo que implica que cada 30 segundos (aprox.) sucederá una llamada a `SYNC` que volcará los contenidos de la cache de bloques a disco.

Así pues, la diferencia en tiempo entre ambos instantes puede llegar a ser de unos 30 segundos.

D) El enlace que se nos solicita ha de ser un enlace simbólico. No cabe considerar un enlace físico, dado que estos sólo pueden realizarse dentro de un mismo sistema de archivos.

- Mandato: `ln -s /mnt/flash/fotos /home/pepe/public_html/Ago07/fotos2`
  Llamada: `symlink("/mnt/flash/fotos","/home/pepe/public_html/Ago07/fotos2");`

- La decodificación de una ruta es la labor que realiza internamente el sistema operativo cada vez que algún proceso realiza una llamada al sistema que reciba como argumento una ruta o *path*. El sistema operativo tratará de averiguar a que objeto dentro del árbol de nombres apunta dicha ruta.
  La decodificación es pues el recorrido de una ruta, desde su origen (absoluto o relativo), componente a componente, directorio a directorio, comprobando la existencia de cada componente/subdirectorio en el árbol de nombres, hasta alcanzar el v-nodo que represente el archivo/objeto referido por la última componente de dicha ruta.

- El algoritmo `decodifica(ruta)` hace uso sistemático del `avanza(nodo,nombre)`, una subrutina que, dado el v-nodo de un directorio (que represente el punto alcanzado en el recorrido del árbol de directorios), y el nombre de la siguiente componente de la ruta, lee del sistema de ficheros el contenido del directorio alcanzado y busca en él dicho nombre y devuelve el v-nodo correspondiente, si lo encuentra. Así mismo, se va comprobando si se tienen o no permisos para el acceso, si se trata de un punto de montaje, si se trata de un enlace simbólico u otras posibles eventualidades.

```
nodo = decodifica("/home/pepe/public_html/Ago07/fotos2/DSCN2981.jpg")
```

Como es un *path* absoluto se parte del directorio raíz del sistema de ficheros raíz.

```
nodo = <v-nodo raíz de /dev/hda1>
nodo = avanza(nodo,"home");
```

El v-nodo devuelto es un punto de montaje. Se continúa por el i-nodo raíz del sistema montado.

```
nodo = <v-nodo raíz de /dev/hda3>
nodo = avanza(nodo,"pepe");
nodo = avanza(nodo,"public_html");
nodo = avanza(nodo,"Ago07");
nodo = avanza(nodo,"fotos2");
```

El v-nodo devuelto es de un enlace simbólico a `/mnt/flash/fotos`. Hay que decodificarlo.

```
nodo = decodifica("/mnt/flash/fotos")
```

Como es un *path* absoluto se parte del directorio raíz del sistema de ficheros raíz.

```
nodo = <v-nodo raíz de /dev/hda1>
nodo = avanza(nodo,"mnt");
nodo = avanza(nodo,"flash");
```

El v-nodo devuelto es un punto de montaje. Se continúa por el i-nodo raíz del sistema montado.

```
nodo = <v-nodo raíz de /dev/MF0a>
nodo = avanza(nodo,"fotos");
return nodo;
nodo = avanza(nodo,"DSCN2981.jpg");
return nodo;
```

- Como se ha mostrado, durante la decodificación de una ruta, el sistema operativo examina cada v-nodo alcanzado para comprobar, entre otras cosas, si se trata de un punto de montaje. En el caso de pasar sobre un punto de montaje, el v-nodo contendrá indicación de qué sistema de archivos está montado sobre dicho directorio. El paso siguiente es ir a la tabla de superbloques (estructura en memoria donde residen los superbloques de todos los sistemas de ficheros montados), localizar el que está montado sobre este punto y retornar el v-nodo de su i-nodo raíz, para continuar desde éste la decodificación.

E) El montaje directo de `/dev/MF0a` sobre un directorio de la cuenta del usuario presenta los siguientes inconvenientes:

- Tener montado "dos veces" el mismo dispositivo sobre dos puntos de montaje distintos del árbol de nombres, supone nada más y nada menos que tal árbol deje de serlo.
  Solemos decir del árbol de nombres de UNIX que es más bien un grafo dirigido acíclico. Decimos esto porque cada v-nodo (de cada directorio del árbol) tiene un único padre, esto es, que hay una única ruta por la que se llega a él. Esto es siempre así, salvo dos excepciones:
  - Los enlaces físicos permiten que una hoja del árbol (un archivo convencional) "cuelgue" de dos o más directorios simultáneamente. Pero esto no es un problema porque son "hojas", no directorios. Se impiden los enlaces físicos a directorios para prevenir la creación de ciclos que convertirían el árbol en grafo cíclico.
  - Los enlaces simbólicos, por su lado, se permiten entre sistemas de ficheros distintos y apuntando a directorios. Pero esto no supone ningún problema porque los enlaces simbólicos son objetos especiales, distinguibles como tales.

  El montaje de un sistema de ficheros se refleja sobre los v-nodos de dos directorios: el punto de montaje del sistema de ficheros subyacente y el directorio raíz del sistema de ficheros montado. Al tratarse de v-nodos. estas anotaciones no irán nunca a disco, y decimos que el montaje de sistemas de ficheros es algo virtual.
  El problema es que montar dos veces un mismo sistema de archivos precisaría que el v-nodo del directorio raíz del sistema montado tuviese dos padres, correspondientes a los dos directorios punto de montaje. Esto destruiría la estructura en árbol del los nombres en UNIX e impediría su correcta gestión.

- Permitir que cualquier usuario monte dispositivos de cualquier tipo presenta fundamentalmente el problema de la seguridad. Esto es así porque los sistemas de ficheros UNIX no son sólo contenedores de datos bajo un nombre de fichero, sino que esos ficheros tienen propietarios, pueden ser ficheros ejecutables y esos propietarios pueden ser cualquiera o el administrador o "*root*". Recuerde cómo se utilizan los bits SETUID y SETGID.
  Todas estas consideraciones pueden ser tenidas en cuenta por el administrador a la hora de parametrizar las operaciones de montaje permitidas, pero para garantizar la seguridad es mucho más fácil simplemente impedir el montaje indiscriminado y sólo permitir unos pocos montajes automáticos muy controlados.