# Introduction to Kernel modules

| Date | @November 28, 2024 |
| --- | --- |

## Linux devices

Linux provides an abstraction to make communication with I/O easy. In this way software developer does not need to know every detail of the physical device and the portability can be increased by using the same abstraction for different I/O devices.

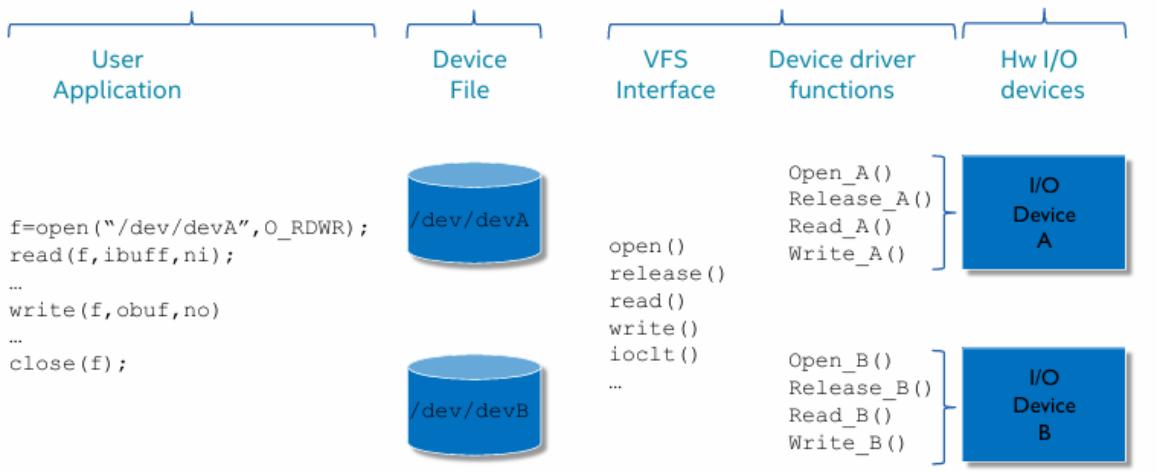
Linux recognizes three classes of devices:

- Character devices, which are devices that can be accessed as stream of words (e.g., 8-/16-/32-/... bits) as in a file; reading word n requires reading all the preceding words from 0 to n-1.
- Block devices, which are devices that can be accessed only as multiples of one block, where a block is 512 bytes of data or more. Typically, block devices host file systems.
- Network interfaces, which are in charge of sending and receiving data packets through the network subsystem of the kernel

## Virtual file system (VFS) abstraction

Character/block devices are accessed as files stored in the file system, as each device is associated with a device file. The typical usage is to open the device file, read or write data from or to device file and then close the device file. Linux forwards the open/read/write/close operations to the I/O device associated to the device file. The operations for each I/O device are implemented by a custom piece of software in the Linux kernel: the device driver.

### VFS: example




The root file system shall host one device file for each I/O device the user application needs to use.

The user application deals with the I/O device using the file abstraction: data are read/written to the device file associated with the I/O device.
As with regular files, the device file shall be opened before use and closed after use.
The low-level I/O primitives are used as defined in fcntl.h/unistd.h

The VFS establishes the association between the low-level I/O primitives used in the user application and the corresponding device driver functions.

## The device file concept

The device file is the intermediator through which a user application can exchange data with a device driver. The device file does not contain any data, while its descriptor contains the relevant information to identify the corresponding driver:

- the device file type, which could be wither c (character device), b (block device) or p (named pipe, an inter process communication mechanism)
- the major number, which is an integer number that identifies univocally a device driver in the Linux kernel
- the minor number, which is used to discriminate among multiple instances of I/O devices handled by the same device driver