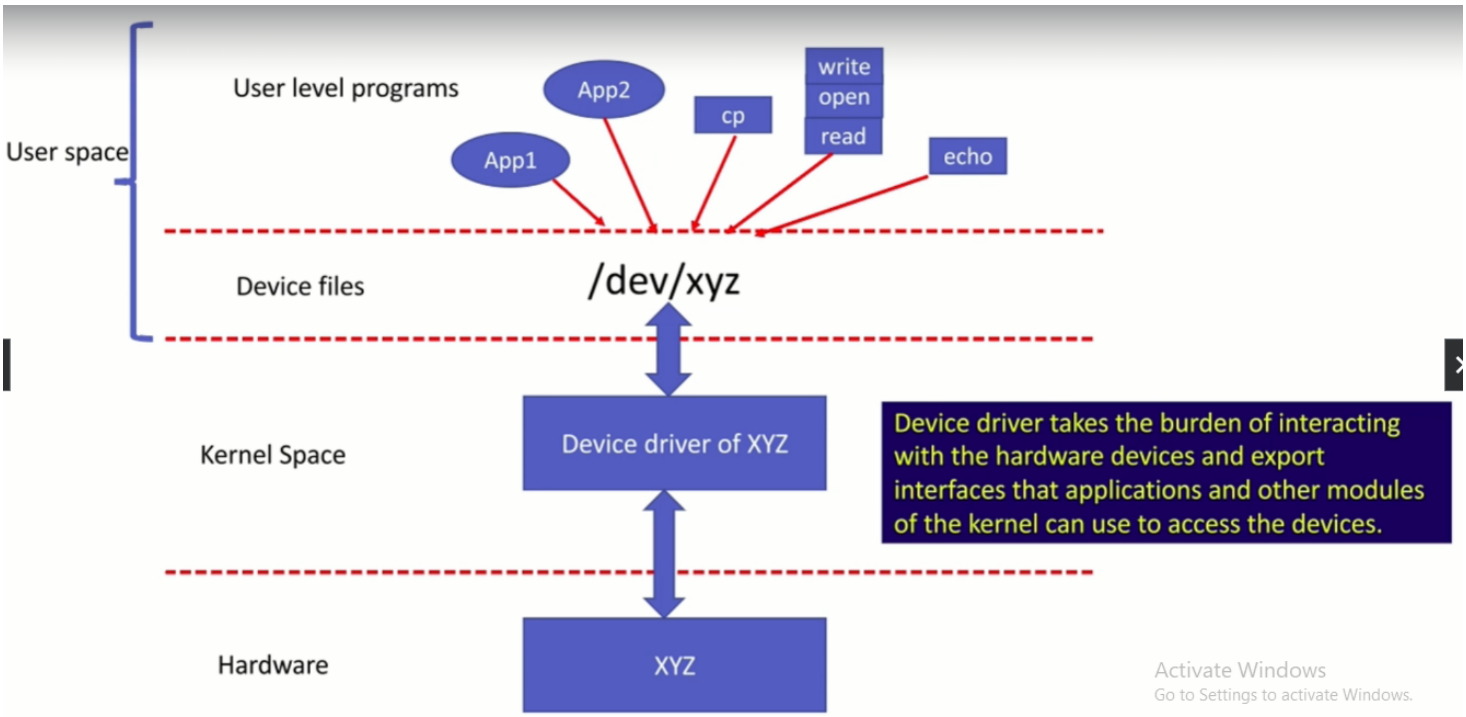
what is device driver ?

- Device driver is a piece of code that configures and manages a device.
- The device driver code knows, how to configure the device, sending data to the device, and it knows how to process requests which originate from the device.
- When the device driver code is loaded into the operating system such as Linux, it exposes interfaces to the user-space so that the user application can communicate with the device.
- without the device driver, the OS or the application will not have a clear picture of how to deal with a device.

Basically, device driver is a code which knows how to handle a particular
device. It gives a right interfaces a for the user-space to access that device, and the device driver code also abstracts various hardware details from the user-space applications.



The user applications can just use a traditional write, open, read, system calls to talk to the device. But those system calls are carefully handled by the driver code to touch various registers of the device.

The kernels job here is to connect the system calls to the drivers system called implementation methods.

Let's say, there is a device xyz and when you load a driver of that device, then that device driver exposes the interfaces to the user-space. So that the user-space applications can communicate with the device.

Device driver takes the burden of interacting with the hardware devices and export interfaces that applications and other modules of the kernel can use to access the device.

Let's say you have a device xyz. And the kernel or the operating system doesn't know how to handle a device. Because, a device has various configuration registers, it has various configuration values which needs to be program to the hardware. A before using that hardware and the kernel doesn't know how to handle the interrupts generated by the hardware.

That's a reason why you should load a device driver.

A driver knows what are the configuration values which needs to be programmed to the hardware, what are its a register address is, and how to handle the interrupts which originate from the hardware.

When you load a device driver, it connects to the kernel and it utilizes kernel services to talk to the user-space.

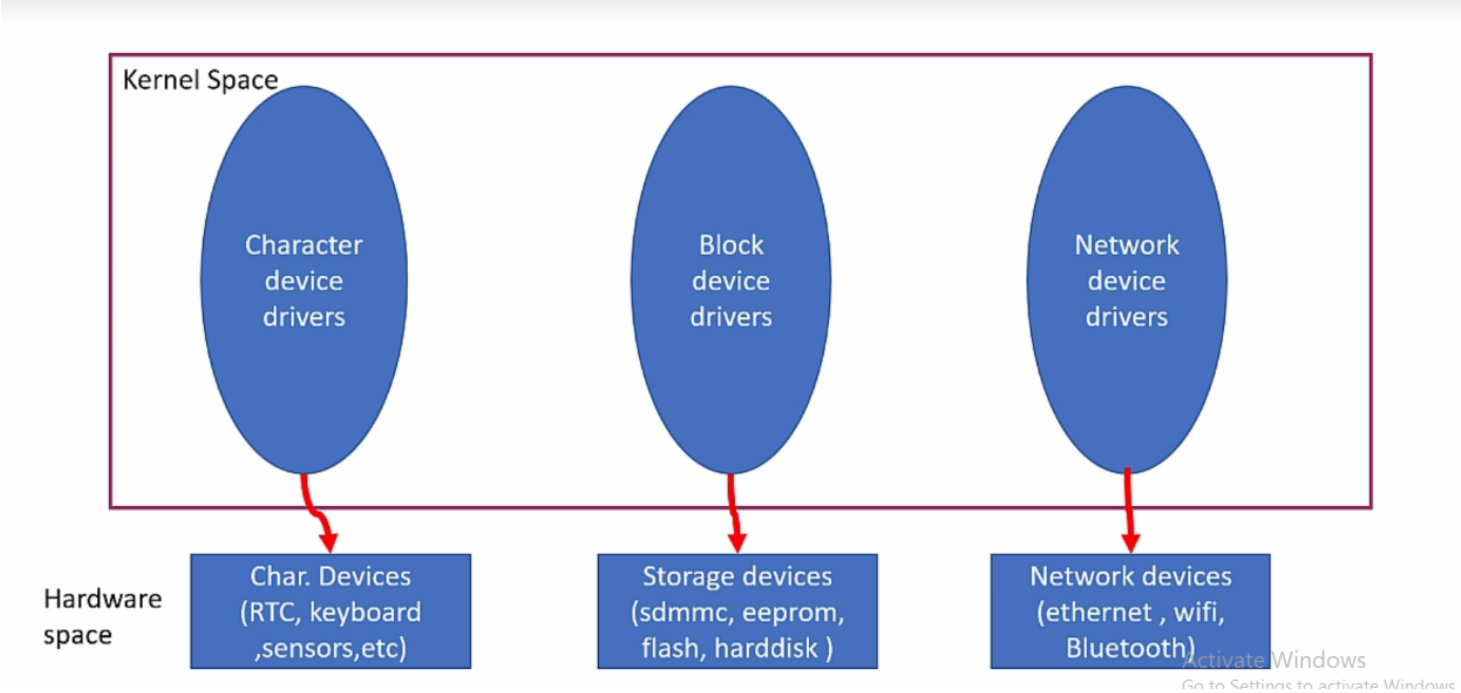
A driver should give an interface to the user level programs to communicate with a hardware.

Basically, we do read-write operations on a device.

That's why, the device driver has to create a device file interface at the user's space, so that user level programs can communicate with the hardware, using a traditional file access system calls such as open, close, read, write, etc..

So, here the job of the kernel is to connect system call execution from the user-space to the driver's system call handler methods. That will be taken care by the kernel.

Types of device driver



What's a character device driver or char driver?

Character driver accesses data from the device sequentially.i.e., byte by byte (like a stream of characters) not as a chunk of data.

Sophisticated buffering strategies are usually not involved in char drivers. Because when you write 1 byte, it directly goes to the device without any intermediate buffering, or delayed write back, or there is no a dirty buffer management as in block drivers.

Some of the examples of char devices are sensors, RTC, keyboard, serial port, parallel port, all these are character devices and you need character drivers to manage these devices.

For example, let's say you have a RTC hardware or RTC device. And to take care of that RTC device you have a RTC driver in the kernel space.

In the user-space, you just use your traditional read-write APIs or system calls a to talk to the RTC device.

For example, when you use write system call to write 1 byte of data, it directly goes to the hardware via the driver.

The driver knows how to forward that data to the RTC device. But this is not the case in a block drivers.

For example, let's say you have a memory device such as SDMMC card. So, the SDMMC specification or the design doesn't allow you to a write data byte by byte.

You either have to write in chunks such as a 512 bytes, or 1024 bytes.But in the character device you can write byte by byte.

What are block drivers?

The device which handles data in chunks or blocks is called a block device. Block drivers are more complicated than char drivers because the block drivers should implement advanced buffering strategies to read and write to the block drivers, and the disk caches are also involved.

Examples are mass storage devices such as hard disks, SDMMC, Nand flash, USB camera, etc..

What is Network Device Driver?

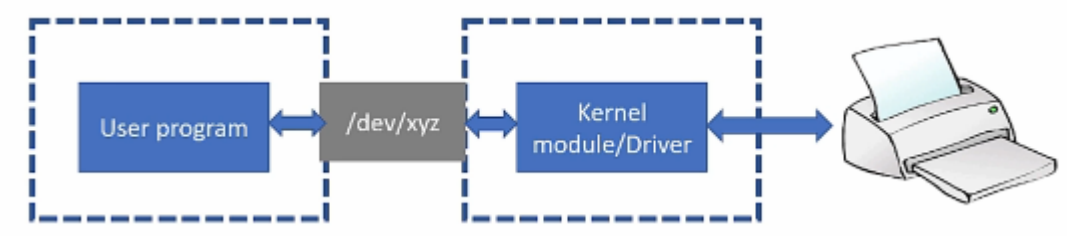
This type manages the networking hardware, enabling communication over networks. It ensures that data flows smoothly between the computer and network devices, like Ethernet cards and Wi-Fi adapters.

what is device file ?

The driver has to create some a interfaces at the user-space. So, that the user level programs can use that interface to communicate with the hardware those are called as device files.

- Most of the devices are accessed as a file in Unix/Linux system
- A device file is a special file or a node which gets populated in /dev directory during kernel boot time or device/driver hot plug events. When you boot your Linux kernel, the /dev directory is automatically it will get populated with device files, which are a generated from various drivers of the kernel. There is a special program in user level called Udev, which actually populates this /dev directory with various a device files.
- By using device file, user application and drivers communicate with each other.

- Device files are managed as a part of VFS subsystem of the kernel, where VFS stands for virtual file system. Because, this is also like a file access. So, the device files are also managed by the VFS subsystem of the kernel.



if you are wondering who creates this device file, the creation of this device file has to be triggered from your driver.

If you just go to the /dev directory, so you can see here various device files are already populated in your system.

```
kiran@kiran-fastbiteba:/dev$ ls -l
crw-------  1 root  root     10, 227 Mar 16 09:52 mcelog
crw-rw----  1 root  video    240,   0 Mar 16 09:52 media0
crw-------  1 root  root     241,   0 Mar 16 09:52 mei0
crw-r-----  1 root  kmem       1,   1 Mar 16 09:52 mem
crw-------  1 root  root      10,  56 Mar 16 09:52 memory_bandwidth
drwxrwxrwt  2 root  root            40 Mar 16 09:52 mqueue
drwxr-xr-x  2 root  root            60 Mar 16 09:52 net
crw-------  1 root  root      10,  58 Mar 16 09:52 network_latency
crw-------  1 root  root      10,  57 Mar 16 09:52 network_throughput
crw-rw-rw-  1 root  root       1,   3 Mar 16 09:52 null
crw-------  1 root  root      10, 144 Mar 16 09:52 nvram
crw-r-----  1 root  kmem       1,   4 Mar 16 09:52 port
crw-------  1 root  root     108,   0 Mar 16 09:52 ppp
crw-------  1 root  root      10,   1 Mar 16 09:52 psaux
crw-rw-rw-  1 root  tty        5,   2 Mar 16 12:08 ptmx
drwxr-xr-x  2 root  root            0 Mar 16 09:52 pts
crw-rw-rw-  1 root  root       1,   8 Mar 16 09:52 random
crw-rw-r--+ 1 root  netdev    10, 242 Mar 16 09:52 rfkill
lrwxrwxrwx  1 root  root            4 Mar 16 09:52 rtc -> rtc0
crw-------  1 root  root     249,   0 Mar 16 09:52 rtc0
brw-rw----  1 root  disk       8,   0 Mar 16 09:52 sda
brw-rw----  1 root  disk       8,   1 Mar 16 09:52 sda1
crw-rw----  1 root  disk      21,   0 Mar 16 09:52 sg0
```

entries with character C are character device file.
d- directory
b- block device

Exercise : pseudo character driver

- Write a character driver to deal with a pseudo character device
- The pseudo-device is a memory buffer of some size
- The driver what you write must support reading, writing and seeking to this device
- Test the driver functionality by running user-level command such as echo, dd, cat and by writing user lever programs