Sistemas Operativos
2° Parcial 1C2024 - 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. Podría lograrse ya que cada proceso tiene su propio direccionamiento lógico, por lo tanto, aunque un mismo proceso no pueda ocupar toda la memoria, sí podría ocuparse por completo con un grado de multiprogramación lo suficientemente elevado.
   La capacidad de reubicar procesos se da debido justamente a traducir direcciones lógicas en físicas, por lo que un proceso ya compilado y con direcciones lógicas ya definidas puede ser movido en la memoria cambiando las estructuras de traducción (lista de particiones, tabla de segmentos, tabla de páginas, etc.)

2. Para detectar si existe thrashing en este sistema, dado que la sustitución es local, no serviría medir el uso de CPU ni la frecuencia de PFs en general, sino que debemos medir la frecuencia de PFs de cada proceso. Una vez detectado, puede solucionarse asignándole más marcos a los procesos que estén en ese estado.
   Si no quedaran marcos libres, podría considerarse liberar algunos marcos reduciendo la cantidad de marcos asignados a otros procesos que tengan una frecuencia de PFs muy baja, o bien reducir el grado de multiprogramación (por ejemplo, suspendiendo algún proceso).

3.

	Espacio	Granularidad
Unix	Poco	Baja
Matriz	Mucho	Alta

Los permisos deben estar ubicados en el FCB, dado que su estuvieran en la entrada de directorio podría ocurrir un conflicto si existieran varias entradas apuntando al mismo FCB.

4. Extended implementa una asignación de bloques indexada combinada. El esquema combinado utiliza tanto punteros directos a bloques de datos como indirectos a bloques de índices.
   Esta combinación permite que el esquema se adapte perfectamente a los diferentes tamaños posibles de un archivo:

- Los archivos pequeños no necesitan un índice separado, por lo que se reducirá el espacio desperdiciado en bloques de índice que, para este tipo de archivos, puede ser grande si se lo evalúa de forma proporcionada a la cantidad de bloques de datos que contienen.
- Se pueden aún así referenciar a muchos bloques de datos, por lo que no se limita el tamaño del archivo, permitiendo tener archivos muy grandes.

5. A. Falso. No se pueden crear hardlinks sobre archivos de otro file system.
   B. Falso. Esta afirmación sería verdadera en un esquema de asignación enlazada pura. FAT, por otro lado, utiliza una tabla que almacena los punteros que en el otro caso serían almacenados en los bloques de datos en disco. Esta tabla se encuentra cargada en memoria, por lo que si bien realizar un acceso directo al último bloque de un archivo sí requiere leer los punteros de todos los bloques que lo preceden, estas lecturas son sobre memoria y no generan un impacto tan grande como realizar los accesos a disco.

Práctica

1. a)
   Para saber a qué bloques del FS debemos acceder, necesitamos averiguar a partir de qué bloque del archivo se quiere leer y cuántos.
   Estando el puntero del archivo situado en el byte 8000 y ser tamaño de bloque 8 KiB (8192 bytes) podemos deducir que debemos leer desde el 1er bloque del archivo, el cual corresponde al bloque 15 del FS.
   Luego, se quieren leer 10 KiB (10240 bytes) por lo que estaremos leyendo 192 bytes del 1er bloque, el 2do bloque completo (8192 bytes) y el resto (1856 bytes) del 3er bloque.
   Por lo tanto, se leerán los bloques 15, 16 y 17.

   Para saber la cantidad de marcos es una lógica similar.
   Sabiendo que se escribe desde la DL 8000h, corresponde a: [página 4 | offset 0].
   Por lo que se escribirán solamente 2 marcos, el correspondiente a la página 4 completo (8192 bytes) y el resto en el marco correspondiente a la página 5.

   b)
   i) 64 KiB = 8 bloques, si el FS tiene 10 bloques libres la operación se puede realizar, sin embargo requerirá compactar previamente si el archivo no tiene al menos 8 bloques libres contiguos a su último bloque.
   ii) La respuesta dependerá de si el proceso B abre el archivo en modo solo lectura o no. Si lo abre en modo solo lectura podrá operar inmediatamente, de lo contrario tendrá que esperar a que el proceso A lo cierre.

2. El máximo espacio teórico direccionable de un file system está dado por:
(Cantidad de bloques direccionados por el puntero) . (Tamaño de bloque)
2^(Tamaño puntero) . (2^10) = 2^42
2^(Tamaño puntero) = 2^42/2^10
Tamaño de puntero = 32 bits

a) La configuración del inodo no permite direccionar un archivo tan grande:
Cantidad punteros por bloque = 2^10/2^2 = 2^8
El inodo puede direccionar archivos de:
12 . 2^10 + 2^8 . 2^10 + (2^8)^2 . 2^10 + (2^8)^3 . 2^10
12 KiB + 256 KiB + 64 MiB + 16 GiB
Un poco más de 16 GiB…

Para resolver el problema podríamos:
- Añadir un puntero cuádruple para direccionar archivos de hasta un poco más de (2^8)^4 . 2^10 = 2^42 = 4 TiB
- Aumentar el tamaño del bloque

b) En FAT:
Tamaño máx teórico FS = Tamaño máx teórico archivo = 2^28 . 2^10 = 2^38 = 256 GiB > 160 GiB
Como el disco es de 4 TiB, el FS en FAT tendría un tamaño máximo real de 256 GiB.
Podría almacenar el archivo, pero no podría aprovechar al máximo su disco de 4 TiB.

c) 530 KiB = 530 bloques
Contamos:
- 12 bloques de datos por puntero directo (quedan 518 bd por leer)
- 2^8 = 256 bloques de datos por índice de 1 nivel + bloque índice (quedan 262 bd por leer)
- 256 bloques de datos por el primer bloque de índices del índice de 2 niveles + 2 bloques de índice (quedan 6 bd por leer)
- 6 bloques de datos por el segundo bloque de índice del índice de 2 niveles + 1 bloque de índice

Sumamos:
12 BD + 256 BD + 1 BI + 256 BD + 2 BI + 6 BD + 1 BI
530 BD + 4 BI
534 bloques

3. a) 2^{18} = 262144 bytes = 256 KB

   256/16 = páginas de 16 KB = 16384 bytes → 2^{14} → 14 bits de offset → (18-14) 4 bits para pagina en la dirección física: #Frame (4 bits) | #Offset (14 bits)

   Las direcciones lógicas son de 20 bits -> #Pag (6 bits) | #Offset (14 bits)

PB - 2C010h: TLB **MISS**, se **accede a TP** y se encuentra la página, no hay PF. **DF**: 04010h

PA				
pag	p	f	acc	carga
...				
5	1	5	65	10
7	1	12	75	75
...				
10	0	2	15	15
11	1	2	80	50
...				
13	1	8	20	20

PB				
pag	p	f	acc	carga
...				
3	1	9	48	35
...				
11	1	1	85	30
...				
15	1	7	60	60

TLB			
pag	f	proceso	l
11	2	A	50
7	12	A	75
11	1	B	85
15	7	B	60

PA - 2F022h: TLB **HIT**, la TP solo se actualiza y no hay PF. **DF**: 0B022h

PA				
pag	p	f	acc	carga
...				
5	1	5	65	10
7	1	12	75	75
...				
10	0	2	15	15
11	1	2	90	50
...				
13	1	8	20	20

PB				
pag	p	f	acc	carga
...				
3	1	9	48	35
...				
11	1	1	85	30
...				
15	1	7	60	60

TLB			
pag	f	proceso	l
11	2	A	50
7	12	A	75
11	1	B	85
15	7	B	60

PA - 19333h: TLB **MISS**, se **accede a TP** y no se encuentra la página, **hay PF**. **DF**: 21333h

PA				
pag	p	f	acc	carga
...				
5	1	5	65	10
6	1	8	95	95
7	1	12	75	75
...				
10	0	2	15	15
11	1	2	90	50

PB				
pag	p	f	acc	carga
...				
3	1	9	48	35
...				
11	1	1	85	30
...				
15	1	7	60	60

TLB			
pag	f	proceso	l
6	8	A	95
7	12	A	75
11	1	B	85
15	7	B	60

...				
13	0	8	20	20

PB - 1E041h: TLB **MISS**, se **accede a TP** y no se encuentra la página, **hay PF**. **DF**: 0E041h

PA				
pag	p	f	acc	carga
...				
5	1	5	65	10
6	1	8	90	90
7	1	12	75	75
...				
10	0	2	15	15
11	1	2	90	50
...				
13	0	8	20	20

PB				
pag	p	f	acc	carga
...				
3	1	9	48	35
...				
7	1	3	100	100
...				
11	1	1	85	30
...				
15	1	7	60	60

TLB			
pag	f	proceso	l
6	8	A	95
7	12	A	75
11	1	B	85
7	3	B	100

PA - 30000h: TLB **MISS**, se **accede a TP** y no se encuentra la página, **hay PF**. **DF**: 14000h

PA				
pag	p	f	acc	carga
...				
5	0	5	65	10
6	1	8	90	90
7	1	12	75	75
...				
10	0	2	15	15
11	1	2	90	50
12	1	5	105	105
13	0	8	20	20

PB				
pag	p	f	acc	carga
...				
3	1	9	48	35
...				
7	1	3	100	100
...				
11	1	1	85	30
...				
15	1	7	60	60

TLB			
pag	f	proceso	l
6	8	A	95
12	5	A	105
11	1	B	85
7	3	B	100

b) Al tener memoria virtual, el tamaño máximo de cada proceso no está limitado por la memoria real, sino por el direccionamiento lógico.
Por lo tanto, teniendo 20 bits de direccionamiento lógico -> tam máx proceso = 2^{20} = 1 MiB

La máxima fragmentación interna que puede sufrir un proceso es:
Max fragmentación = Tam página - 1B = 16 KiB - 1B = 16384 B - 1 B = 16383 B