

What is Kernel?

The Linux Kernel is a set of software components called **modules** that work together coherently as a single unit to enable programs, services and applications to run smoothly and efficiently on the system.

Modules are basically **device drivers** used for controlling hardware devices (controller cards, peripheral devices) as well as software components (LVM, File Systems and RAID)

Files related to kernel architecture are installed in **/boot** directory

RHEL allows you to generate and store several custom kernels with varying configuration and required modules included, but only one of them active at a time. Other kernels may be loaded by rebooting the system and selecting an alternate kernel.

Why it is needed to know?

The default kernel installed during installation is usually adequate for most system needs, however, it requires a rebuild when a new functionality is added or removed. The new functionality may be introduced by updating or upgrading the kernel.

How?

We will see in further topics

Operation 1: Checking / Listing / Displaying / Verifying Kernel

To check the version of the kernel (currently running)

#uname -r or #uname -a (see for 3rd column)

Or

#rpm -q kernel

```
[root@server1 ~]# uname -r
3.10.0-514.6.1.el7.x86_64
[root@server1 ~]# █
```

The output indicates that the kernel version currently in use is 3.10.0-514.6.1.el7.x86_64

Or

```
[root@server1 ~]# rpm -q kernel
kernel-3.10.0-229.el7.x86_64
kernel-3.10.0-514.6.1.el7.x86_64
[root@server1 ~]#
```

But it shows all kernels present in the system

Understanding of Kernel version

```
[root@server1 ~]# uname -r
3.10.0-514.6.1.el7.x86_64
```

1st number → shows that this is the third **major version** of the Linux Kernel. The major number changes when there are significant alterations, enhancements and updates to the previous major version.

2nd number → shows that this is the 10th **major revision** of the third major version.

3rd number → shows that this is the zeroth patched version of this kernel to fix minor bugs, security holes and minor enhancements and so on.

4th number → shows that this is the 514 version of Red hat customized kernel

5th number → indicates that this is the sixth build of the 514 version of the Red hat customized kernel

(el7) → indicates that this kernel is for Red Hat Enterprise Linux 7

Understanding Kernel Directory Structure

Kernel files are stored at four different locations in the system directory - These are

1. **/boot**
2. **/proc**
3. **/lib/modules**
4. **/usr/src**

The /boot File System:

The /boot file system is created at system installation time and its purpose is to store kernel related information including current running kernel and associated files. This file system also stores any updated or modified kernel too.

```
# ll /boot
total 27575
-rw-r--r-- 1 root root 64551 Oct 10 16:43 config-2.6.18-92.el5
drwxr-xr-x 2 root root 1024 Jan 17 21:12 grub
-rw----- 1 root root 3104359 Dec 19 18:18 initrd-2.6.18-92.el5.img
drwx----- 2 root root 12288 Dec 19 08:34 lost-found
-rw-r--r-- 1 root root 87586 Oct 10 16:44 symvers-2.6.18-92.el5.gz
-rw-r--r-- 1 root root 903969 Oct 10 16:43 System.map-2.6.18-92.el5
-rw-r--r-- 1 root root 1791572 Oct 10 16:43 vmlinuz-2.6.18-92.el5
```

The output indicates that the current kernel is vmlinuz-2.6.32-304.el7, its boot image is stored in the initrd-2.6-XY.img file and configuration is located in the config-2.6.XY file

Here one more directory is very important i.e. grub2

The key file in this **/boot/grub2** dir is **grub.conf**, which maintains a list of available kernels and defines the default kernel to load at boot time. Here is an excerpt from this file:

```
# ll /boot/grub
```

```
total 304
```

```
-rw-r--r-- 1 root root 63 Dec 12 03:39 device.map
-rw-r--r-- 1 root root 7584 Dec 12 03:39 e2fs_stage1_5
-rw-r--r-- 1 root root 7456 Dec 12 03:39 fat_stage1_5
-rw-r--r-- 1 root root 6720 Dec 12 03:39 ffs_stage1_5
-rw-r--r-- 1 root root 621 Dec 12 03:39 grub.conf
-rw-r--r-- 1 root root 6720 Dec 12 03:39 iso9660_stage1_5
-rw-r--r-- 1 root root 8192 Dec 12 03:39 jfs_stage1_5
lrwxrwxrwx 1 root root 11 Dec 12 03:39 menu.lst -> ./grub.conf
-rw-r--r-- 1 root root 6880 Dec 12 03:39 minix_stage1_5
-rw-r--r-- 1 root root 9248 Dec 12 03:39 reiserfs_stage1_5
-rw-r--r-- 1 root root 32428 Jan 4 2007 splash.xpm.gz
-rw-r--r-- 1 root root 512 Dec 12 03:39 stage1
-rw-r--r-- 1 root root 104988 Dec 12 03:39 stage2
-rw-r--r-- 1 root root 7072 Dec 12 03:39 ufs2_stage1_5
-rw-r--r-- 1 root root 6272 Dec 12 03:39 vstafs_stage1_5
-rw-r--r-- 1 root root 8896 Dec 12 03:39 xfs_stage1_5
```

```
# cat /boot/grub/grub.conf
```

```
.....
default=0
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
title Red Hat Enterprise Linux Server (2.6.18-92.el5)
    root (hd0,0)
    kernel vmlinuz-2.6.18-92.el5 ro root=dev/VolGroup00/lvol1 rhgb quiet
    initrd initrd-2.6.18-92.el5.img
```

The /proc File System:

/proc is a virtual File System that is created in memory. Its contents are created at boot time and destroyed when the system is shutdown. Under this directory, we have current hardware configuration and status information. A directory listing of /proc is as below

```
# ll /proc
.....
dr-xr-xr-x  5 root root 0 Jan 2 21:13 1
dr-xr-xr-x  5 root root 0 Jan 2 21:14 10
dr-xr-xr-x  5 root root 0 Jan 2 21:14 11
.....
dr-xr-xr-x  5 root root 0 Jan 2 22:32 72
dr-xr-xr-x 11 root root 0 Jan 2 22:33 acpi
-r--r--r--  1 root root 0 Jan 2 23:19 buddynfo
dr-xr-xr-x  6 root root 0 Jan 2 22:32 bus
-r--r--r--  1 root root 0 Jan 2 23:19 cmdline
-r--r--r--  1 root root 0 Jan 2 23:19 cpufreq
-r--r--r--  1 root root 0 Jan 2 23:19 crypto
-r--r--r--  1 root root 0 Jan 2 23:19 devices
.....
```

The output displays several files and sub-directories. Some sub-dir names are numerical and contain information about a specific process with process ID matching the sub-dir name. Within each of those sub-dir, there are files and further sub-directories, which includes information such as memory segment specific to that particular process.

Also the data stored in this /proc directories are used by many system utilities including top, uname and vmstat for displaying information in better way.

The /lib/modules Directory:

This directory holds information about kernel modules. Under this directory, there are several sub-dir specific to the kernels available on the systems. For example, ll output on /lib/modules shows there are 4 kernels configurations stored on this current system.

```
# ll /lib/modules
total 32
drwxr-xr-x  6 root root 4096 Jan 17 21:17 2.6.18-92.el5
drwxr-xr-x  6 root root 4096 Jan 17 21:12 2.6.18-92.el5debug
drwxr-xr-x  6 root root 4096 Jan 17 21:10 2.6.18-92.el5PAE
drwxr-xr-x  6 root root 4096 Jan 17 21:09 2.6.18-92.el5xen
```

Now do ll on the default kernel sub-dir

```
# ll /lib/modules/2.6.18-92.el5
total 1140
lrwxrwxrwx 1 root root    43 Dec 19 08:44 build -> ../usr/src/kernels/2.6.18-92.el5-i686
drwxr-xr-x  2 root root  4096 Oct 10 16:44 extra
drwxr-xr-x  9 root root  4096 Dec 19 08:44 kernel
-rw-r--r--  1 root root 245095 Jan 17 21:17 modules.alias
-rw-r--r--  1 root root   69 Jan 17 21:17 modules.ecwmap
-rw-r--r--  1 root root 185925 Jan 17 21:17 modules.dep
-rw-r--r--  1 root root  147 Jan 17 21:17 modules.ieee1394map
-rw-r--r--  1 root root   375 Jan 17 21:17 modules.inputmap
-rw-r--r--  1 root root  2160 Jan 17 21:17 modules.isapnpmap
-rw-r--r--  1 root root   74 Jan 17 21:17 modules.ofmap
-rw-r--r--  1 root root 179642 Jan 17 21:17 modules.pcimap
-rw-r--r--  1 root root   589 Jan 17 21:17 modules.serioimap
```

There are several files and a few sub-dir displayed in the output above.. These files and sub-dir hold module specific information.

One of the key sub-directories is **/lib/modules/2.6.32-XY/kernel/drivers** where modules categorized in groups are stored in various sub-directories as shown in the listing below:


```
# ll /lib/modules/2.6.18-92.el5/kernel/drivers
total 280
drwxr-xr-x  2 root root 4096 Dec 19 08:44 acpi
drwxr-xr-x  2 root root 4096 Dec 19 08:44 ata
drwxr-xr-x  2 root root 4096 Dec 19 08:44 atm
drwxr-xr-x  4 root root 4096 Dec 19 08:44 block
drwxr-xr-x  2 root root 4096 Dec 19 08:44 bluetooth
drwxr-xr-x  2 root root 4096 Dec 19 08:44 cdrom
drwxr-xr-x  7 root root 4096 Dec 19 08:44 char
drwxr-xr-x  2 root root 4096 Dec 19 08:44 cpufreq
drwxr-xr-x  2 root root 4096 Dec 19 08:44 crypto
drwxr-xr-x  2 root root 4096 Dec 19 08:44 dma
drwxr-xr-x  2 root root 4096 Dec 19 08:44 edac
drwxr-xr-x  2 root root 4096 Dec 19 08:44 firewire
drwxr-xr-x  2 root root 4096 Dec 19 08:44 firmware
drwxr-xr-x  2 root root 4096 Dec 19 08:44 hwmon
drwxr-xr-x  5 root root 4096 Dec 19 08:44 ide
drwxr-xr-x  3 root root 4096 Dec 19 08:44 ide
drwxr-xr-x  6 root root 4096 Dec 19 08:44 infiniband
drwxr-xr-x  8 root root 4096 Dec 19 08:44 input
drwxr-xr-x  8 root root 4096 Dec 19 08:44 isdn
drwxr-xr-x  2 root root 4096 Dec 19 08:44 leds
drwxr-xr-x  2 root root 4096 Dec 19 08:44 md
drwxr-xr-x  5 root root 4096 Dec 19 08:44 media
drwxr-xr-x  4 root root 4096 Dec 19 08:44 message
drwxr-xr-x  3 root root 4096 Dec 19 08:44 misc
```

Several module categories such as ata, block, syslogd, cdrom, char, firewire, ide, input, net, parport, pci, scsi, serial, usb and video exist. These categories contain driver modules to control hardware devices associated with them.

The /usr/src Directory:

This directory contains source code for kernel. All information related to building a new kernel from the source code is located here. The /usr/src directory contains two sub-directories as shown below

```
# ll /usr/src
total 16
drwxr-xr-x 5 root root 4096 Aug 16 10:25 kernels
drwxr-xr-x 7 root root 4096 Jan 30 2007 redhat
```

The kernel sub-dir contains the current kernel information and the red hat sub-dir contains the associated source code. Five sub-directories under red hat are created as shown below:

```
# ll /usr/src/redhat
total 40
drwxr-xr-x 2 root root 4096 Jan 30 2007 BUILD
drwxr-xr-x 4 root root 4096 Jan 30 2007 RPMS
drwxr-xr-x 2 root root 4096 Jan 30 2007 SOURCES
drwxr-xr-x 2 root root 4096 Jan 30 2007 SPECS
drwxr-xr-x 2 root root 4096 Jan 30 2007 SRPMS
```


Kernel Parameters

There are hundreds of parameters (kernel related) whose values affect the overall behavior of the kernel, and the running system and running applications inside.

Runtime values of these parameters are stored in files under /proc/sys directory structure. In turn, each of these sub-directories contains the values of parameters as per the category.

-dev → parameters for specific devices connected to the system

-fs → file system configuration (quotas, inodes etc.....)

-kernel → kernel-specific configuration

-net → network configuration

-vm → use of the kernel's virtual memory

To see Linux Kernel Parameters

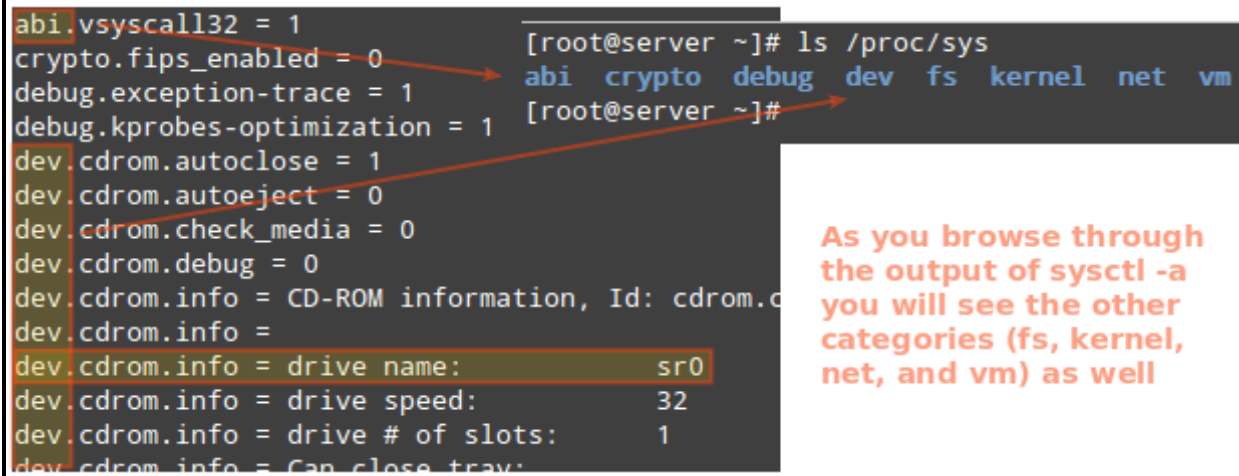
To list all the kernel parameters in the system

```
#sysctl -a
```

The exact number of parameters that can be modified can be viewed with:

```
# sysctl -a | wc -l
```

Let's take a look at the first few lines. Please note that the first characters in each line match the names of the directories inside [/proc/sys](#):



```
abi.vsyscall32 = 1
crypto.fips_enabled = 0
debug.exception-trace = 1
debug.kprobes-optimization = 1
dev.cdrom.autoclose = 1
dev.cdrom.autoeject = 0
dev.cdrom.check_media = 0
dev.cdrom.debug = 0
dev.cdrom.info = CD-ROM information, Id: cdrom.c
dev.cdrom.info =
dev.cdrom.info = drive name: sr0
dev.cdrom.info = drive speed: 32
dev.cdrom.info = drive # of slots: 1
dev.cdrom.info = Can close tray:

[root@server ~]# ls /proc/sys
abi  crypto  debug  dev  fs  kernel  net  vm
```

As you browse through the output of sysctl -a you will see the other categories (fs, kernel, net, and vm) as well

For example, the highlighted line:

```
dev.cdrom.info = drive name: sr0
```

Indicates that `sr0` is an alias for the optical drive. In other words, that is how the kernel “sees” that drive and uses that name to refer to it.

Based on what we have explained so far, it is easy to see that the name of a parameter matches the directory structure inside `/proc/sys` where it can be found.

For example:

```
dev.cdrom.autoclose → /proc/sys/dev/cdrom/autoclose
```

```
net.ipv4.ip_forward → /proc/sys/net/ipv4/ip_forward
```

And can be altered on the fly by changing associated files, which will then remain effective until either the value is readjusted or the system is rebooted.

Set or Modify Linux Kernel Parameters

To change the kernel parameter we can use either

sysctl or echo

To set the value for a kernel parameter we can use `sysctl`, but using the `-w` option and followed by the parameter’s name, the equal sign, and the desired value.

Another method consists of using `echo` to overwrite the file associated with the parameter

```
# echo 0 > /proc/sys/net/ipv4/ip_forward  
# sysctl -w net.ipv4.ip_forward=0
```

It is important to note that kernel parameters that are set using `sysctl` will only be enforced during the current session and will disappear when the system is rebooted.

So above methods is only temporary one.

To make permanent change of Kernel parameter

To set these values permanently, edit `/etc/sysctl.conf` with the desired values. For example, to disable packet forwarding in `/etc/sysctl.conf` make sure this line appears in the file:

```
net.ipv4.ip_forward=0
```

Then run following command to apply the changes to the running configuration.

```
# sysctl -p
```

Another approach:

A better and easier way to set individual runtime parameters is using .conf files inside `/etc/sysctl.d`, grouping them by categories.

For example, instead of setting `net.ipv4.ip_forward=0` and `net.ipv4.icmp_echo_ignore_all=1` in `/etc/sysctl.conf`, we can create a new file named `net.conf` inside `/etc/sysctl.d`:

```
# echo "net.ipv4.ip_forward=0" > /etc/sysctl.d/net.conf
# echo "net.ipv4.icmp_echo_ignore_all=1" >> /etc/sysctl.d/net.conf
```

If you choose to use this approach, do not forget to remove those same lines from `/etc/sysctl.conf`.

Kernel Modules

In windows, as we know drivers, the same thing in Linux called as modules.

- Kernel modules are essentially chunks of code that we can load or unload into the kernel on demand while operating system is running.
- Modules commonly come into the form of drivers for hardware and file systems.
- In the past, if we want to use kernel module then we need to recompile the whole Linux kernel to include module for e.g. Network card module
- But now we can load modules as per the demand without recompiling the kernel just by using commands.
- All the kernel modules are stored in /lib/modules directory
- Navigate to dir /lib/module and do ls, its the same as uname -r

```
#cd /lib/modules
```

```
#ls
```

```
2.6.32-XY.....
```

```
#uname -r
```

```
2.6.32-XY....
```

To search all kernel modules in the system using find command

```
#find / -name *.ko
```

Note: observe that all the modules listed in the above location may be supported or currently loaded modules.

Listing Kernel modules

To list currently loaded modules

```
#lsmod
```

```
#lsmod | more
```

Module	Size	Used by	
tcp_lp	12665	0	
nls_utf8	12557	1	
isofs	39844	1	
bnep	19704	2	
bluetooth	372662	7 bnep	
rftkill	26536	3 bluetooth	
fuse	87741	3	
coretemp	13435	0	
ext4	562391	1	
crct10dif_pclmul	14289	0	
crc32_pclmul	13113	0	
crc32c_intel	22079	0	
ghash_clmulni_intel	13259	0	
snd_ens1371	25243	3	
aesni_intel	52846	0	
mbcache	14958	1 ext4	
jbd2	102940	1 ext4	
lrw	13286	1 aesni_intel	
snd_rawmidi	30824	1 snd_ens1371	
gf128mul	14951	1 lrw	
glue_helper	13990	1 aesni_intel	
ablk_helper	13597	1 aesni_intel	
cryptd	20359	3 ghash_clmulni_intel,aesni_intel,ablk_helper	
snd_ac97_codec	130476	1 snd_ens1371	
ppdev	17671	0	
ac97_bus	12730	1 snd_ac97_codec	
snd_seq	63074	0	
snd_seq_device	14497	2 snd_seq,snd_rawmidi	
snd_pcm	103996	2 snd_ac97_codec,snd_ens1371	
raid1	35530	1	
vmw_balloon	13415	0	

This indicate a numerical value that shows the other modules depend on the listed module, if few case it shows name also.

Its the space consumed by module in memory

Name of kernel module

To check particular module is loaded or not

#lsmod |grep -i cdrom

#lsmod |grep -i fat

#lsmod |grep -i e1000

Think

```
[root@rhel3 ~]# lsmod | grep e1000
e1000                149270  0
[root@rhel3 ~]# ifconfig
eno16777736: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 192.168.1.244  netmask 255.255.255.0  broadcast 192.168.1.255
    ether 00:0c:29:3c:a0:5d  txqueuelen 1000  (Ethernet)
    RX packets 1947  bytes 131741 (128.6 KiB)
    RX errors 0  dropped 3  overruns 0  frame 0
    TX packets 116  bytes 10666 (10.4 KiB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
    inet 127.0.0.1  netmask 255.0.0.0
    loop txqueuelen 0  (Local Loopback)
    RX packets 4  bytes 340 (340.0 B)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 4  bytes 340 (340.0 B)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Remove Kernel modules

```
[root@rhel3 ~]# rmmod e1000
[root@rhel3 ~]# ifconfig
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
    inet 127.0.0.1  netmask 255.0.0.0
    loop txqueuelen 0  (Local Loopback)
    RX packets 8  bytes 680 (680.0 B)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 8  bytes 680 (680.0 B)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Now no longer we have Ethernet devices because of unavailability of network modules.

```
[root@rhel3 ~]# lsmod | grep e1000
[root@rhel3 ~]#
```


Loading Kernel modules

We can add kernel modules back into kernel dynamically just by typing “modprobe”

```
[root@rhel3 ~]# modprobe e1000
[root@rhel3 ~]# lsmod | grep e1000
e1000                149270  0
[root@rhel3 ~]# ifconfig
enol6777736: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.244 netmask 255.255.255.0 broadcast 192.168.1.255
    ether 00:0c:29:3c:a0:5d txqueuelen 1000 (Ethernet)
    RX packets 251 bytes 1620 (1.5 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 18 bytes 2569 (2.5 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    loop txqueuelen 0 (Local Loopback)
    RX packets 12 bytes 1020 (1020.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 12 bytes 1020 (1020.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

[root@rhel3 ~]#
```

More info about Kernel modules

We can also find more detailed information about the kernel modules.

Suppose I am troubleshooting the network card driver and i want to know the version of the driver then i can use “modinfo”

```
[root@rhel3 ~]# modinfo e1000 | more
```

```

filename:      /lib/modules/3.10.0-229.el7.x86_64/kernel/drivers/net/ethernet/intel/e1000/e1000.ko
version:      7.3.21-k8-NAPI
license:      GPL
description:   Intel(R) PRO/1000 Network Driver
author:       Intel Corporation, <linux.nics@intel.com>
rhelversion:  7.1
srcversion:   92BCA75ABE6D82F424CAE7D
alias:        pci:v00008086d00002E6Esv*sd*bc*sc*i*
alias:        pci:v00008086d000010B5sv*sd*bc*sc*i*
alias:        pci:v00008086d00001099sv*sd*bc*sc*i*
alias:        pci:v00008086d0000108Asv*sd*bc*sc*i*
alias:        pci:v00008086d0000107Csv*sd*bc*sc*i*
alias:        pci:v00008086d0000107Bsv*sd*bc*sc*i*
alias:        pci:v00008086d0000107Asv*sd*bc*sc*i*
alias:        pci:v00008086d00001079sv*sd*bc*sc*i*
alias:        pci:v00008086d00001078sv*sd*bc*sc*i*
alias:        pci:v00008086d00001077sv*sd*bc*sc*i*
alias:        pci:v00008086d00001076sv*sd*bc*sc*i*
alias:        pci:v00008086d00001075sv*sd*bc*sc*i*
alias:        pci:v00008086d00001028sv*sd*bc*sc*i*
alias:        pci:v00008086d00001027sv*sd*bc*sc*i*
alias:        pci:v00008086d00001026sv*sd*bc*sc*i*
alias:        pci:v00008086d0000101Esv*sd*bc*sc*i*
alias:        pci:v00008086d0000101Dsv*sd*bc*sc*i*
alias:        pci:v00008086d0000101Asv*sd*bc*sc*i*
alias:        pci:v00008086d00001019sv*sd*bc*sc*i*
alias:        pci:v00008086d00001018sv*sd*bc*sc*i*
alias:        pci:v00008086d00001017sv*sd*bc*sc*i*
alias:        pci:v00008086d00001016sv*sd*bc*sc*i*
alias:        pci:v00008086d00001015sv*sd*bc*sc*i*
alias:        pci:v00008086d00001014sv*sd*bc*sc*i*

```

This shows that the driver file and its version and so on.

Blacklist Kernel modules

To blacklist or disable any module to load, we can update the file
/etc/modprobe.d/blacklist.conf