Sistemas Operativos
2° Parcial 2C2022 - TM - 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 interna: Aumentaría.
   Tamaño de las tablas de páginas: Se reduciría ya que se usarían más bits para el offset.
   Aprovechamiento de la TLB: Se aprovecharía más ya que sería más probable tener TLB hits.
2. No se podría descartar, aun teniendo unos pocos procesos en thrashing haciendo swapping de páginas constantemente, el sistema podría estar usando su CPU al máximo debido a la ejecución normal del resto de los procesos.
3. a) Verdadero. El bit de "modificado" siempre es necesario para, a la hora de sustituir una página, sea cual sea la que haya seleccionado como víctima el algoritmo, saber si debemos escribir la misma en swap antes de reemplazarla por la nueva.
   b) Falso. No es posible crear hardlinks a otros filesystems.
4. Existen dos tipos de tabla: una global, donde se incluye información sobre los archivos abiertos por cualquier proceso y campos generales (como el contador de aperturas e información del archivo en memoria) y a su vez, una tabla por proceso, donde se guarda un puntero a la tabla global y campos por cada proceso (como el puntero de lectura y el modo de apertura).
   Cuando un archivo se abre por primera vez, es escrito en la tabla global y en la del proceso. Cuando el mismo archivo se abre por otro proceso, se escribe sólo en la del proceso y se actualiza el contador de aperturas en la global.
5.

	Continua	Enlazada	Indexada
Fragmentacion	Sufre fragmentación interna (en el último bloque) y externa.	Sufre fragmentación interna (en el último bloque).	Sufre fragmentación interna (en el último bloque y en el bloque índice).
Recuperabilidad	Siempre es posible recuperar el resto del archivo.	No es posible recuperar los bloques siguientes al dañado.	Siempre es posible recuperar el resto del archivo salvo que se dañe el bloque índice.

Práctica

1. Los archivos tienen que soportar 16 GiB y buscamos aquel tamaño de bloque con menor desperdicio.
   Llamando al tamaño de bloque como “BL”, sabemos entonces que:
   Ptrs_x_blq = BL/4
   12 BL + BL x BL/4 + BL x (BL/4)ˆ2 + BL x (BL/4)ˆ3 >= 16 GiB

   También sabemos que el término más significativo en el tamaño del archivo es "BL x (BL/4)ˆ3". Por ende podemos calcular:
   BL x (BL/4)ˆ3 = 16GiB
   BLˆ4 / 2ˆ6 = 2 ˆ34 → BLˆ4 = 2ˆ40 → BL = 2ˆ10 → **1 KiB**

   a) Un bloque más chico que 1 KB no permitiría archivos de 16GB. Un bloque más grande generaría mayor fragmentación.
   b) 1 KB x 2 ˆ32 = 2 ˆ42 → **4 TiB**
   c) Tomando bloques de 1 KiB, tendríamos 256 punteros por bloque y necesitaríamos leer 300 bloques de datos. Por lo tanto, no nos alcanza con los punteros directos ni con el indirecto simple (12 + 256 = 268, me faltarían direccionar 32 bloques). Por lo tanto necesito también acceder al bloque indirecto doble y un indirecto simple más, entonces:
   300 bloques de datos + 2 IS + 1 ID = **303** bloques

2. a) Al tener memoria virtual, el tamaño de un proceso solamente se ve limitado por el direccionamiento lógico, por lo tanto: 2ˆ32 = **4GiB**
   b) Con páginas de 1MiB, la dirección lógica se divide en 20 bits para offset y 12 bits para número de página, por lo tanto:
   - DL: 01319B10h => página 013h, offset 19B10h => **página 19**
     Está presente en TLB y corresponde al **marco 20**, por lo tanto no hay PF ni acceso a tabla de páginas.
     DF: marco 014h, offset 19B10h => **01419B10h** para ambos algoritmos ya que no hay reemplazo.
   - DL: 01222ABCh => página 012h, offset 22ABCh => **página 18**
     No está presente, por lo tanto hay PF y acceso a tabla de páginas.
     LRU: reemplazaría página 0 (**marco 7**) ya que es la que hace más tiempo que no se accede. DF: marco 007h, offset 22ABCh => **00722ABCh**
     Clock Modificado: reemplazaría página 20 (**marco 5**) ya que es la única con M=0 y todas tienen U=1. DF: marco 005h, offset 22ABCh => **00522ABCh**

3. Teniendo direcciones de 16 bits, si los procesos utilizan 16 segmentos como máximo y sus tablas de páginas son de 16 entradas, necesitaría 4 bits para número de segmento y 4 bits para número de página, por lo que me quedarían 8 bits de offset. La dirección lógica quedaría entonces compuesta como:
   [4 bits (seg) | 4 bits (pag) | 8 bits (offset)]

a) Podría como máximo tener fragmentación interna en la última página de cada segmento del proceso, entonces:
   Frag.int.max = (tam_pagina - 1) * cant_max_segmentos
   = 255 B * 16 = **4KiB - 16 B**
   Podría como máximo tener fragmentación interna en el último cluster del archivo:
   Frag.int.max = (tam_cluster - 1 B) = **8KiB - 1 B**
b) Tamaño máximo proceso (sin memoria virtual)
   = min(dir_lógico, dir_físico)
   = min(64KiB, 64KiB) = **64KiB**
   Tamaño máximo archivo (en FAT) = Tamaño del FS
   = cant_clusters * tam_cluster
   = 2^16 * 8KiB
   = 2^16 * 2^13B = 2^29B = **512MiB**

c) La dirección física estaría compuesta como: [8 bits (frame) | 8 bits (offset)]
   0511h: frame 5 => segmento 1, página 0 => **1011h**
   1F33h: frame 31 => segmento 2, página 1 => **2133h**
   2011h: frame 32 => segmento 3, página 2 => **3211h**
   FF44h: frame 255 => segmento 0, página 2 => **0244h**

d) Tamaño FAT = cant_entradas * tam_entrada = 2^16 * 2^1 B = 2^17 B = **128 KiB**