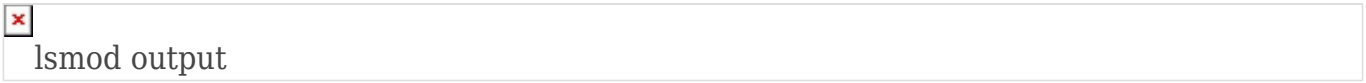
Kernel module is a piece of program which can be loaded or unloaded in the kernel dynamically. It is used to extend the functionality of the kernel without rebooting the system. When we attach a new hardware in our computer we have to install a device driver for it. Device driver is a kernel module which allows the kernel to communicate with the hardware device.

Now why do we need a kernel module? Because we don't want to install or build a new kernel each time we need a new hardware. Consider the hassle of reinstalling the operating system again just to use a newly bought webcam.

Here we'll mainly talk about Linux kernel, not Windows or MacOS. We can easily check the list of installed kernel module in our Linux system. There is a command *lsmod* which returns the list of modules installed in the system. Here is a typical output of *lsmod* command.


lsmod output

Your output might be different from this one as different system might have different modules installed. In this picture the first column is the name of the module, 2nd column is the size of the module and the third column shows which other module is using this module. So as we can see from here, kernel modules has inter-dependency. Some modules can't work without the help of other modules.

There are some other useful command related to kernel module. Let's have a look on those:

- ***lsmod***: list the installed module in the system
- ***insmod:*** install a kernel module in the system. Need *root* privilege to run this command. Usage format: *sudo insmod [MODULE_NAME]*
- ***rmmod:*** remove an installed module from the system. This also requires root privilege. Usage format: *sudo rmmod [MODULE_NAME]*.
- ***modinfo:*** gives detailed information of a kernel module. Usage format: *modinfo [MODULE_NAME]*
- ***modprobe:*** this command inserts or removes a module intelligently. So if there is dependency of one module on another one, say *Module2* is dependent on *Module1*, then this command will first load *Module1* and then *Module2.*

Types of Devices & Modules:

Linux distinguishes all devices into three fundamental classes. These are

Char Device: These are devices that can be accessed as a stream of bytes. These devices can be considered as a file. We can read/write from a char device the same way we read/write a file. The char device behavior is implemented by a char driver. Hence these devices act like a file, so a char driver normally has to implement *open*, *read*, *write* and *close* system calls. Char devices are accessed through a file in the filesystem. We'll discuss more about it.

Block Device: Block devices are normally read from or written to as a block of data, normally 512 bytes or a larger power of two. Like char device, block devices are also accessed through a file in the filesystem. A block driver implements the behavior of a block device.

Network Device: These are devices that can exchange data with other hosts. A network interface driver support this devices.

Major Number & Minor Number:

We already said, char and block devices are accessed through a file in the filesystem. These files are called *device file* and are located in the */dev* directory. *ls* command inside */dev* directory will show all device files in the system. Applications can read from or write to these device files as normal files. Here goes an output of *ls -l* command from my Ubuntu-14.04 system:

```
crw-------   1   root  root     5,    1   Apr 6 14:18      console
crw-rw----   1   root  video   29,    0   Apr 6 14:18      fb0
crw-rw----+  1   root  kvm     10,  232   Apr 6 14:18      kvm
crw-rw-rw-   1   root  root     1,    3   Apr 6 14:18      null
brw-rw----   1   root  disk     1,    0   Apr 6 14:18      ram0
crw-rw-rw-   1   root  root     1,    8   Apr 6 14:18      random
brw-rw----   1   root  disk     8,    0   Apr 6 14:18      sda
crw-rw-rw-   1   root  root     1,    9   Apr 6 14:18      urandom
crw-rw-rw-   1   root  root     1,    5   Apr 6 14:18      zero
```

In the above output, every line starts with a '*c*' or a '*b*'. Here '*c*' stands for character device and '*b*' stands for block device. The last column is the name of the driver. Now the numbers in 5th and 6th column is very important. They are called the *major* number and the *minor* number respectively. The *major* number identifies the driver associated with the device. One *major* number is associated with only one driver. But one driver can handle multiple device. For example, same device driver is used for handling multiple hard drive in a computer.

Each hard drive is a different device, but they have the same device driver. In the above example, */dev/random* and */dev/zero* both are managed by driver 1.

The *minor* number is used by the kernel as a reference to determine which device is being referred to. Device driver developers can get a direct pointer to the device from the kernel, or they can choose to use the minor number as an index in some local array of devices.

So far we've learned what is a kernel module, types of devices and kernel modules. Next check my writing on how to write a kernel module for Linux.