Sistemas Operativos
2° Rec 2do Parcial 2C2023 - 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. Sin MV, los procesos se deben de cargar completos en memoria, por lo cual nunca podría producirse trashing, en cambio, si tenemos MV y sustitución local, podría darse el caso de que el proceso quiera utilizar algunas páginas más que las que puede tener cargadas constantemente, produciendo continuamente trashing, un ejemplo podría ser: si puedo tener 4 páginas en memoria, pero el proceso accede a las mismas 6 páginas, van a haber 2 páginas que probablemente se estén quitando y que voy a requerir en unos instantes próximos.

2. Segmentación Pura:
   Formato DL: #seg | offset
   Estr. adm: 1 Tabla de Segmentos
   Accesos a MM: 2, lectura de la Tabla y el acceso a la DF
   Fragmentación: unicamente externa

   Segmentación Paginada:
   Formato DL: #seg | #pag | offset
   Estr. adm: 1 Tabla de segmentos + 1 Tabla de Páginas por segmento
   Accesos a MM: 3, lectura de la Tabla de Segmentos, lectura de TP, y el acceso a la DF
   Fragmentación: únicamente interna y en la última página de cada segmento

3. a) F, los hardlinks no pueden referenciar archivos de otros volúmenes a pesar de ser el mismo tipo, dado que no son un archivo nuevo, sino que son una entrada de directorio donde se referencia al archivo que se quiere linkear mediante su número de inodo, y cada volúmen tiene sus propios inodos.
   b) F, el bit de modificado es necesario siempre que estemos hablando de paginación (o alguna de sus variantes) con memoria virtual, dado que si una de estas páginas es modificada y eventualmente será reemplazada por otra, antes debe de ser actualizada su información en disco, sin importar el tipo de algoritmo de sustitución utilizado.

4. Un lock es una herramienta del sistema operativo para tener mutua exclusión sobre archivos de forma más avanzada que usando semáforos. Los locks sugeridos deben ser ubicados correctamente por todos los procesos para que se garantice la mutua exclusión mientras que los obligatorios garantizan al proceso que tenga el lock tomado la mutua exclusión aún si el resto de los procesos no piden el lock.

5. El esquema utilizado en FAT es una variante del esquema de asignación de bloques enlazada, en la cual el puntero al siguiente bloque se encuentra al final del propio bloque, mientras que en FAT, ese puntero se encuentra en la TABLA FAT del filesystem y no el los bloques de datos, de esta manera facilita el acceso a clusters directamente, ya que no deberá leer todos los clusters para llegar al que requiere, sino que leerá las entradas que se indiquen en la tabla y una vez llegado al cluster requerido, se leerá dicho bloque, evitando lecturas innecesarias.

Práctica

1. Dado que se quiere direccionar la totalidad del disco, sabemos que el tamaño máximo direccionable nos tiene que dar el tamaño del disco:
   256 GiB = Tam.Puntero * Tam.Cluster
   Como queremos aprovechar lo más posible el almacenamiento, lo más conveniente es utilizar la mayor cantidad de punteros para tratar de disminuir el tamaño de clusters, de manera tal que con las versiones de FAT presentadas obtenemos:

   FAT32: 256 GiB = 2^38 = 2^28 * 2^10 -> Utilizando los 28 bits disponibles para punteros obtengo clusters de 1 KiB.

   FAT16: 256 GiB = 2^38 = 2^16 * 2^22 -> Utilizando los 16 bits disponibles para los punteros requiero clusters de 4MiB, lo cual en caso de tener archivos pequeños (hablando dentro de los KiB o Bytes), podría producirse mucha fragmentación interna, desperdiciando espacio de almacenamiento.

2. Por tener una memoria física de 32 MiB, la cual está dividida en 8192 frames, podemos concluir que el tamaño de cada frame es de 2^12 = 4 KiB, por ser paginación las páginas serán del mismo tamaño.

   Las referencias propuestas se traducen a las siguientes páginas:
   8100 E -> #Pag 1 9000 L -> #Pag 2 40990 E -> #Pag 10 3500 L -> #Pag 0

a) Corriendo el algoritmo LRU con últimas referencias siendo las páginas 17 - 19 - 10, siendo la página 17 la que hace más tiempo NO se referencia, y la 10 siendo recientemente usada:

Marco	Pagina	1 E	2 L	10 E	0 L
2	10 M	10 M	10 M	10 M	10
6	**17**	1 M	1 M	**1 M**	0
8	19	**19**	2	2	2
		PF	PF		PF

En esta traza se reemplazó la página 1 que fue escrita, por lo cual representa un acceso para bajarlas a disco además del PF.

b) Corriendo el algoritmo Clock Modificado:

Marco	Pagina	1 E	2 L	10 E	0 L
2	10 UM	10 UM	-> 10 M	-> 10 UM	10 M
6	**-> 17**	1 UM	1 M	**1 M**	0 M
8	19 U	**-> 19 U**	2 U	2 U	-> 2 U
		PF	PF		PF

En esta traza se reemplazó la página 1 que fue escrita, por lo cual representa un acceso para bajarlas a disco además del PF.

3. Inodo: 6 PD + 1 IS + 1 ID, los punteros son de 4 bytes y los bloques de 1 KiB
   Punteros x Bloque = 2^10 / 2^2 = 256

a)

Tamaño máx.Teórico del FS = Cantidad Punteros * Tam.Bloque=2^32 * 2^10=2^42=4TiB
Tamaño máx.Teórico de un Archivo = (6+256+256^2) * 1024 bytes = 64 MiB (aprox)
Si las direcciones son de 16 bits, un proceso podría ocupar 64 KiB

b)

Lecturas posibles de bloques de datos con la configuración de inodo:
6 PD -> #0 - #5
1 IS -> #6 - #261
1 ID -> #262 - #65797

El byte 5600 se encuentra en el BD #5
El byte 10720 se encuentra en el BD #10
Por lo que he de leer 6 bloques de datos (bloque 5,6,7,8,9 y 10)

El primer bloque de datos puedo leerlo con el ultimo PD, por lo que para los siguientes 5 bloques de datos se requiere utilizar un puntero indirecto:

Accesos a bloques = 6 BD + 1 BP[is] = 7 Bloques

c)

Dado que tenemos direcciones de 16 bits, y hasta un máximo de 16 paginas por proceso, significa que para indicar el #pág se requieren 4 bits, quedando los 12 restantes para el offset, concluyendo que las páginas son de 4 KiB.
Como el archivo ocupa 32 MiB, son en total 2^15 bloques de datos, en cada página del proceso entran 4 bloques de datos, por lo tanto: 16 * 4 = 64 bloques que pueden cargarse en memoria.