Sistemas Operativos
2° Parcial 1C2024 – TT – Resolución

Aclaración: La mayoría de las preguntas o ejercicios no tienen una única solución. Por lo tanto, si una solución particular no es similar a la expuesta aquí, no significa necesariamente que la misma sea incorrecta. Ante cualquier duda, consultar con el/la docente del curso.

Teoría

1. Las métricas a analizar son la utilización del CPU con un valor bajo y la cantidad en aumento de los fallos de página produciendo un aumento en cantidad de operaciones de E/S. Para solucionar el problema se requiere más memoria principal aunque también puede reducirse el nivel de multiprogramación permitiendo el crecimiento del conjunto residente de los procesos que general los fallos de página.
2. Se podría usar paginación jerárquica para reducir el tamaño de la tabla de páginas y considerarse una TLB para disminuir los tiempos de acceso a memoria.
3. Se debe abrir el archivo y luego la operación básica renombrar o mover es suficiente para mover el archivo a otro directorio. Luego debe cerrarse el archivo. En caso de que se mueva a otro directorio Es necesario:

Directorio Origen	**Directorio Destino en otro FS**
Abrir el archivo	
	Crear el archivo
	Abrir el archivo
Leer el archivo	
	Truncar/Agrandar el archivo
	Escribir el archivo
Cerrar el archivo	
	Cerrar el archivo
Eliminar el Archivo	

(No es la única respuesta posible)

4. El inodo contiene, entre otros datos:
    a. Propietario

b. Grupo
c. Permisos PGU
d. Tipo de archivo (Regular, Directorio, Softlink, etc)
e. Timestamps de acceso, modificación y creación
f. Contador de hardlinks
g. Tamaño
h. Punteros directos e indirectos

Al crear un hardlink no se crea un nuevo inodo, simplemente se crea una nueva entrada de directorio que referencia al inodo existente y se aumenta el contador hardlinks.
Al crear un softlink es necesario crear un nuevo inodo ya que se crea un archivo cuyo contenido es la ruta al archivo original.

5.

	Contiguo	**Indexado**
Información requerida para el acceso a los bloques del archivo	Se requiere conocer el primer bloque del archivo y el tamaño	Se requiere acceso a la tabla índice de los bloques y el tamaño
Fragmentación	Interna: Si. Externa: Si.	Interna: Si. Externa: No.
Limitaciones para la creación o el crecimiento del archivo	Para la creación debe encontrarse una porción de bloques contiguos y para el crecimiento debe haber bloques libres contiguos al último bloque del archivo	Para la creación y el crecimiento debe existir bloques libres y la tabla índice de bloque debe tener suficientes entradas para direccionarlos.

Práctica

1. .
    a. Dado que cada posición contiene el siguiente bloque del archivo, el bloque inicial es el 5 y revisando el contenido en la tabla se obtiene que los bloques del archivo son 5 -> 10 -> 6 -> 8 -> 3.
    b. Dado que es un archivo de 5 bloques de 8192 bytes cada uno y que para tener la mayor fragmentación interna posible se infiere que del último bloque sólo se usa 1 byte. Entonces son 4 bloques completos más 1 byte:
        i. (8192 bytes * 4) + 1 byte = 32769 bytes.
    c. El byte n° 12.000 se encuentra en el segundo bloque del archivo.
        i. 12000 / 8192 = 1,46 - se encuentra en bloque 10 (iniciando en 0).
    d. Dado que el resto de los bloques de la tabla están en cero y que en la tabla sólo hay dos “Fin de Archivo”, el resto de los bloques están en uso por el otro archivo. Son 2, 4 y 7. Los bloques 0,1,9 y 11 están libres.

e. El tamaño máximo teórico del file system es la cantidad máxima de bloques que puede soportar por el tamaño de bloque. En FAT 32:
   i. 2^28 * 2^13 = 2^41 = 2 TiB
f. El tamaño máximo real del file system es la cantidad de bloques por el tamaño de bloque. La tabla FAT es de 64 MiB y como se trata de un FAT32 cada entrada es de 4 bytes. Por lo tanto, la cantidad de entradas es el tamaño de la tabla dividido el tamaño del puntero (2^26 / 2^2 = 2^24). Tiene 16.777.216 entradas. Como cada entrada de la tabla hace referencia a un bloque 8192:
   i. Entonces el tamaño real del file system es 16.777.216 * 8192 = 137.438.953.472 bytes = 128 GiB

2. **1) A1012222h** -> **000A2222h** (PF no, AM 0 por éxito en tlb, si no contamos el paralelismo de ir a intentar conseguir la dirección física en caso no exitoso, frame número 10, tlb igual),
   2) 030110A1h -> **001510A1h** (PF si, AM 1 TP1, 1 TP2, 1 luego para buscar el dato?, número frame 21, tlb asumo que puedo sumar una nueva fila debajo para este proceso con el marco 0015 para la página 0301)
   3) 03030000h -> **000B0000h** (PF no, AM 1 TP1, 1 TP2, 1 luego para buscar el dato? número frame 11, tlb asumo que puedo sumar una nueva fila debajo para este proceso con el marco 000B para la página 0303)

b) Si se permitiera, ocurre una modificación de la segunda traducción lógica.

3. .

a) Estructuras actualizadas:

Info directorio: Propietario, grupo => U1, G1
Permisos directorio: rw-r--r--

Contenido directorio

Nombre	Inodo
a.txt	20
b.txt	22
c.txt	21
d.txt	23

Inode bitmap

20	1
21	1
22	1
23	1
24	1

Block bitmap

100	1
101	1
102	1
103	1
104	1

Inodo: c.txt
id: 21
tipo: regular
puntero: 103

Bloque 103
"hola"

Inodo: d.txt
id: 23
tipo: softlink
puntero: 104

Bloque 104
"/usr/share/c.txt"

b) No, porque U2 no es el propietario del directorio, y sus permisos son de lectura (ya sea que pertenece al grupo G1 o no). Para lograrlo el propietario debería cambiar los permisos dela archivo, y dar permisos de escritura sobre el grupo G1 (si es que U2 pertenece al mismo) o al resto.

c) La dirección tendría el formato: |nro seg|nro pag|offset|.

- Para referenciar hasta 15 volumenes, se necesitan 4 bits
- Una fragmentación máxima de 1023 bytes indica que las paginas son de 1kb (10 bits para representarlo)
- Para el numero de pagina, nos quedan: 17 bits total - 4 bits - 10 bits = 3 bits.

El formato final seria: 4 bits para el nro segmento | 3 bits para el nro de pagina | 10 bits para el offset.