Formateo de Particiones

Cuando formateamos una partición de un dispositivo de almacenamiento, lo que hacemos es darle un formato, es decir, es darle un sistema de archivos. Le escribimos todos los bytes necesarios para conformar las estructuras de datos del filesystem en cuestión.

El concepto de que un formateo "borra" todos los datos, es algo relativo.

Primero que nada, el concepto de "borrar" un archivo de por ejemplo el disco rígido, no es algo completamente real. Lo que se hace es una **eliminación lógica**, se lo "elimina" del filesystem de la partición donde se encuentre.

Por ejemplo, en Ext2, cuando borramos un archivo lo que pasa son 2 cosas (a grandes rasgos): primero, al *directory entry* que tenga la asociación *nombre_de_archivo – n°_de_inodo*, se le setea el n° de inodo a 0 (inodo nulo); y segundo, en el *bitmap de inodos* del *block group* correspondiente se marca ese inodo como 'libre' (*info sobre esto en el PDF de ext2/3, buscando la palabra 'delete'*). De esta maneara, **se pierde la referencia al archivo dentro del filesystem de la partición**, ergo, **se pierde la referencia al archivo en el disco**. Pero tanto los metadatos (inodo y directory entry) como los datos (el archivo como tal), siguen estando en el disco, los bytes siguen ahí! (y *seguirán ahí hasta que por ejemplo creemos un nuevo archivo, lo cual el nuevo inodo podría ubicarse justo en la posición del inodo que 'se borró' y los datos del archivo ubicarse justo donde estaban los datos que 'borramos', entonces habremos sobrescrito los bytes del disco*).

La forma de efectivamente eliminar un archivo (**eliminación física**), es o bien escribiendo en disco todos bytes en 0 en las posiciones donde estaba ubicado el archivo, o bien que un nuevo archivo que creemos sobrescriba los bytes de un archivo borrado 'lógicamente'.

Entonces ahora sí, siguiendo con los formateos, hay 2 tipos:

- **Formateo rápido**: lo que hace es escribir de 0 todas las estructuras de datos de un determinado filesystem, por ende, para el filesystem (*y por ende para nuestro sistema operativo*), la partición en cuestión estará 'como nueva', 'vacía' (*todos los inodos y bloques de datos marcados como libres, salvo los que se ocuparon para el seteo inicial, como la creación del directorio raíz, etc*). Aunque obviamente toda la estructura en sí del filesystem algo de espacio ocupará en disco. Pero en ningún momento se escriben 0's porque sí (*solo se escriben si la creación del filesystem lo demanda, o sea por ejemplo algún campo del superbloque en 0*), entonces todos los datos que teníamos en la partición antes de formatearla, seguirán estando ahí, salvo que la escritura de las estructuras del filesystem, los haya sobrescrito*****.

- **Formateo completo**: al igual que el formateo rápido, escribe un filesystem de 0 en una partición, pero rellena todo lo restante de la partición con bytes en 0 (por eso tarda más, tienen que escribir un montón de bytes). Entonces, acá efectivamente se borran todos los archivos que tengamos. No habrá forma de recuperarlos (salvo casos muy puntuales).

Nota1*: *Por esta razón es que existen programas 'recuperadores de archivos borrados', recrean las referencias a los archivos (o por ahí directamente leen los bloques de datos de la partición en busca de secuencias de bytes que tengan alguna 'coherencia' entre sí)*.

Nota2*: *Si formateamos una partición con el mismo filesystem que teníamos antes, habrá más chances de poder recuperar algún archivo que si por ejemplo la formateamos con un filesystem distinto, ya que cada filesystem tiene distintas estructuras que se escriben en distintos lugares del disco*.

Info sobre formateos:

- https://www.lifewire.com/what-does-it-mean-to-format-something-2625882
- https://superuser.com/questions/182629/what-does-format-do

Más info:

- https://security.stackexchange.com/questions/121673/to-what-extent-does-formatting-a-disk-securely-remove-its-data
- https://superuser.com/questions/1107104/does-formatting-really-remove-everything-on-a-physical-hard-drive/1107109
- https://superuser.com/questions/1046136/windows-7-does-formatting-a-disk-actually-write-zeros-to-it
- https://www.lifewire.com/free-data-destruction-software-programs-2626174
- https://www.diskpart.com/articles/write-zeros-to-hard-drive-8523.html
- https://www.webopedia.com/insights/completely-erase-harddrive/
- https://www.quora.com/How-exactly-does-data-recovery-software-work

Nota: siendo estrictamente correctos, hay una sutil diferencia entre 'formatear un disco' y 'formatear una partición'. Por eso cuando nos queramos referir a "borrar" todo, lo correcto es decir 'formatear una partición', que quiere decir '*escribir un filesystem en una partición*', o sea, darle un nuevo formato. Recordemos que los filesystems se asocian a particiones, no al disco como tal (*dispositivo de almacenamiento = tabla de particiones + particiones*).

Más info sobre este detalle:

- https://www.quora.com/Does-formatting-a-hard-drive-completely-erase-all-files-on-it
- https://stackoverflow.com/questions/49662151/what-is-the-difference-between-formatting-and-mounting-a-file

Creating the Ext2 Filesystem

There are generally two stages to creating a filesystem on a disk. The first step is to format it so that the disk driver can read and write blocks on it. Modern hard disks come preformatted from the factory and need not be reformatted; floppy disks may be formatted on Linux using the *superformat* utility program. The second step involves creating a filesystem, which means setting up the structures described in detail earlier in this chapter.

Ext2 filesystems are created by the *mke2fs* utility program; it assumes the following default options, which may be modified by the user with flags on the command line:

- Block size: 1,024 bytes
- Fragment size: block size (block fragmentation is not implemented)
- Number of allocated inodes: one for each group of 4,096 bytes
- Percentage of reserved blocks: 5 percent

The program performs the following actions:

1. Initializes the superblock and the group descriptors.
2. Optionally, checks whether the partition contains defective blocks; if so, it creates a list of defective blocks.
3. For each block group, reserves all the disk blocks needed to store the superblock, the group descriptors, the inode table, and the two bitmaps.
4. Initializes the inode bitmap and the data map bitmap of each block group to 0.
5. Initializes the inode table of each block group.
6. Creates the */root* directory.
7. Creates the *lost+found* directory, which is used by *e2fsck* to link the lost and found defective blocks.
8. Updates the inode bitmap and the data block bitmap of the block group in which the two previous directories have been created.
9. Groups the defective blocks (if any) in the *lost+found* directory.

Comando para formatear particiones en Linux: mke2fs (o para crear otros fs: mkfs)

Comando para formatear particiones en Windows: diskpart (o click derecho en la partición)