Sistemas Operativos
2° Parcial 1C2019 - 1° Rec - 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.

	Fijo	Dinámico	Buddy
Nivel de multiprogramación	La cantidad de particiones limita el nivel de multiprogramación	No tiene problemas	La cantidad de particiones limita el nivel, pero dado que las particiones pueden dividirse, este límite no suele ser un problema
Overhead	Bajo. Asignar consiste en revisar un vector de particiones libres. Liberar es el proceso opuesto.	Bajo para asignar. Se revisa una lista de particiones libres. Es posible necesitar compactación, haciéndolo alto. Para liberar, es bajo, porque simplemente se marca la partición como libre, y se consolida con las vecinas.	Medio para asignar. Se dividen las particiones en dos hasta encontrar la óptima. Medio para liberar, se marca la partición como libre y se consolidan todos los buddies
Fragmentación	Interna	Externa	Ambas

Nota: la comparación es simplemente entre estos tres esquemas. Algo que podría ser medio aquí, podría ser alto o bajo, comparado con otros.

2. La entrada podría verse así, incluyendo los campos más relevantes

Nombre	Inodo	Tamaño	Creación	Otros
file.txt	N	X	Fecha1	...
hl.txt	N	X	Fecha1	...
sl	M	Y	Fecha2	...

3. Un FCB podría guardar información acerca de los permisos de los usuarios, en formato Propietario-Grupo-Universo, una lista de control de accesos o bien el propietario del archivo, permitiendo que ese usuario sea el único que puede accederlo.

El FCB se guarda dentro de la metadata del filesystem (por ejemplo, en la tabla de inodos en el caso de UFS), salvo casos particulares como FAT, que la distribuyen en las entradas de directorio o bien en la metadata de cada archivo.

4. a) Verdadero. Es necesario definirlas en tiempo de ejecución, sino se pierde la posibilidad de reubicar las páginas en cualquier lugar libre de la memoria
   b) Verdadero. Cada nivel generará un acceso extra a memoria.

5.

	RAID 0	RAID 0+1	RAID 5
Redundancia	No tiene	Al incluir RAID 1, tiene discos espejados	Incluye el equivalente a un disco de paridad, distribuido en todos los discos.
Overhead	Bajo, porque divide los archivos en bandas	Bajo, divide los archivos en bandas y las espeja	Medio, porque divide a los archivos en bandas y calcula la paridad de cada banda
Tolerancia a fallos	No soporta caídas	Soporta la caída de una copia de cada disco	Soporta la caída de un sólo disco

Práctica

1)
Capacidad del disco = Cilindros x Cabezas x Sectores por pista x Tam Bloque
128 MB = C x 8 x 64 x 2 KB
C = 128

FSCAN:
Cola 1 = 50 - 60 - 100
Cola 2 = 100 - 100 - 102 - 127 (tope) - 80 - 10
Cola 3 = 10

LOOK: 60 - 80 - 100 - 100 - 100 - 102 - 10 - 10

2)
Existen muchas configuraciones válidas a plantear:
Una opción -> mantener tamaño de bloque. Se podría tener 4 ptrs directos para optimizar el acceso de los archivos más chicos. Por otro lado, para llegar a referenciar el máximo de los archivos “normales” necesitaríamos 4 ptrs ind simples. Por último podríamos agregar un doblemente indirecto para tener archivos potencialmente más grandes.

Ptrs por bloque = 2^10/2^2) 2^8 = 256

a) Max teórico FS ->FAT : 2^28*2^10 = 256GiB UFS : 2^32 * 2^10 = 4TiB
b) Máx File téorico -> FAT == FS = 256GiB UFS : (2^8)^2 * 2^10 + 4*(2^8)*2^10= 2^26 + 2^20 = 65MiB aprox
c) Frag interna. En ambos se va a dar en el último bloque de datos. Sin embargo, en UFS también podría tener en los bloques de punteros
d) Archivo 1MiB -> 2^10 bloques de datos. FAT -> 1024 accesos UFS -> 1024 accesos a BD + 4 accesos a BP
e) Creación nuevo archivo
    i) FAT: entrada de directorio apuntando al primer bloque + modificar las entradas enlazando los bloques en la FAT
    ii) UFS: asignar inodo + bloques , modificar bitmaps de inodos y bloques + modificar contenido de inodo en tabla de inodo, crear entrada de directorio apuntando al inodo, modificar superbloque

3)
Formato dirección: 4 bits segmento, 2 bits página, 10 bits desplazamiento

lectura(0x00E5)
- Segmento 0, Página 0: Permisos ok. Página presente en memoria. Dos tablas accedidas.
- Dirección física: 11*1024 + offset = 11264 + offset = 0x2C00 + 0xE5 (offset) = 0x2CE5

ejecución(0x1D61)
- Segmento 1, Página 3: Permisos ok, pero página inexistente en el segmento. Una tabla accedida.
- Dirección física: No existe (Error de direccionamiento)

lectura(0x1389)
- Segmento 1, Página 0: Permisos impiden lectura. Una tabla accedida.
- Dirección física: No existe (Error de permisos)

lectura (0x23AC)
- Segmento 2, Página 0: Permisos ok. Dos tablas accedidas.
- Dirección física: 20*1024 + offset = 20480 + offset = 0x5000 + 0x3AC = 0x53AC

escritura(0x23FD)
- Segmento 2, Página 0. Permisos ok. Offffset 0x3FD => 1021 bytes (excede largo segmento)
- Dirección física: no existe (error de direccionamiento).

Ninguna tabla es modificada