Sistemas Operativos
1° Rec 2° Parcial 1C2024 - Resolución

<u>Aclaración</u>: 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.

	Fragmentación	Estructuras Administrativas	Overhead
Particionamiento Dinámico	Externa únicamente	1 Tabla de particiones y tabla de huecos libres	Muy alto si tiene compactación
Paginación Pura	Interna únicamente	1 Tabla de páginas por proceso	Medio
Segmentación Paginada	Sólo interna en la última página de cada segmento	1 Tabla de Segmentos por proceso + 1 Tabla de páginas por cada segmento del proceso	Medio alto (más que paginación pura)

2. a) Falso, el bit de modificado es siempre requerido cuando se tiene un sistema con memoria virtual, independientemente del algoritmo de reemplazo implementado, debido a que al momento de reemplazar una página en memoria, es necesario saber si esta fue modificada para ser escrita en disco.
   b) Verdadero, implementar memoria virtual habilita la posibilidad de cargar parcialmente los procesos en memoria, pudiendo ocupar ese espacio para cargar otros procesos, a diferencia de un sistema sin esta implementación que requiere cargar al proceso completo.

3. FAT tiene como estructura administrativa una única tabla FAT que contiene los punteros a los clusters de datos, pudiendo tener en cada entrada un flag representativo (libre, corrupto/error, fin del archivo) o el número de cluster al que hace referencia.
   El esquema de asignación de bloques es un enlazado-indexado, debido a que para recorrer un archivo se debe recorrer la tabla donde cada entrada indica dónde se encuentra el siguiente bloque del archivo.

4. El acceso directo tipo Hardlink, es un acceso directo que no puede referenciar archivos de otro filesystem debido a que para crear uno se agrega una entrada de directorio con el número de inodo al que se quiere referenciar, mientras que un softlink es un archivo nuevo, por lo que debe utilizar un inodo “nuevo” donde en su primer puntero directo se referencia a un bloque de datos que posee dentro la “ruta” o nombre completo del archivo al que se quiere referenciar. Teniendo esto en cuenta, no es posible implementar HL en FAT dado que no existen los inodos en este filesystem.

5. Entre las estructuras más conocidas existen:
    - Tabla o lista indexada de bloques libres
    - Lista enlazada de bloques libres
    - Bitmap de bloques libres

    Siendo que queremos optimizar el espacio en disco, la mejor opción es el bitmap, dado que requiere utilizar un bit por cada bloque, indicando de una manera ligera que bloques se encuentran ocupados, a diferencia de las otras estructuras que son por su naturaleza más pesadas.

Práctica

1)

1.
    a. Tam Max Teorico FS: 2^32 * 2^12 B = 2^44 B
    b. Tam Max Teorico Archivo: Punt x Bloque: 2^12B / 4B = 2^10
    (5 + 1*(2^10) + 1*(2^10)^2)*2^12B = (5 + 2^10 + 2^20)*2^12B ~= 2^32 B
2. 25123 B / 2^12 B = 6,13 bloques de datos => 7 BD. Como hay solo 5 punteros directos, necesitamos un bloque de indirección simple. El archivo ocupa en disco: 7 BD + 1 BP = 8 bloques (2^13 B).
3.
    a. /etc/so ---> config_so —— inodo X
    /home/Escritorio ---> link_to_config_so —— inodo Y (o X, pero de otro FS).
    b. Dos inodos.
    Uno contendrá el tamaño del archivo config_so (25123 B), tipo regular, inodo X, cinco punteros directos usados y el indirecto simple, entre otros atributos
    El segundo contendrá el tamaño del archivo link_to_config_so (tamaño de la ruta almacenada, unos 16 B aprox), tipo Soft Link (al ser particiones distintas no puedo referenciar por número de inodo), inodo Y (o si es el mismo ID, hay que aclarar que esta en otro FS), un puntero directo usado, entre otros atributos.

2)
Sabiendo que cada página pesa 256 bytes, entonces sabemos que necesitaremos 8 bits para el offset en la dirección lógica. A su vez, cada proceso tiene 4 segmentos, por lo que 2 bits serán utilizados para el número de segmento y quedarán 6 bits para el número de página, quedando la dirección lógica de la siguiente forma:

Segmento	Página	Offset
2 bits	6 bits	8 bits

A.

DL	Segmento	Página	Marco	Offset	DF
02ABh	0	2	7	ABh	07ABh
C110h	3	1	21	10h	1510h
8013h	2	0	9	13h	0913h
01FAh	0	1	18	FAh	12FAh

B. Tamaño máximo teórico de un proceso: 2^16 = 64KiB > 32KiB memoria física
Tamaño máximo real: 32KiB

C. Cómo sólo hay 6 marcos libres en la memoria y el proceso utiliza 9 marcos (de los cuales 3 son para el segmento de código y 6 para el resto de segmentos), sólo es posible cargar otra instancia del mismo programa si se utilizan mecanismos de compartición de páginas. Creando un nuevo proceso P2 que comparta las páginas pertenecientes al segmento 0 del proceso P1 ya que, por ser el segmento de código, nunca se modifican.

D. 255 B x 4 = 1020 Bytes

3)

a) La tabla FAT ocupa 64 MiB, siendo el file system FAT32, nuestros punteros ocupan 32 bits o 4 Bytes, por lo que disponemos de 2^24 entradas o 2^24 clusters de datos, dado que tenemos un volumen de 256 GiB, obtenemos que cada cluster tiene un tamaño de 16 KiB.
Finalmente, si dividimos el tamaño del archivo alojado en memoria por el tamaño de cluster obtenemos que son requeridos 64 clusters.

b) Como el archivo ocupa inicialmente 1 MiB en memoria, partido en frames de 8 KiB, representan 128 páginas presentes, por lo que al menos será necesario 128 accesos para traducir cada dirección lógica (obteniendo el frame donde se encuentra) y otros 128 accesos para leer la data de dichos frames.
Por otro lado, para persistir el archivo son necesarios 1 acceso para agregar la entrada de directorio para el archivo nuevo, 64 accesos a la tabla FAT en memoria y otros 64 eventuales accesos para escribir los clusters.
Resultando en:
   - 320 accesos a memoria (Traducciones + lectura frames + tabla FAT)
   - 65 accesos a disco (entrada dir + dados por la escritura final de los clusters)

c) No se perdió el archivo completo, solo los clusters mencionados, porque si están dañados no será posible leerlos. El resto del archivo se encuentra en el disco, pero se debe acudir a alguna herramienta o al respaldo de la FAT para encontrar la continuación del archivo guardado luego de cada uno de los tres clusters, dado que la tabla (si se actualizó) no indicará el siguiente cluster en esos casos sino un código de error en su lugar.
Ej:
122 — 78
123 — LIBRE
124 — ERROR
125 — 39
...