Sistemas Operativos
2° Parcial 1C2022 – TT – 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. Una ventaja podría ser que permite reubicar procesos en memoria ram. Otra podría ser que permite independizarse del tamaño de la memoria ram al momento de compilar el programa. Por otro lado, un proceso no puede conocer de manera directa la dirección física correspondiente a una dirección lógica, debido a que implicaría poder acceder a las tablas de páginas, que se encuentran en espacio de kernel.
2. Formato: x bits para el nro de frame (falta info de la ram), 1 bit de presencia, 1 bit de uso, 1 bit de modificado
3. V o F
    a. La utilización de Memoria Virtual requiere que las direcciones se traduzcan en tiempo de ejecución.
        i. Verdadero, La traducción en tiempo de ejecución permite, que luego de un page fault, una página ubicada en memoria virtual pueda asignarse a cualquier marco libre.
    b. En la paginación jerárquica o por niveles, la existencia de una TLB puede reducir la cantidad de page faults.
        i. Verdadero. La función de la TLB es facilitar la traducción de la dirección lógica a dirección física. Cuando se utiliza paginación jerárquica y ante una fallo de TLB, la traducción de la página puede implicar acceder a porciones de la tabla de páginas que no se encuentran en memoria, generando de esta manera un page fault.
4. Bitmap de bloques (1 bit por cada bloque, indicando si está libre u ocupado), lista de bloques libres (una estructura separada con nodos apuntando entre sí, conteniendo el número de bloques libres). Por otro lado, se podría reconstruir la tabla recorriendo todos los FCBs de todos los archivos, para relevar los bloques que se encuentran ocupados, y luego por diferencia concluir cuales están libres.
5. Si, podría ocurrir en un filesystem como ext2 si la ruta del archivo borrada se correspondiera con un hardlink, y dicho link no fuera el único apuntando al archivo. Por otro lado, la syscall seek() es un ejemplo de un caso que no implica acceso al disco.

.

Práctica

1. FS

a) Si, se puede copiar.
Se necesitarán: 16 MiB/1 KiB = 2^14 bloques = 16384 bloques de datos en total:
- **10 directos** = + 10 bloques de datos (total 10 bloques)
- 1 KiB / 8 bytes = 2^7 punteros por bloque = **1 bloque de punteros (indirección simple)** + **128 bloques de datos (total 138 bloques)**
- **1 bloque de punteros (indirección doble) + 16384 bloques de datos (total 16522 bloques)**
Total direccionable por archivo: **16522 bloques (para un archivo de 16384, alcanza)**

Cantidad de bloques necesarios de datos: 16384
Cantidad de bloques necesarios de punteros:
- 1 por indirección simple (10 bloques)
- 1 por bloque 1er nivel en indireccion doble
- 127 por bloques 2do nivel en indireccion doble:
    - Bloques restantes para direccionar: 16384 - 10 - 128 = 16246. Para direccionarlos, necesito 16246/128 = 126,.. => 127 bloques de punteros.

Respuesta: 16384 bloques de datos + (1+1+127) bloques de punteros + <u>16513 bloques en total</u>

b) Byte 10000 está en el bloque = 10000 B/1 KiB = 9,76… -> último bloque puntero directo
Byte 400000 está en el bloque = 40000 B/1 KiB = 390,62… -> apuntado por segundo (390 - 10 directos - 128 ind.simple - 128 1er bloque ind. doble - 128 2do bloque ind. doble) bloque de punteros de indirección doble.
Total bloques de datos a leer: 382 (considerar que hay que leer el primero y el último. Es la resta 391 - 10, más uno)
Entonces:
1 bloque de datos
+ 1 bloque indirección simple + 128 bloques de datos
+ 1 bloque de indirección doble
+ 2 bloques indirección simple (del doble)
+ 128 bloques de datos
+ 125 bloques de datos
= 386 bloques en total

c) Se debería usar un soft link ya que permite referenciar a un archivo a través de distintos FS. No se podría usar un hardlink porque los id de los inodos podrían repetirse y/o diferir entre ambos.

2.
a)
20CABh no generó Page Fault.

Se sabe que las páginas son de 4KiB por lo que CABh corresponde al offset.
Tenemos que averiguar que parte de 20h pertenece al número de segmento y que parte pertenece al número de página.
Algo a tener en cuenta es que por más que no se pueda ver el resto de las entradas de las tablas de páginas ni las otras tablas de páginas si es que hay más segmentos, sabemos que todas las páginas del proceso que están presente en memoria se encuentran en ese fragmento porque la asignación es fija de 6 frames y hay 6 páginas con el bit de presente encendido.
Por este motivo, la única forma válida de partir la dirección es con 3 bits para el segmento y 5 bits para el número de página ya que el la página 0 del segmento 1 se encuentra en memoria principal y las otras combinaciones posibles no.

Tomando eso en cuenta, podemos evaluar que sucede con cada referencia:

	Inicial			02FBEh -> S0 P2			60ABCh -> S3 P0			01EF1h -> S0 P1			82ABBh -> S4 P2		
Frame	S	P	U	S	P	U	S	P	U	S	P	U	S	P	U
0	1	0	1	1	0	1	1	0	1	1	0	1	1	0	0
1	2	2	1	**2**	2	1	**2**	2	1	2	2	0	4	2	1
3	0	0	1	0	0	1	0	0	1	0	0	0	**0**	0	0
5	3	2	0	3	2	0	3	2	0	0	1	1	0	1	1
17	0	2	0	0	2	1	0	2	1	**0**	2	1	0	2	0
20	3	0	0	3	0	0	3	0	1	3	0	1	3	0	0

b)
Si bien en memoria principal el proceso podría tener 8 segmentos, lo cual implicaría que la máxima fragmentación interna sería (4KiB - 1b) * 8, podemos considerar que estamos limitados a 6 por la cantidad de frames

3.
Es necesario averiguar a cual pagina se desea acceder, para ello hay que averiguar cuántos bits indican el número de página, para la dirección 0AC6h (en binario 101011000110)
Como la fragmentación máxima es 1023 bytes, entonces los clusters son de 1024 bytes, y por lo tanto las páginas son del mismo tamaño. Por lo tanto, se necesitan 10 bits para expresar el offset de una página, lo cual determina que los bits restantes indican la página, por lo tanto se desea acceder a la página 2 (10 en binario).
Como la página 2 no se encuentra en memoria, será necesario acceder al 3er bloque del archivo proceso1.swp, el cual contiene dicha página

Listado de accesos:

1- 1 acceso a TP (fallo de página) (Memoria, lectura)
2- 1 acceso al contenido del directorio para ubicar el 1er cluster del archivo (Memoria, lectura)
3- 2 accesos a la FAT para encontrar el 3er cluster, nro 22 (Memoria, lectura)
4- 1 acceso a disco para leer el cluster deseado (Disco, lectura)
5- 1 acceso a TP para actualizar la tabla con la ubicación de la página (Memoria, escritura)
6- 1 acceso a TP para leer la ubicación de la página 2 (Memoria, lectura)
7- 1 acceso a memoria para leer la página 2 (Memoria, lectura)

.