Sistemas Operativos
1° Rec 2° Parcial 1C2023 - 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.

	Multiprogramacion	Overhead	Fragmentación
Seg Pura	El nivel de multiprogramación está limitado por el tamaño de los segmentos y por la cantidad de memoria real ya que no es muy utilizado con MV.	Sin contemplar compactación, es menor debido a que para acceder a una dirección física es necesaria solamente una traducción mediante la tabla de segmentos.	Posee únicamente fragmentación externa dado que los segmentos se crean del tamaño requerido.
Seg Paginada	El nivel de multiprogramación es más alto ya que puede hacer uso de la paginación bajo demanda.	Es mayor ya que cada traducción requiere accesos tanto a tablas de segmentos como a tablas de páginas.	Tiene solamente fragmentación interna en la última página de cada segmento.

2. Para empezar, todo se refiere a una estructura tipo UFS, cualquier relación con FAT o NTFS sería errónea ya que si bien en ese contexto existe el SL, no existe el HL.
   SL → apunta al path del archivo
   HL → apunta al mismo número de inodo
   Para eliminar el archivo definitivamente, el usuario tendría que eliminar tanto la entrada de directorio "original" como la del HL, ya que el inodo solamente se libera cuando su contador de HL llega a 0.

3. Es posible, dado que ante un evento de thrashing masivo con sustitución global, los procesos involucrados quedarían bloqueados eventualmente y el uso de CPU recaería.

4. Es V. La performance es la mejor ya que se trabaja con solo una estructura de lista ordenada de mayor a menor. Aunque si bien es la mejor en ese sentido, tiene una alta tasa de compactación debido a fragmentación externa.

5. Ya sea con una o más particiones swap dentro de un mismo volumen, el método más apropiado de asignación sería el contiguo, ya que al ubicar un bloque necesario se accede secuencialmente al resto de los bloques. En FAT y UFS se utilizan asignación encadenada e indexada respectivamente.

Práctica

1.
    a) El bloque 8 corresponde al bloque de punteros del archivo, por lo cual, su contenido serán los punteros a bloques de datos que siguen del archivo: {4,5,6,9,...}, el resto del bloque no es utilizado por ahora.
    b) En primer lugar, deberíamos actualizar su inodo para que represente el nuevo tamaño, luego, el archivo pasaría a ocupar 9 bloques de datos en lugar de 7, por lo tanto deberíamos asignarle 2 bloques nuevos, suponemos el 2 y el 7 que están libres, marcándolos ocupados en el bitmap. Por último deberíamos agregar los punteros correspondientes en el bloque 8 (bloque de punteros), quedando su contenido como: {4,5,6,9,2,7,...}
    c) Si truncáramos el archivo a 300 KiB, el mismo pasaría a ocupar más de 259 bloques de datos que es lo que podemos direccionar con los 3 punteros directos + el puntero indirecto simple debido a que hay 256 punteros por bloque. Por lo tanto, deberíamos asignar un bloque de punteros (doble) cuyo puntero agregaremos al inodo para poder direccionar un nuevo bloque de punteros (simple) que apuntará a los bloques de datos restantes.

2.
    a) **190 KiB** por ser el proceso de mayor tamaño (PB).
    b) **No**. No se pudieron cargar los procesos **PG y PE**. 74 KiB SO + primeros 5 procesos ejecutando (en orden): PB, PD, PF, PA, PC = 74 + 5*190 = 1024 KiB.
    c) Frag SO = 0, Frag A = 15, Frag B = 0, Frag C = 5, Frag D = 95, Frag F = 65 = **180 KiB**
    d) **No. No se puede cargar PE**. Cálculo: 74 + 190 + 95 + 125 + 175 + 185 = 844 KiB ocupados.
    844 + PG = 844 + 135 = 979. Restante: 1024 - 979 = 45 KiB (PE necesita 155 KiB)

3.

Direcciones

Calculamos el número de página (relativo al segmento) dividiendo la cantidad de bytes que nos corremos desde el puntero por el tamaño de página, por lo tanto:

if (puntero[1336] == 'A') => lectura página 1 => lectura cluster 1
puntero[3073] = 'B' => escritura página 3 => escritura cluster 3
if (puntero[2502] == 'C') => lectura página 2 => lectura cluster 2
puntero[2047] = 'D' => escritura página 1 => escritura cluster 1
if (puntero[7166] == 'E') => lectura página 6 => lectura cluster 6
puntero[6147] = 'F' => escritura página 6 => escritura cluster 6

a)

Pagina/cluster 1 (lectura)	Pagina/cluster 3 (escritura)	Pagina/cluster 2 (lectura)	Pagina/cluster 1 (escritura)	Pagina/cluster 6 (lectura)	Pagina/cluster 6 (escritura)
1	1	1	1	1	1
	3 (m=1)	3 (m=1)	3 (m=1)	6	6 (m=1)
		2	2	2	2
PF	*PF*	*PF*		*PF*	

b)

- Página/Cluster 1: 1 cluster accedido, 2 accesos a FAT (por ser el 2do cluster)
- Página/Cluster 3: 1 cluster accedido, 4 accesos a FAT
- Página/Cluster 2: 1 cluster accedido, 3 accesos a FAT
- Cluster 6:
    - 1 cluster accedido (nro 3) para escritura, 4 accesos a FAT
    - 1 cluster accedido (nro 6) para lectura, 7 accesos a FAT

Totales:

- 5 clusters accedidos
- 20 referencias a la FAT