

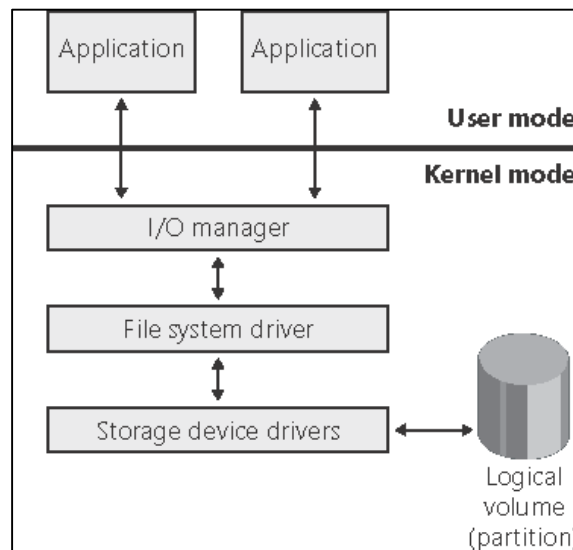
Driver del Filesystem - Concepto

El driver de un filesystem es un componente de **software** encargado de **interpretar** las estructuras de datos del **filesystem** presente en una **partición** de un **dispositivo de almacenamiento**.

Es quien sabe cómo ubicar un archivo dentro del filesystem, por ende, es quien implementa las funciones que permiten que se puedan completar las operaciones básicas de archivos (*read, write, seek, create, delete, etc*).

Se le dice driver porque es quien sabe cómo manejar un filesystem. Al igual que por ejemplo el [driver](#) de la impresora sabe cómo manejarla (más bien, sabe cómo comunicarse con ella y darle órdenes, el que realmente maneja las partes físicas de la impresora es su [controlador/placa_verde](#)). Pero a diferencia de los drivers convencionales (como el de la impresora), no se comunica directamente con una pieza de hardware (no se comunica con un controlador), sino que se va comunicando con el driver del dispositivo de almacenamiento donde se haya la partición con el filesystem, para decirle que lea tales bytes o escriba otros en el disco, a medida que va interpretando/moviéndose_por el filesystem.

Por ende, el driver del filesystem es una capa de abstracción más entre el sistema operativo y el hardware. Cuando un usuario por ejemplo quiere ver el contenido de un archivo .txt, la aplicación (por ejemplo, el bloc de notas), se lo avisa al sistema operativo (llamada al sistema), este se comunica con el driver del filesystem de la partición donde se encuentre el .txt, y este se comunica con el driver del dispositivo de almacenamiento que contiene dicha partición.



[File System Driver Architecture](#)

Info device drivers (joyitas):

- <https://tldp.org/LDP/tlk/dd/drivers.html>
- <https://www.oreilly.com/library/view/linux-device-drivers/0596005903/ch01.html>
- <https://www.opensourceforu.com/2014/10/an-introduction-to-device-drivers>

Driver del Filesystem - Kernel

Como todo driver, el driver del filesystem es un componente fundamental de un sistema operativo, por ende, está dentro del **kernel**. Es un módulo 'built-in' del kernel, es decir, cuando se inicia el sistema y se carga el kernel en memoria, estarán cargados los drivers (*igual no descarto que también haya drivers 'dinámicos', o sea, que también sean parte del conjunto de programas del kernel, pero que se carguen en la memoria reservada para el kernel cuando se requieran*).

Por ejemplo, Linux soporta varios filesystems, es decir, tiene varias implementaciones de drivers de filesystems, como por ejemplo ext2/3/4 (los fs nativos de Linux), fat32, NTFS, exFat, etc.

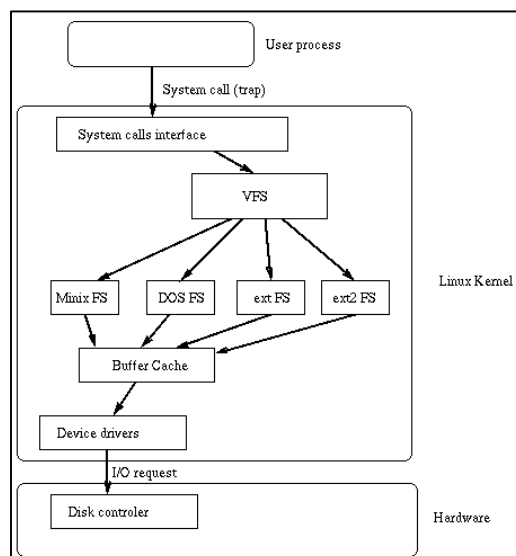
Todos ellos los podemos ver acá: <https://github.com/torvalds/linux/tree/master/fs>.

Por otro lado, también existen drivers de filesystem que no forman parte del kernel, no se ejecutan dentro del 'kernel-space', sino que se ejecutan dentro del 'user-space' ([FUSE](#)).

Por ejemplo, hay un driver FUSE de NTFS que viene dentro del paquete '[ntfs-3g](#)' (*que también contiene el paquete de utilidades 'ntfsprogs', que nos permite por ejemplo ejecutar el comando 'mkntfs' para formatear una partición*), o sea lo podemos descargar desde el gestor de paquetes de cualquier distribución Linux (*igual en general ya viene instalado en la distro*). No es un módulo (.ko) del kernel como tal. Hoy en día por lo que leí, los usuarios de Linux prácticamente hacen uso de este driver FUSE para manejar particiones con NTFS, ya que el driver integrado en el kernel tiene ciertas limitaciones (*igual los desarrolladores de 'ntfs-3g' colaboraron con el driver integrado, que fue parte del proyecto 'The Linux-NTFS Project', así que son viejos conocidos jeje*).

Info útil:


- <https://unix.stackexchange.com/questions/87625/what-is-difference-between-user-space-and-kernel-space>
- <https://arstechnica.com/gadgets/2021/08/paragon-is-working-to-get-its-ntfs3-filesystem-into-the-linux-kernel/> (*joyita*)
- <https://unix.stackexchange.com/questions/146080/how-do-i-find-out-what-filesystem-drivers-are-compiled-in-into-the-linux-kernel>
- [The Virtual File System](#)



Driver del Filesystem - Extras

Conceptos interesantes sobre los filesystems:

- https://es.wikipedia.org/wiki/Sistema_de_archivos
- <https://superuser.com/questions/1309282/no-filesystem-is-same-as-no-operating-system>
- <https://www.quora.com/How-do-I-move-files-without-any-operating-system-in-computer>

**Matteo Ianeselli**, both loves and hates computer programming with passion
Answered 5 years ago · Author has 1.9K answers and 4.8M answer views






Current mass-storage hardware doesn't have the concept of *files*: all it provides is a set of commands to read/write fixed-size blocks of data. How those blocks are organized as *files* in a *filesystem* is something that is implemented in software, usually by the kernel of the operating system running on the computer.

A filesystem, in the end, is basically a complex data structure that's kept off-memory: to deal with it, you need the software implementing it.

Bootloaders like [GNU Grub](#) are barely able to *read* some kinds of filesystems only because they come with the machine code minimally implementing what's needed to recognize some popular filesystem types and how to read a file from them. But that code does not fit in the boot sector, so it has to be loaded in memory by other, means in a second or third stage.

UEFI-based computers come with firmware implementing the filesystem and partitioning scheme that is mandated by the EFI, which is a FAT-like filesystem, but quite obviously doesn't know anything about other filesystem types.

395 views · View upvotes

 2    

O sea, el **Filesystem** como tal es la estructura interna de una partición de un dispositivo de almacenamiento, con el fin de facilitar el almacenamiento y posterior recuperación de datos (poder ubicar los archivos dentro de un dispositivo de almacenamiento). Un filesystem es un proceso de *Diseño*, los que crean un filesystem diseñan sus estructuras de datos y deciden cómo las distribuirán en el dispositivo de almacenamiento (además de decidir otras cuestiones, como convenciones de nombrado de archivos, etc, o sea, qué se va a poder hacer y de qué manera).

Luego, se debe implementar un programa (*Desarrollo*) que efectivamente estructure una partición con el filesystem que se diseñó: que grabe los bytes de cada estructura donde corresponda (esto es básicamente la acción de '**formatear**' una partición).

Y por último, se debe implementar el programa (*Desarrollo*) que sepa manejar el filesystem: que sepa cómo leerlo/interpretarlo, que sepa moverse por los bytes, identificar las estructuras de datos y encontrar los archivos almacenados. O sea, es un programa que implementa todas las funciones necesarias para que se puedan realizar las operaciones básicas de archivos (create/delete, open/close, read/write, etc). Este programa (o mas bien, conjunto de programas), es el **Driver del Filesystem**.

Más sobre filesystems:

- <https://stackoverflow.com/questions/what-is-the-form-of-file-system-like-ext3-ext2-ext>

Filesystem en
sentido **amplio**

Unfortunately, what "filesystem" exactly means depends on the context.

Most commonly, "filesystem" describes the **on-disk format** of volumes/partitions. In that sense, APFS, ext, FAT32, and XFS are some of different filesystems. For example, you may hear people say something like "APFS supports alternate data streams, XFS doesn't".

Many times the term "filesystem" is used to describe the **ecosystem**: the on-disk format of volumes/partitions, plus the OS drivers that read and write these formats. For example, you may hear people say something like "ext2 is not crash-safe, upgrade your partitions to ext3" (see below).

Sometimes, the term "filesystem" is used to describe the **driver** that reads and writes the on-disk format. For example, you may hear people say something like "ZFS doesn't work on windows, but NFS works everywhere".

NFS, in particular, has a driver, but doesn't have an on-disk format, because it just asks a remote server to store everything on its own filesystem, whichever that is.

The distinction between on-disk format and the driver which reads and writes it is particularly confusing for ext2/3/4. **The ext family of filesystem drivers shares a common on-disk format** - if you dig up an ext* partition, nowhere on the partition will it say "my version is ext3". Instead, the partition will have a list of features.

What does have different versions is the driver - there are drivers for ext2, ext3, and ext4, and each version adds support for new **features**. So, you can create a partition using the ext3 driver, and the partition will use features supported by the ext3 driver - both the ext3 and ext4 drivers can read it. Then you upgrade it using the ext4 driver in order to use a new on-disk feature (like high-resolution timestamps), but then you have to read it using the ext4 driver - the ext3 driver can no longer read it because it doesn't support the new feature.

Filesystem en
sentido **estricto**

Occasionally you may also see "filesystem" referring to a specific volume/partition that has **files and directories**. For example, you may hear people say something like "My filesystem is corrupted".

Links donde se menciona al driver del filesystem:

- <https://tldp.org/HOWTO/Module-HOWTO/x73.html> (*joyita*)
- <https://ext4.wiki.kernel.org/index.php/UpgradeToExt4>

Info sobre el desarrollo de drivers:

- <https://unix.stackexchange.com/questions/508314/how-do-i-implement-a-file-system-driver-driver-in-linux>
- <https://www.apriorit.com/dev-blog/195-simple-driver-for-linux-os>
- <http://www.osdever.net/tutorials/view/brans-kernel-development-tutorial>
- <https://superuser.com/questions/360178/what-does-make-install-do>

Libro buenísimo sobre Linux y Sistemas Operativos en general:

- <http://1.droppdf.com/files/87BCs/the-linux-programming-interface.pdf>

Diferencia driver (soft) y controlador (hard):

- https://informatica.fandom.com/wiki/Diferencia_entre_controlador_y_driver
- <http://www.differencebetween.net/technology/difference-between-device-driver-and-device-controller/>
- <https://www.quora.com/What-is-the-difference-between-device-driver-and-device-controllers>

Links extras:

- <https://unix.stackexchange.com/questions/84069/what-is-a-device-controller-and-where-does-it-fit-in-between-the-kernel-and-the>
- <https://stackoverflow.com/questions/60693988/bus-driver-vs-device-driver-vs-device-controller>
- <https://stackoverflow.com/questions/13249788/keeping-device-functionality-inside-device-controller-rather-than-os-kernel-wha>
- <https://stackoverflow.com/questions/3438770/are-device-drivers-specific-to-device-controllers-or-peripheral-devices>
- <https://stackoverflow.com/questions/11175804/device-driver-stack>

File System Filter Drivers:

- <https://docs.microsoft.com/en-us/windows-hardware/drivers/ifs/about-file-system-filter-drivers>
- <https://www.easefilter.com/kb/understand-minifilter.htm>