

KERNEL MANAGEMENT

The kernel is the operating system .It performs all the core task like managing memory and disk access it will connect to all the hardware that makes your system. It gives you the multitasking and multi user support .It handles all the communication with all the devices like CD ROM USB drive .Basically user sends the request signal that go through the kernel to the device .Based on different different hardware the configuration of the kernel will very .Suppose you have to add a new device to the system then you have to change the kernel support for the specific devices. you can download the binary version of the kernel or you can download the source code and compile it. Its now the job of the system administrator to worry for the code of the kernel but you must know how to add and remove kernel module and detect if any kernel module not working or malfunctioning .And he must know how to compile and add new kernel to the system

kernel sets up several presses some process are internal to the kernel. We can see the internal process the with this command

=> **ps aux | egrep '[**

```
[root@localhost ~]# ps aux | egrep '\['
root      2  0.0  0.0      0   0 ?        S    16:11   0:00 [kthreadd]
root      3  0.0  0.0      0   0 ?        S    16:11   0:00 [ksoftirqd/0]
root      5  0.0  0.0      0   0 ?        S<   16:11   0:00 [kworker/0:0H]
root      6  0.0  0.0      0   0 ?        S    16:11   0:00 [kworker/u256:0]
root      7  0.0  0.0      0   0 ?        S    16:11   0:00 [migration/0]
root      8  0.0  0.0      0   0 ?        S    16:11   0:00 [rcu_bh]
root      9  0.0  0.0      0   0 ?        R    16:11   0:02 [rcu_sched]
root     10  0.0  0.0      0   0 ?        S<   16:11   0:00 [lru-add-drain]
root     11  0.0  0.0      0   0 ?        S    16:11   0:00 [watchdog/0]
root     13  0.0  0.0      0   0 ?        S    16:11   0:00 [kdevtmpfs]
root     14  0.0  0.0      0   0 ?        S<   16:11   0:00 [netns]
root     15  0.0  0.0      0   0 ?        S    16:11   0:00 [khungtaskd]
root     16  0.0  0.0      0   0 ?        S<   16:11   0:00 [writeback]
root     17  0.0  0.0      0   0 ?        S<   16:11   0:00 [kintegrityd]
root     18  0.0  0.0      0   0 ?        S<   16:11   0:00 [bioaset]
root     19  0.0  0.0      0   0 ?        S<   16:11   0:00 [bioaset]
root     20  0.0  0.0      0   0 ?        S<   16:11   0:00 [bioaset]
root     21  0.0  0.0      0   0 ?        S<   16:11   0:00 [kblockd]
root     22  0.0  0.0      0   0 ?        S<   16:11   0:00 [md]
root     23  0.0  0.0      0   0 ?        S<   16:11   0:00 [edac-poller]
root     24  0.0  0.0      0   0 ?        S<   16:11   0:00 [watchdogd]
root     30  0.0  0.0      0   0 ?        S    16:11   0:00 [kswapd0]
root     31  0.0  0.0      0   0 ?        SN   16:11   0:00 [ksmd]
```

this process are all kernel process. Some of these processes are very important for the system administrator to know.

For example

- ‘**kthreadd**’ manages the kernel thread

- ‘**md/0**’ manages the raid subsystem

- ‘**kswapd**’ manages the swap space available for the system

[this generally doesn't impact the system administrator. Some times the system can misbehave and can occur memory overflow if it happens you will see the kswapd process on the top of the process list which you can find in the ‘**top**’ command]

Kernel Modules:

kernel modules lies in the directory under '**/lib/modules**'
=> **cd /lib/modules**

```
[root@localhost modules]# ls
3.10.0-957.el7.x86_64
[root@localhost modules]#
[root@localhost modules]#
```

```
tanvirrahman@pop-os:/lib/modules
> ls
5.0.0-21-generic
tanvirrahman@pop-os:/lib/modules
>
```

in this directory there can be multiple directory .each directory for each kernel .Under this directory there are a lot of files that can be for different different devices.

```
[root@localhost 3.10.0-957.el7.x86_64]# ls
build  modules.alias      modules.builtin    modules.dep.bin    modules.modesetting modules.softdep    source  weak-updates
extra  modules.alias.bin  modules.builtin.bin modules.devname     modules.networking  modules.symbols    updates
kernel modules.block      modules.dep        modules.drm        modules.order       modules.symbols.bin vdsso
```

```
tanvirrahman@pop-os:/lib/modules/5.0.0-21-generic
> ls
build  kernel  modules.alias      modules.builtin    modules.dep    modules.devname  modules.softdep  modules.symbols.bin  vdsso
initrd misc    modules.alias.bin  modules.builtin.bin modules.dep.bin modules.order     modules.symbols  updates

tanvirrahman@pop-os:/lib/modules/5.0.0-21-generic
> 
```

kernel module varies depending on the hardware manufacturer modules loaded by the kernel at the boot time. you can manage which driver will be loaded and which driver will not.

To find which model is loaded we use the '**lsmod**' command.[you have to be a root user for that]

=>**lsmod**

```
[root@localhost 3.10.0-957.el7.x86_64]#  
[root@localhost 3.10.0-957.el7.x86_64]# lsmod  
Module                Size  Used by  
ip6t_rpfilter          12595   1  
ipt_REJECT             12541   2  
nf_reject_ipv4         13373   1 ipt_REJECT  
ip6t_REJECT            12625   2  
nf_reject_ipv6         13717   1 ip6t_REJECT  
xt_conntrack           12760  11  
ip_set                 45644   0  
nfnetlink              14490   1 ip_set  
ebtable_nat            12807   1  
ebtable_broute         12731   1  
bridge                151336   1 ebtable_broute  
stp                   12976   1 bridge  
llc                   14552   2 stp,bridge  
ip6table_nat           12864   1  
nf_conntrack_ipv6      18935   7  
nf_defrag_ipv6         35104   1 nf_conntrack_ipv6  
nf_nat_ipv6            14131   1 ip6table_nat
```

To add any kernel modules we use the '**modprobe**' command for example if we want to add the blue tooth module to the system this command is used

=>**modprobe bluetooth**

```
[root@localhost 3.10.0-957.el7.x86_64]#
[root@localhost 3.10.0-957.el7.x86_64]# modprobe bluetooth
[root@localhost 3.10.0-957.el7.x86_64]#
```

Synthetic File System

There are two different type of file system. one is the the real file system that lies on some disk. like **‘/root’**, **‘/boot’** this are the real file system.

There are another file system that are fake file system that are created by the kernel .Its useful for the system administrator to access the internal variable within the kernel. one of the file system is **‘/proc’**. This is meant for process information. Inside the directory there are a lot of directory with numbers.

```
tanvirrahman@pop-os:/proc
> ls
1      1082 1234 1370 151 1771 1981 2072 2168 24 3 42 514 838 diskstats locks sysrq-trigger
10     1098 1237 1372 1516 1777 1994 2073 217 240 30 43 52 845 dma mdstat sysvipc
1001   11 1243 1373 152 1780 1996 2074 2179 2408 31 437 53 846 driver meminfo thread-self
1002   1101 1250 1374 16 1781 2 2079 2180 241 32 44 539 868 execdomains misc timer_list
1004   1130 1291 1375 1617 1794 20 208 2188 242 327 45 54 869 fb modules tty
1011   1143 13 1380 163 1798 2000 2081 2197 243 33 452 55 884 filesystems mounts uptime
1019   1147 1306 1381 1647 18 2002 2085 22 244 3362 453 56 897 fs mtrr version
1023   1149 1311 1383 1654 1800 2004 2089 2202 2447 34 46 581 898 interrupts net version_signature
1024   1150 1314 1385 1666 1812 2015 209 2268 2465 35 460 586 9 iomem pagetypeinfo vmallocinfo
1025   1151 1318 1387 1673 1815 2019 2095 2280 2468 350 463 59 902 ioports partitions vmnet
1028   1152 1319 1389 1680 183 2027 2098 23 2483 36 47 6 acpi irq pressure vmstat
1034   1153 1335 1395 1683 184 2036 21 2300 25 37 477 60 asound kallsyms sched_debug zoneinfo
1035   1154 1338 1398 1690 19 2040 2102 2312 2524 378 48 61 buddyinfo kcore schedstat
1036   1155 1341 14 17 1911 2045 2106 2326 2530 379 49 697 bus keys scsi
1037   1180 1345 1400 172 1914 2048 2108 2349 2558 38 5 699 cgroups key-users self
1047   12 1357 1412 1725 1919 2053 2111 236 2567 39 50 7 cmdline kmsg slabinfo
1055   1201 1358 1454 1733 1935 2058 2115 237 26 396 502 8 consoles kpagecgroup softirqs
1073   1220 1362 15 1766 1940 2062 2116 238 27 4 505 825 cpuinfo kpagecount stat
1079   1221 1363 150 1768 1942 2066 2118 2385 28 40 51 835 crypto kpageflags swaps
1080   1232 1368 1503 1769 1965 2070 2133 239 29 41 512 837 devices loadavg sys
```

these are all process id. process id 1 is the init process. so if you go to the **‘/proc/1’** it will show you the detail of that process.

```
root@pop-os:/proc/1
> ls
attr      cmdline  environ io      mem      ns        pagemap  sched    smaps_rollup  syscall  wchan
autogroup comm      exe      limits mountinfo numa_maps patch_state schedstat  stack    task
auxv      coredump_filter fd        loginuid mounts    oom_adj   personality sessionid  stat      timers
cgroup    cpuset    fdinfo   map_files mountstats oom_score projid_map setgroups  statm     timerslack_ns
clear_refs cwd        gid_map  maps      net       oom_score_adj root      smaps      status     uid_map

root@pop-os:/proc/1
> |
```

Another synthetic file system is **‘/sys’** although it works with the devices but the main target is the same which is accessing the settings of the kernel.