EXAMEN SISTEMAS OPERATIVOS AVANZADOS

MARTES 17 ENERO 2017

NOMBRE Y FIRMA:

P1:	/1	P3:	/2	LAB5,P1:	/1.5
P2	/2	P4:	/2	LAB5,P2:	/1.5

PROBLEMA 1 (1 PUNTO)

Suponiendo que el tamaño de bloque del sistema de ficheros es 4KB, cuanto es el tamaño de fichero máximo que soportaría:

a) Un sistema de ficheros realmente simple, donde cada i-nodo tiene únicamente 10 punteros directos. Cada puntero tiene 32 bits (4 bytes).

b) Un sistema de ficheros, mejorado, donde los i-nodos además de los 10 punteros directos, tienen 1-puntero indirecto, y un puntero doble indirecto. Los punteros siguen requiriendo 32bits.

c) Un sistema de ficheros compacto que intenta ahorrar el espacio dedicado a los i-nodos. Para ello, el i-nodo únicamente tiene capacidad para un puntero. Si el fichero requiere más de 4KB lo que hace es incluir un puntero en el final de cada bloque de datos al siguiente bloque.

PROBLEMA 2 (2 PUNTOS)

a) En un sistema de ficheros FFS (sin *journaling*) suponer que en el proceso de actualización de un fichero (al que se estaba agregando un bloque de datos) la corriente del sistema se interrumpe (y el sistema no tiene SAI). Suponiendo que el corte de luz se produce antes de que la aplicación escriba el bloque de datos en memoria, ¿Qué problemas de consistencia nos podemos encontrar para cada una de las combinaciones de escritura en el disco?

(…)

b) ¿Cómo logra los sistemas basados en *journaling* "emular" la escritura atómica del *journal*?

c) ¿Cuál sería el rendimiento de un sistema de ficheros con "*data journaling*" (también llamado *journaling* físico) con respecto a uno con "*metadata journaling*"?

d) En los sistemas que usan "*metada journaling*", Como se evita que una operación antigua deshaga cambios más recientes en el sistema de ficheros tras un crash-recovery.

PROBLEMA 3 (2 PUNTOS)

a) Suponer que tenemos un disco con un solo *track*, y una política de planificación interna FIFO. El tiempo de rotación del disco es R. No hay *seek* (solo hay un *track*). Supongamos que el coste de transferencia es nulo. ¿Cuál es (aproximadamente) el tiempo de acceso de peor caso si tenemos escribir 3 sectores (diferentes)?

b) En estas mismas condiciones, ¿Cuál será el tiempo de acceso de peor caso si en lugar de FIFO usamos **Shortest Positioning Time First**?

c) Ahora, suponer que el disco tiene tres *tracks*. El tiempo de *seek* entre dos *track* adyacentes es ***S***. ¿Cuál es el tiempo de acceso de peor caso si el disco usa FIFO?

d) ¿Y si de nuevo pasamos a usar SPTF?

PROBLEMA 4 (2 PUNTOS)

Suponer un sistema FFS con 10 grupos de cilindros (o grupos de bloques). El directorio raíz está en el grupo de bloques 0 y comienza vacío. Suponer que todos los directorios y ficheros en el problema tienen un bloque de datos (a no ser que se indique de otro modo).

a) Existe un fichero llamado "/foo" en el grupo de bloques 3. Describir los accesos a disco que deberían llevar a cabo para acceder al contenido del fichero. Los accesos se deben especificar con una secuencia #{I,D}, es decir si queremos acceder a un inodo en el grupo de bloque 6 deberíamos indicar 6I.

b) Se quiere crear un fichero vacío, llamado "bar" en el directorio en el que esta "foo".

c) Suponer que ahora que alguien quiere determinar (mediante un ls) el tamaño de los ficheros en el directorio "raíz".

d) Suponer que creamos otro fichero llamado "big" que va creciendo de forma progresiva hasta llenar el disco. Queremos que el ancho de banda al disco utilizado sea al menos 20MB/s, siendo el seek de 10ms y el ancho de banda disponible de 40MB/s. Suponer que los cambios en el mismo grupo de bloques no requieren *seek*.

PREVIO

Descargar el repositorio de trabajo desde: https://gitlab.com/AOSUC/examen3/

Dentro hay un directorio llamado “xv6” que contiene una implementación parcial (y defectuosa) de la práctica 5. Se proveen de dos programas de usuario (que forman parte de los *tests* empleados) llamados P1.c y P2.c. Ninguno de los dos logra pasar. Indicar debajo donde están los problemas en cada uno de ellos (indicando fichero, función, la línea/s incorrecta y la/s corregida/s). Notar que la implementación proporcionada usa memmove (como hace xv6 en los ficheros regulares).

LAB 3 P1 (1.5 PUNTO)

LAB 3 P2 (1.5 PUNTO)