# Sistemas Operativos
# 2° Parcial 2C2018 - 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. En ambos casos borra la entrada de directorio
   Si es un hardlink, tiene que decrementar el valor de contador de referencias. Si llega a 0, tiene que liberar el inodo y los bloques asignados. Si es un softlink tiene que liberar el inodo y los bloques en caso de tener (si no alcanza con el inodo). En este último caso, el archivo original al que apunta se mantiene sin cambios.
   Como extra, tambsién debería actualizar el superbloque con la cantidad de inodos/bloques libres.
2. a) 1ra opción: Si se requiere minimizar el overhead al calcular direcciones físicas entonces convendría utilizar un esquema de asignación contigua. Si aparte nos piden que no tenga fragmentación interna entonces deberíamos plantear particiones de tamaño variable.
   2da opción: Si se requiere minimizar el overhead para calcular direcciones físicas también se podría utilizar un esquema de segmentación, dado que las direcciones se calculan, al igual que en asignación contigua, con base y offset. Por otro lado también cumple la condición de que no produce fragmentación interna por tener tamaño variable.
   b) Los esquemas de asignación contigua son los más sencillos a la hora de realizar la traducción. Dentro de los mismos, la estrategia que minimiza ambos tipos de fragmentación es buddy system, al tratar de acercar el tamaño de la partición a la potencia de 2 más cercana
3. La tlb es una memoria de tipo cache, la cual nos permite almacenar las entradas de las últimas páginas accedidas, pudiendo conocer el número de página y en qué marco se encuentra.
   La utilidad es poder, de forma rápida, conocer en qué marco se encuentra alguna página para agilizar la traducción de DL a DF.
   Se lee cuando se quiere acceder a una página, en el momento de realizar la traducción, , se escribe cuando se actualiza una entrada de la tabla de páginas (por ejemplo se carga una nueva página en memoria, se desaloja una página o se finaliza un proceso)  de algún proceso o cuando la tabla no tiene pid y se cambia de proceso.

Es importante porque ayuda a evitar accesos a memoria para leer las TP jerárquicas, aparte que si alguna no está en RAM la tengo que traer de disco. Por lo tanto, minimiza el tiempo de acceso efectivo a memoria.

4. En FAT considerando que siempre se obtiene el bloque encontrando primero su posición en la FAT en memoria y se lo va a buscar a disco directamente, entonces la única posibilidad de que en UFS tenga la misma cantidad de accesos (1) es que el bloque de datos deseado se encuentre direccionado por alguno de los punteros directos del inodo del archivo.
   Para que la cantidad de accesos en UFS sea distinta se podría plantear que el bloque de datos buscado este dentro de algún nivel de indirección, teniendo primero que buscar el bloque de punteros, para luego buscar el bloque de datos.
5. V o F
   a. F. Debería ser sincrónica. En caso de ser asincrónica justamente la idea es mientras aprovechar el tiempo para realizar otra operación.
   b. V. Al separar los pedidos en distintas colas (2 o varias) separan los pedidos en atención de los nuevos que llegan, evitando la potencial inanición.

# Practica

1.

a) Los pedidos iniciales son de las pistas: 3-17-9-69-72-85 y los que llegan en t=16s son de las pistas: 66 y 70.

Como inicialmente está en la 51 subiendo y se usa C LOOK el orden de pedidos será:
51-69-70-72-85-3-9-17-66
Tiempo total:
Pista 51 a pista 85: 34ms
Pista 85 a pista 3: 0ms (despreciable)
Pista 3 a pista 66: 63ms
Total: 97ms

b) Si el disco tiene 10 sectores por pista entonces los pedidos son de: 4-23-12-96-101-119 y los que llegan en t=16s de las pistas: 92 y 98.

El orden de atención sera: 51 - 23 - 12 - 4 - 92 - 96-98 - 101 - 119
Tiempo total:
Pista 51 a pista 4: 47ms
Pista 4 a pista 119: 115ms
Total: 162ms

2. a) Si nos dice que se pueden direccionar 4G bloques, entonces las direcciones son de 32 bits (4B) cada una.

2^32 = 4G

Por otro lado, si el puntero indirecto simple apunta a 256 bloques de datos entonces:
Cant ptros x bloque = 256 = tam bloque / tam ptro
256 * 4B = tam bloque
1024B = 1KiB = tam bloque

El tamaño máximo teórico de un archivo es aproximadamente lo direccionado por su puntero de mayor indirección (doble): 2*(256)^2) * 1KiB = 128MiB
El tamaño máximo real también es de 128MiB, dado que el tamaño teórico del FS no me lo limita(1TiB) y el tamaño de la partición tampoco(4GiB).

b) Para que el tamaño del archivo sea de al menos 256MiB:

Opcion A: se deben agregar dos punteros indirectos dobles.

(10+256+4*(256)^2) * 1KiB = 256MiB (aproximadamente)

Opcion B: se debe agregar un puntero indirecto triple

(10+256+2*(256)^2+(256)^3) * 1KiB=

En la opción B utilizamos menos espacio para punteros en el inodo pero para archivos "grandes" requerimos más accesos que en la A.

3.

1.

En paginación, en promedio se desperdicia media página, en este caso, al tener segmentación, se desperdicia media página por segmento. Entonces, por proceso se desperdiciaría 1/2pág * 5 (5 segmentos).
Como dice que el sistema suele tener un nivel de multiprogramación de 8, significa que tenemos que multiplicarle 8 a lo anterior para saber el promedio de frag int del sistema.
20 KiB = ½ * Tam Päg * 5 * 8
**Tam pág = 1 KiB = Tam marco**
Con este dato sabemos que los últimos 10 bits de la direc lógica se utilizan para el offset
También sabemos que los bits de mayor peso van a referirse al nro de segmento, y que el mismo no puede ser mayor a 5.
DL = 0C02221h

C -> 1100 => debe tratarse del segmento 3 , ya que segmento 6 o 12 no son válidos para el sistema. (a su vez, si el segmento fuese 0, el valor de nro de pág obtenido no sería ninguno de los que muestra la TP -> nota del enunciado)
=> 12 bits del medio -> nro pag

C (mitad) 0 2 A(mitad) -> nro de pág → 00 0000 0010 00 = página 8

Mirando en la TP vemos que tiene P = 0 => PF

Dice que la DF es 01221h → nro frame = 0001 00 = 4 => queire decir que la víctima es la pág 9 que está en dicho frame (P = 1).
Podría ser clock modificado ya que vemos otras dos págs presentes que tienen u = 1 pero m = 0
No podría ser LRU ya que justamente tiene el último valor de referencia más alto

2. No posee fragmentación externa ya que la memoria en sí está páginada, por lo que no tienen que estar contiguos los frames en memoria.
Por lo que vemos la dirección NRO PAG | OFFSET tiene 22 bits => 4 MiB por segmento
En total cada proceso podría ser de 20 MiB