Preguntas Filesystems

1. El hecho de que podamos abrir un dispositivo de almacenamiento como un gran **archivo**, es como una visión lógica que nos da el sistema operativo de él? ¿O realmente siempre es así? O sea, cuando en Windows por ejemplo abrimos un bloc de notas, ¿busca los datos en el gran archivo que representa al disco duro? Realmente es así como el SO manipula a los filesystem, es como una capa de abstracción.

2. Entiendo que los ***filesystem*** se asocian a particiones, o sea cada partición que tenga un dispositivo de almacenamiento tendrá un *filesystem* (¿que pueden ser diferentes entre sí no? Sí, pero dentro de una partición debería haber un único filesystem).
Y un *filesystem* es la estructura de la partición; la estructura de cierta manera para que por un lado se almacenen metadatos sobre el propio *filesystem* y los archivos que contiene (tanto archivos de datos como directorios), y por otro se almacenen los datos en sí de los archivos. ¿Así sería? Sí.

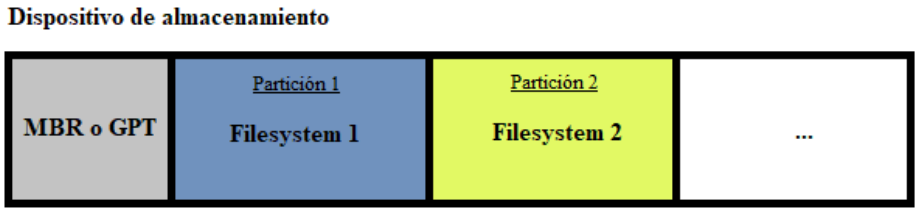
3. El sistema operativo **accede** entonces al *filesystem* de un dispositivo de almacenamiento cuando quiere ubicar/encontrar un archivo, leer sus datos, o escribirle datos no? ¿Esto quiere decir que por ejemplo Windows debe tener varios programas que sepan interpretar diversos *filesystems* (como FAT32, exFAT, NTFS, etc…), o sea, varios parsers? Sí (y también que puedan escribir cosas), no serán .exe puntuales, pero sí. Por eso también dependiendo el sistema operativo que tengamos, vamos a poder acceder a algunos filesystems, y a otros no. Por ejemplo, **nativamente** Windows solo soporta NTFS, ExFAT, UDF y FAT32; si conectamos un dispositivo con un filesystem que no reconoce, nos pedirá que lo formateemos para poder utilizarlo.

4. Y hablando de **parsers**, yo entiendo que un parser (dentro del contexto de *filesystems*) es un programa que lee un *filesystem* en crudo para identificar su estructura. O sea, va leyendo los bytes de los sectores o clusters de un dispositivo de almacenamiento a través de su *filesystem* (¿a través del gran archivo que lo representa? Digamos que sí). La idea del parser es convertir/traducir los bytes en texto legible para nosotros (salvo que esos bytes no tengan un significado para el usuario, y sean meramente para procesamiento interno del sistema, que podrían ser objetos.), para así obtener la información de las estructuras del *filesystem*. ¿Así sería? Sí.

Leer bytes → identificar estructuras del *filesystem* → mostrar de forma legible la información de esas estructuras (tanto información del *filesystem* en sí, como metadatos y datos de directorios y archivos).

5. Cuando hablamos de **MBR** o **GPT**, ¿son formas de organizar las particiones de un dispositivo de almacenamiento? Porque no me quedo del todo claro su función. Sí, son tablas de particiones, accedemos a ellas para saber en qué posición del dispositivo de almacenamiento se encuentran las particiones, además contiene metadatos como si es booteable o no, que filesystem tiene, etc…

Y entiendo que suelen estar en los primeros sectores del dispositivo; entonces cuando hablamos de un *filesystem*, ¿estamos dejando de lado si el dispositivo usa MBR o GPT no? O sea, no es que el *filesystem* abarque "todo" el dispositivo, ¿sino más bien es una parte? ¿Algo así?: Sí.



6. Cuando **formateamos** un dispositivo, ¿lo que se hace es escribirle los bytes correspondientes a la estructura del *filesystem* elegido? Por ejemplo, si elegimos darle un formato de FAT32, ¿el sistema operativo escribirá el área de boot (volumen id), estructurará las FATs, etc..? ¿Entonces los sistemas operativos también deben tener programas que sepan cómo escribir los bytes para formatear una partición con cierto *filesystem*? Sí.

¿Y cuando formateamos también se escribe la estructura de la MBR o GPT? Lo que formateamos es una partición, y no el dispositivo entero; por ende, la tabla de particiones no se tocará. ¿O eso viene "de fábrica" en el dispositivo de almacenamiento? Generalmente sí ya vienen con una tabla de particiones, pero sino desde el administrador de discos de Windows podemos asignarle una e incluso cambiar de MBR a GPT o viceversa. ¿Y se puede usar indistintamente MBR o GPT para cualquier tipo de *filesystem* que haya en las particiones que organiza? Sí, ambas son compatibles con cualquier filesystem que haya en las particiones del dispositivo, las diferencias entre ellas viene por otro lado.

7. Cuando por ejemplo conectamos un pendrive con FAT32, ¿el sistema operativo **carga en memoria** su área de boot y ambas FATs y quedan cargadas hasta que lo desconectemos? Y en cambio los clusters de archivos y directorios, ¿solo se cargan en memoria cuando solicitamos abrir algún archivo? Generalizando, de un *filesystem* solo se carga inicialmente en memoria sus estructuras "organizativas" y no los datos en sí, no? En general, sí es así; de todas formas, los sistemas operativos buscan hacer un uso eficiente de los recursos del sistema, buscan optimizar; por lo que quizás en el caso de FAT mantiene cargada sola una parte y si requiere la otra, recién ahí la va a buscar al disco y la carga en memoria.

8. En las charlas que me pasaste, escuché que varias veces mencionaste "**driver del *filesystem***", y no entiendo bien a que te referís con eso. ¿Es distinto al "driver del dispositivo de almacenamiento"? Repreguntar a Bruno.

9. El hecho de que podamos manejar por ejemplo a un pendrive de la misma manera que un disco duro (que sea posible recorrer sectores o clusters de un pendrive sabiendo que son conceptos de discos mecánicos y no de memorias flash), ¿es una **abstracción** del sistema operativo, no? O sea, ¿nos permite manejar a cualquier dispositivo de almacenamiento por igual? Sí.