Sistemas Operativos
2° Parcial 2C2023 - TM - 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. Fragmentación interna: En promedio aumentaría ya que sería más probable que sobre espacio en la última página.
   Tamaño de las tablas de páginas: Serían más chicas ya que se requerirían menos páginas para direccionar el mismo espacio.
   Aprovechamiento de la TLB: Se aprovecharía más ya que sería más probable que se acceda repetidas veces a las mismas páginas.

2. a) Verdadero, por defecto ejecutar una instrucción puede generar más de un page fault si la misma requiriera acceder a más de una página, por otro lado en paginación jerárquica pueden generarse page faults no solo por la página sino también por alguna de sus tablas si la misma no se encuentra en memoria.
   b) Falso, la TLB nos ahorra tiempo de acceso a memoria debido a que se guarda traducciones hechas previamente que ya tuvieron su page fault. Podría responderse verdadero si consideramos que nos podríamos ahorrar el page fault de una tabla que no está en memoria pero la página a la que queremos acceder si lo está.
3. .
4. .
5. Para que la lectura de archivos de hasta 13500 bytes sea lo más rápida posible deberíamos direccionarlos completamente con punteros directos.
   12500 / 1024 = 12,....
   Necesitamos al menos 13 punteros directos.

   Para que soporte archivos de 30 GiB calculamos primero cuánto nos direcciona un puntero indirecto triple.
   1024 / 4 = 256 = 2^8 punteros por bloque => (2^8)^3 * 2^10 B = 2^34 B = 16 GiB
   Con dos punteros indirectos triples podríamos direccionar 32 GiB (más que lo pedido) y tendríamos justo 15 punteros en el inodo.

Práctica

1. a) Primero identifico cuantos bits para #Seg/#Pag/#Offset. Como tengo páginas de 4KiB son 12 bits para offset y el resto para #Seg/#Pag. Como la referencia 2A023h no genera PF, hace referencia al Segmento 5 Página 2, por lo que los primeros 5 bits son para segmento y los 3 restantes para página.

Segmento	Pagina	Offset
5 bits	3 bits	12 bits

Con el algoritmo clock modificado:

0BFFBh (S1 P3) genera la DF CFFBh

11DDAh (S2 P1) genera la DF FDDAh

29EEAh (S5 P1) genera la DF 1EEAh

0C001h (S1 P4) genera la DF 6001h

	Inicial		0BFFB (Lectura)		11DDA (Escritura)		29EEA (Escritura)		0C001 (Lectura)	
	S|P	bit	S|P	bit	S|P	bit	S|P	bit	S|P	bit
1	s5p1	U	s5p1	U	>s5p1	U	>s5p1	UM	s5p1	M
3	s1p1	UM	s1p1	UM	s1p1	UM	s1p1	UM	s1p1	M
6	s5p2	M	s5p2	M	s5p2	M	s5p2	M	s1p4	U
C	>s1p3	M	>s1p3	UM	s1p3	M	s1p3	M	>s1p3	M
F	s2p2	M	s2p2	M	s2p1	UM	s2p1	UM	s2p1	UM
PF					X				X	
Acc disco					X				XX	
TLB			X		XX		X		XX	

b) Estado final de las páginas y la TLB:

S1	
Frame	Bits
F	-
3	M,P
C	-
C	M,P
3	U,P

S2	
Frame	Bits
6	-
F	U,M,P
F	-
1	-
C	-

S5	
Frame	Bits
6	-
1	M,P
6	-
3	-
F	-

TLB	
pag	frame
0C	6
29	1
11	F
OB	C

C) No sería posible ya que con asignación fija de marcos no se puede implementar reemplazo global. Si lo intentara y eligiese como víctima una página de otro proceso, le estaría dando más frames a un proceso de los inicialmente designados, por lo que no cumpliría con la asignación fija.

2.

a) Tam teórico del FS = 2^16 * 1 KiB
= 2^16 * 2^10 B = 2^26 B = 64 MiB
Limitado por el disco de 16 MiB => Tamaño real del FS = **16 MiB**

b) 16 MiB = cant_entradas * 1 KiB
2^24 B = cant_entradas * 2^10 B => cant_entradas = 2^14
Tam FAT = 2^14 * 2B = 2^15 B = **32KiB**

c) Siguiendo la tabla FAT:
"notas1.txt" tiene los bloques **7, 1, 8 y 9**.
"export.sh" tiene los bloques **10, 11, y 4**.

d) 3500/1024 = 3,....
Necesito acceder al bloque 3 del archivo (su cuarto bloque), por lo que mínimamente tendré que acceder 3 veces a la FAT para averiguar el número de bloque y luego 1 acceso a ese bloque.

3.

a) Tam máximo FS = 2^32 * 2^12 B = 2^44 B = **16 TiB**
Tam máximo proceso = 2^32 B = **4 GiB**

Punteros por bloque = 4 KiB / 4 B = 1024 = 2^10
Tam máximo archivo =- (2^10)^3 * 2^12 B = 2^30 * 2^12 B = 2^42 B = **4 TiB**
Nota: Aproximamos el tamaño máximo de archivo solamente con el puntero indirecto triple:

b) En Ext2, debido a la indexación multinivel de los punteros indirectos, se sufre fragmentación interna no solo en el último bloque de datos del archivo, sino también de los últimos bloques de punteros de cada nivel.
Un proceso en este esquema solamente sufre fragmentación interna en la última página.

c) 1500/1024 = 1,....
El bloque 1500 no es direccionado a través de los punteros directos (0-11) ni por el indirecto simple (los 1024 que siguen), por lo tanto se requiere acceder a 2 bloques de punteros (a través del puntero indirecto doble) y luego al bloque de datos.
Debido a tener paginación en 2 niveles, para traducir la DL a DF debemos acceder a 2 tablas de páginas (en memoria) para luego escribir el marco correspondiente.