netstat

network statistics

netstat

netstat (network statistics) is a command line tool for monitoring network connections both incoming and outgoing as well as viewing routing tables, interface statistics.

- Unix-like Operating Systems and also available on Windows OS as well
- Useful in terms of network troubleshooting and performance measurement.
- basic network service debugging tools
- Tells what ports are open and whether any programs are listening on ports.
- Very import tool for linux network administrators and system administrators
- troubleshoot their network related problems and determine network traffic performance.

```
netstat
Active Internet connections  (w/o servers)
Proto Recv-Q Send-Q Local Address Foreign Address  State
tcp        0        0         prolin1:31027  prolin1:5500     TIME_WAIT
tcp        4        0         prolin1l:1521   applin1:40205    ESTABLISHED
tcp        0        0          prolin1l:1522  prolin1:39957    ESTABLISHED
tcp        0        0         prolin1l:3938   prolin1:31017    TIME_WAIT
tcp        0        0         prolin1l:1521   prolin1:21545    ESTABLISHED
```

- The above output goes on to show all the open sockets
- connection has to have a source and a destination, called *local* and *remote* address. The end points could be on the same server; or on different servers.
- the programs connect to the ~~same server. For instance~~, if two processes communicate among each other, the local and remote addresses will be the same, as you can see in the first line – the local and remote addresses are both the sever “prolin1”.
- However, the processes communicate over a *port*, which will be different. This port is shown next to the host name after the “:” (colon) mark
- The user program ~~sends the data to be~~ sent across the socket to a queue and the receiver reads from a queue at the remote end.

1. The leftmost column “ **Proto**” shows the type of the connection – tcp in this case.
2. The column **Recv-Q** ~~shows the bytes of data~~ in the ~~queue to be sent to the~~ user program that established the c
   This value should be as close to 0 as possible. In busy servers this value will be more than 0 but shouldn’t be v
   The **Send-Q** column denotes the bytes in the queue to be sent to the remote program, i.e. the remote program
   acknowledged receiving it
3. **Local Address** is source of the connection and the port number of the program.

4. **Foreign Address** is the destination host and port number. In the first line, both the source and destination are o
   host: proline 1. The connection is simply waiting. The second line shows and established connection between p
   proiln1 going to the port 40205 of the host applin1.

5. The column **State** shows the status of the connection. Here are some common values.
   - **ESTABLISHED** – that the connection has been established. It does not mean that any data is flowing be
     endpoints; merely that the end points have talked to each other.
   - **CLOSED** – the connection has been closed, i.e. not used now.
   - **TIME_WAIT** – the connection is being closed but there are still packets in the network that are being har
   - **CLOSE_WAIT** – the remote end has shutdown and has asked to close the connection.

The -p option shows the process information

```
netstat -p
Proto  Recv-Q Send-Q Local Address Foreign Address State       PID/Program name
tcp       0      0 prolin1:1521   prolin1:33303   ESTABLISHED  1327/oraclePROPRD1
tcp       0      0 prolin1:1521   applin1:51324   ESTABLISHED 13827/oraclePROPRD1
tcp       0      0 prolin1:1521   prolin1:33298   ESTABLISHED  32695/tnslsnr
tcp       0      0 prolin1:1521   prolin1:32544   ESTABLISHED  15251/oracle+ASM
tcp       0      0 prolin1:1521   prolin1:33331   ESTABLISHED  32695/tnslsnr
```

This clearly shows the process IP and the process name in the last column.

To find out the network statistics for various interfaces, use the -i option

```
netstat -i
Kernel  Interface table
Iface      MTU Met    RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
eth0      1500   0  6860659      0          0             0       2055833      0             0     0        BMRU
eth8      1500   0     2345        0          0             0        833            0             0     0        BMRU
lo       16436    0 14449079     0          0             0       14449079     0             0     0        LRU
```

This shows the different interfaces present in the server (eth0, eth8, etc.) and the metrics associated with the interface.

- **RX-OK** shows the number of packets successfully received (for this interface)
- **RX-ERR** shows number of errors
- **RX-DRP** shows packets dropped
- **RX-OVR** shows packets overrun

The next sets of columns (TX-OK, TX-ERR, etc.) show the corresponding stats for send data.

Flg column is a composite value of the property of the interface. Each letter indicates a specific pro
explanation of the letters.

B – Broadcast

M – Multicast

R – Running

U – Up

O – ARP Off

P – Point to Point Connection

L – Loopback

m – Master

You can use the –interface (note: there are *two* hyphens, not one) option to display the same for a specific interface.

```
netstat --interface=eth0
Kernel Interface table
Iface       MTU Met    RX-OK  RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR  TX-DRP TX-OVR Flg
eth0       1500   0     277903459      0      0               0       170897632      0      0           0       BMsRU
```

netstat -i -e

```
Kernel Interface table
eth0      Link encap:Ethernet   HWaddr 00:13:72:CC:EB:00
          inet addr:10.14.106.0   Bcast:10.14.107.255   Mask:255.255.252.0
          inet6 addr: fe80::213:72ff:fecc:eb00/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:6861068 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2055956 errors:0 dropped:0 overruns:0  carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3574788558 (3.3 GiB)  TX bytes:401608995 (383.0 MiB)
          Interrupt:169
```

Does the output seem familiar? It should; it's the same as the output of the **ifconfig.**

If you'd rather see the output showing IP addresses instead of host names, use the -n option.

The **-s option shows the summary statistics of each protocol,** rather than showing the details of each connection. This can be combined with the protocol specific flag. For instance **-u shows the stats related to the UDP protocol.**

The -s option shows the summary statistics of each protocol, rather than showing the details of each connection. This can be combined with the protocol specific flag. For instance -u shows the stats related to the UDP protocol.

```
netstat -s -u
Udp:
    12764104 packets received
    600849 packets to unknown port received.
    0 packet receive errors
    13455783 packets sent
```

Similarly, to see the stats for tcp, use -t and for raw, -r.

One of the really useful options is the display of the routing table, the -r option.

```
#  netstat -r
Kernel  IP routing table
Destination     Gateway         Genmask          Flags   MSS Window        irtt      Iface
10.20.191.0      *              255.255.255.128    U      0         0        0       bond0
172.22.13.0      *              255.255.255.0      U      0         0        0        eth9
169.254.0.0      *              255.255.0.0        U      0         0        0        eth9
default        10.20.191.1      0.0.0.0            UG     0         0        0        bond0
```

The second column of netstat output– **Gateway**–shows the gateway to which the routing entry points. If no gateway is used, an asterisk is printed instead. The third column– **Genmask**–shows the “generality” of the route, i.e., the **network mask for this route**.

The fourth column, **Flags**, displays the following flags that describe the route:

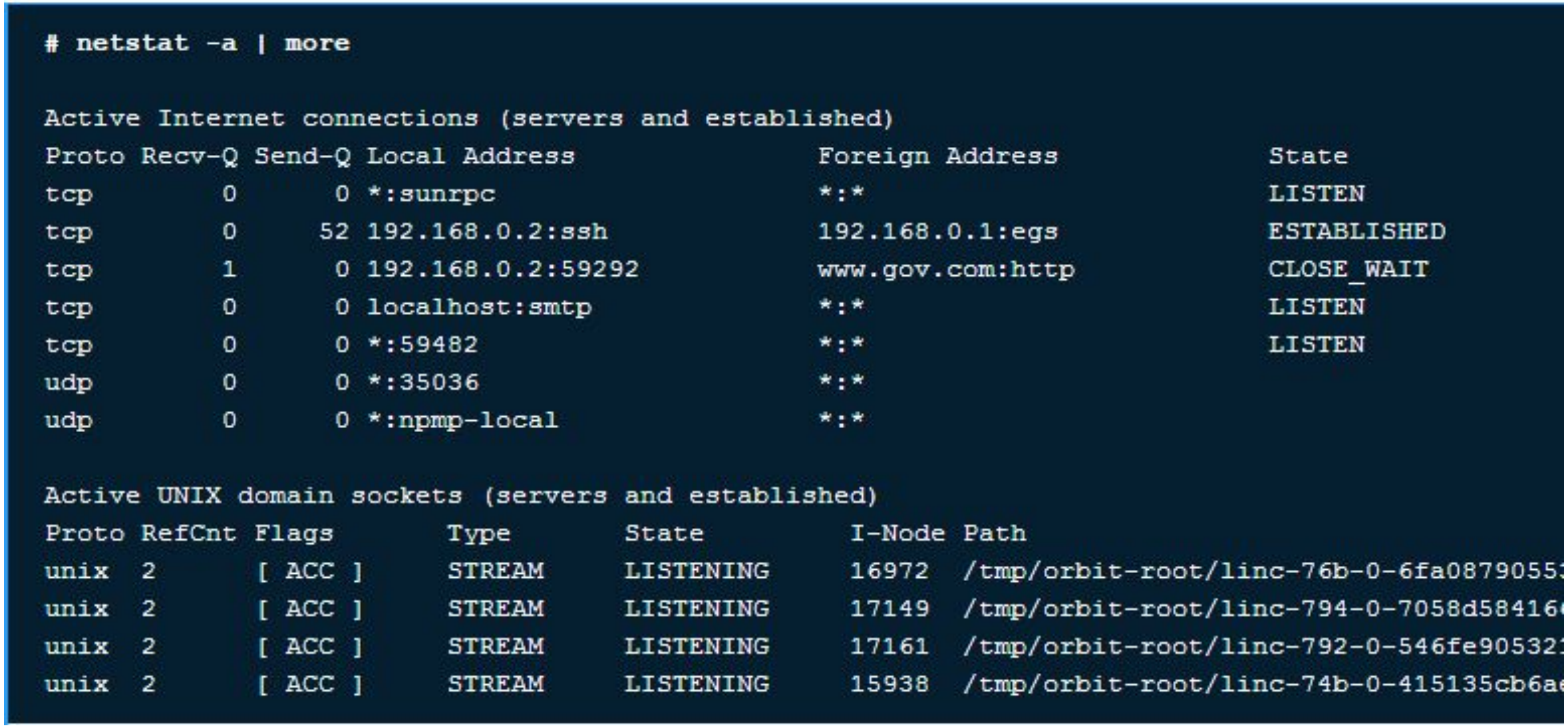
- **G** means the route uses a gateway.
- **U** means the interface to be used is up (available).
- **H** means only a single host can be reached through the route. For example, this is the case for the loopback entry 127.0.0.1.
- **D** means this route is dynamically created.
- **!** means the route is a reject route and data will be dropped.

The next three columns show the **MSS**, **Window,** and **irtt** that will be applied to TCP connections established via this route.

- **MSS** stands for Maximum Segment Size – the size of the largest datagram for transmission via this route.
- **Window** is the maximum amount of data the system will accept in a single burst from a remote host for this route.
- **irtt** stands for Initial Round Trip Time.
- The TCP protocol has a built-in reliability check. If a data packet fails during transmission, it's re-transmitted. The protocol keeps track of how long it takes for the data to reach the destination and acknowledgement to be received. If the acknowledgement does not come within that timeframe, the packet is retransmitted. The amount of time the protocol has to wait before re-transmitting is set for the interface once (which can be changed) and that value is known as *initial round trip time*. A value of 0 means the default value is used.

Finally, the last field displays the network interface that this route will use.

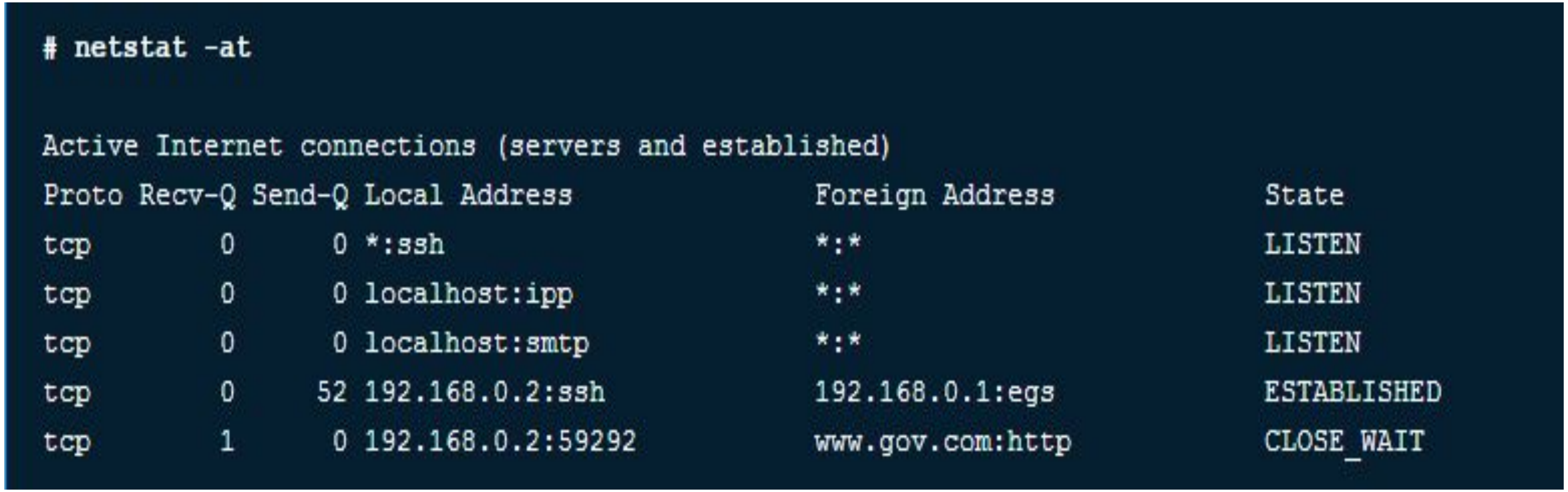
Listing all the LISTENING Ports of TCP and UDP connections



```
# netstat -a | more

Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address               Foreign Address             State
tcp        0      0 *:sunrpc                    *:*                         LISTEN
tcp        0     52 192.168.0.2:ssh             192.168.0.1:egs             ESTABLISHED
tcp        1      0 192.168.0.2:59292           www.gov.com:http            CLOSE_WAIT
tcp        0      0 localhost:smtp              *:*                         LISTEN
tcp        0      0 *:59482                     *:*                         LISTEN
udp        0      0 *:35036                     *:*
udp        0      0 *:npmp-local                *:*

Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node Path
unix  2      [ ACC ]     STREAM     LISTENING     16972  /tmp/orbit-root/linc-76b-0-6fa0879055
unix  2      [ ACC ]     STREAM     LISTENING     17149  /tmp/orbit-root/linc-794-0-7058d58416
unix  2      [ ACC ]     STREAM     LISTENING     17161  /tmp/orbit-root/linc-792-0-546fe90532
unix  2      [ ACC ]     STREAM     LISTENING     15938  /tmp/orbit-root/linc-74b-0-415135cb6a
```

Listing TCP Ports connections

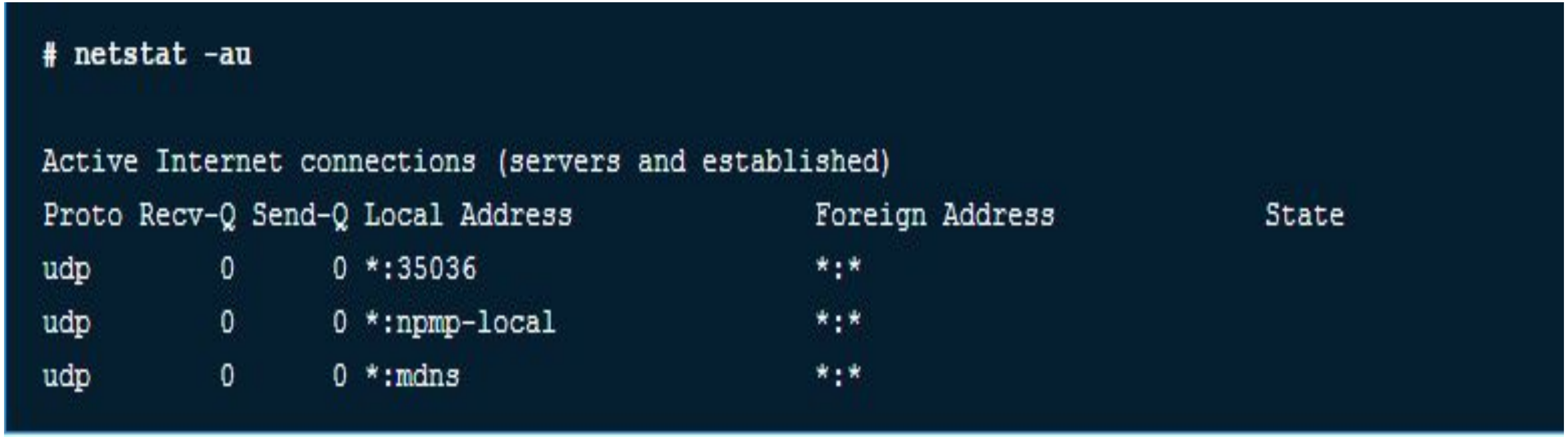
Listing only TCP (Transmission Control Protocol) port connections using netstat -at.



```
# netstat -at

Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address               Foreign Address             State
tcp        0      0 *:ssh                       *:*                         LISTEN
tcp        0      0 localhost:ipp               *:*                         LISTEN
tcp        0      0 localhost:smtp              *:*                         LISTEN
tcp        0     52 192.168.0.2:ssh             192.168.0.1:egs             ESTABLISHED
tcp        1      0 192.168.0.2:59292           www.gov.com:http            CLOSE_WAIT
```

Listing UDP Ports connections

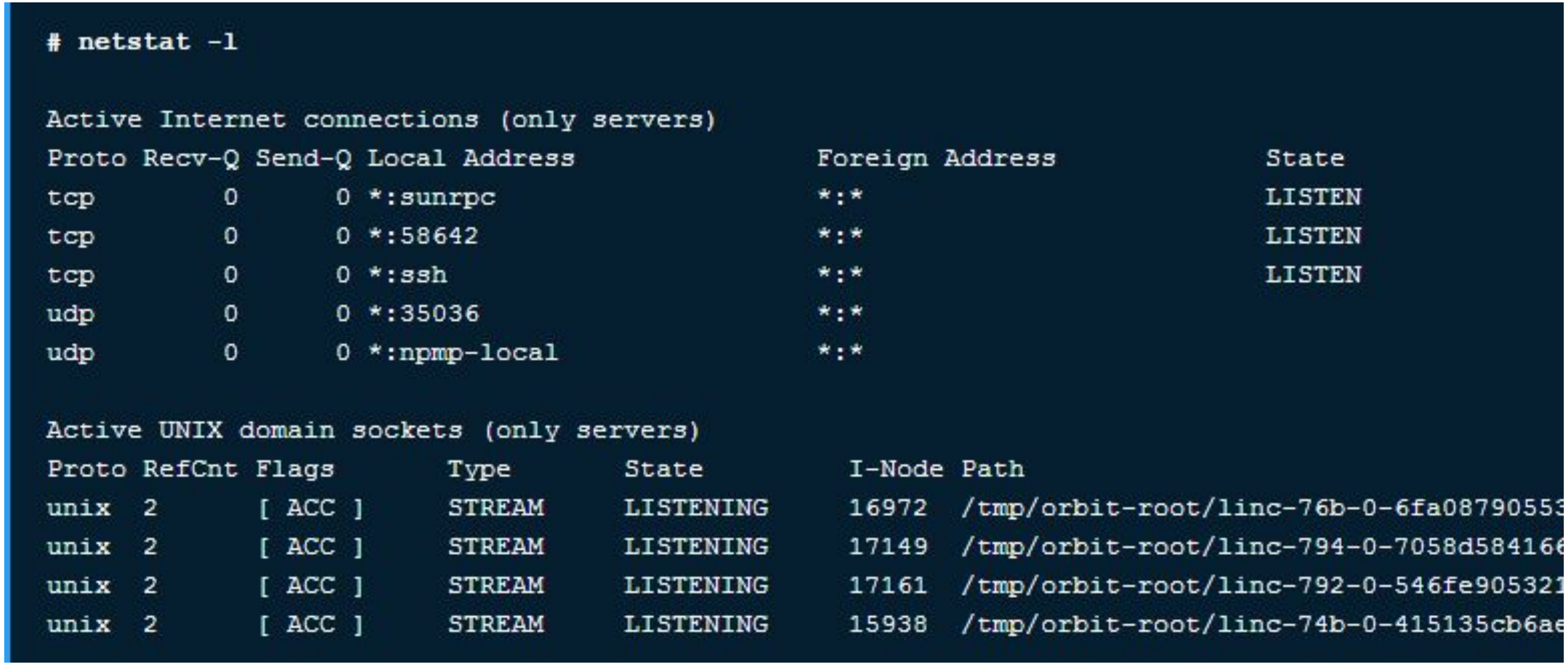
Listing only UDP (User Datagram Protocol) port connections using netstat -au.



```
# netstat -au

Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address               Foreign Address             State
udp        0      0 *:35036                     *:*
udp        0      0 *:npmp-local                *:*
udp        0      0 *:mdns                      *:*
```

Listing all LISTENING Connections

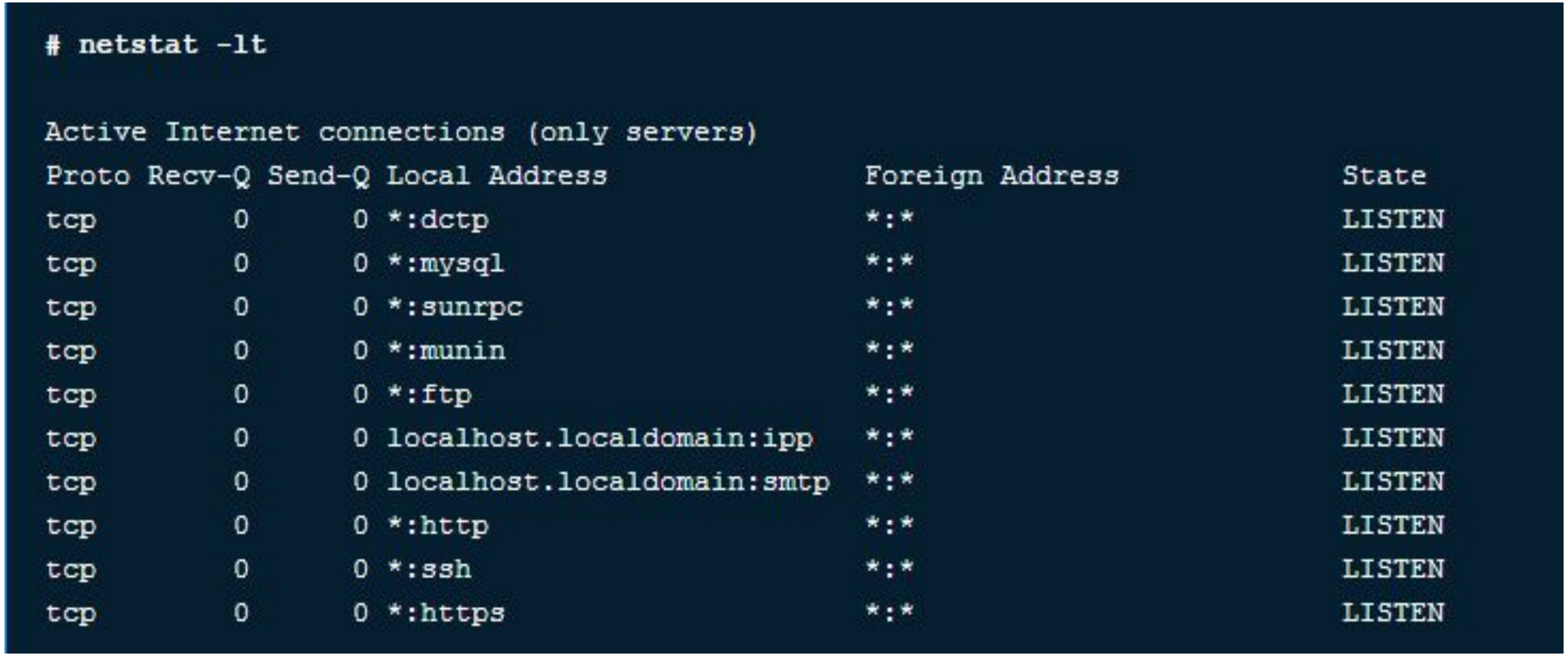
Listing all active listening ports connections with netstat -l.



```
# netstat -l

Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address               Foreign Address             State
tcp        0      0 *:sunrpc                    *:*                         LISTEN
tcp        0      0 *:58642                     *:*                         LISTEN
tcp        0      0 *:ssh                       *:*                         LISTEN
udp        0      0 *:35036                     *:*
udp        0      0 *:npmp-local                *:*

Active UNIX domain sockets (only servers)
Proto RefCnt Flags       Type       State         I-Node Path
unix  2      [ ACC ]     STREAM     LISTENING     16972  /tmp/orbit-root/linc-76b-0-6fa08790553
unix  2      [ ACC ]     STREAM     LISTENING     17149  /tmp/orbit-root/linc-794-0-7058d584166
unix  2      [ ACC ]     STREAM     LISTENING     17161  /tmp/orbit-root/linc-792-0-546fe905321
unix  2      [ ACC ]     STREAM     LISTENING     15938  /tmp/orbit-root/linc-74b-0-415135cb6ae
```

Listing all TCP Listening Ports

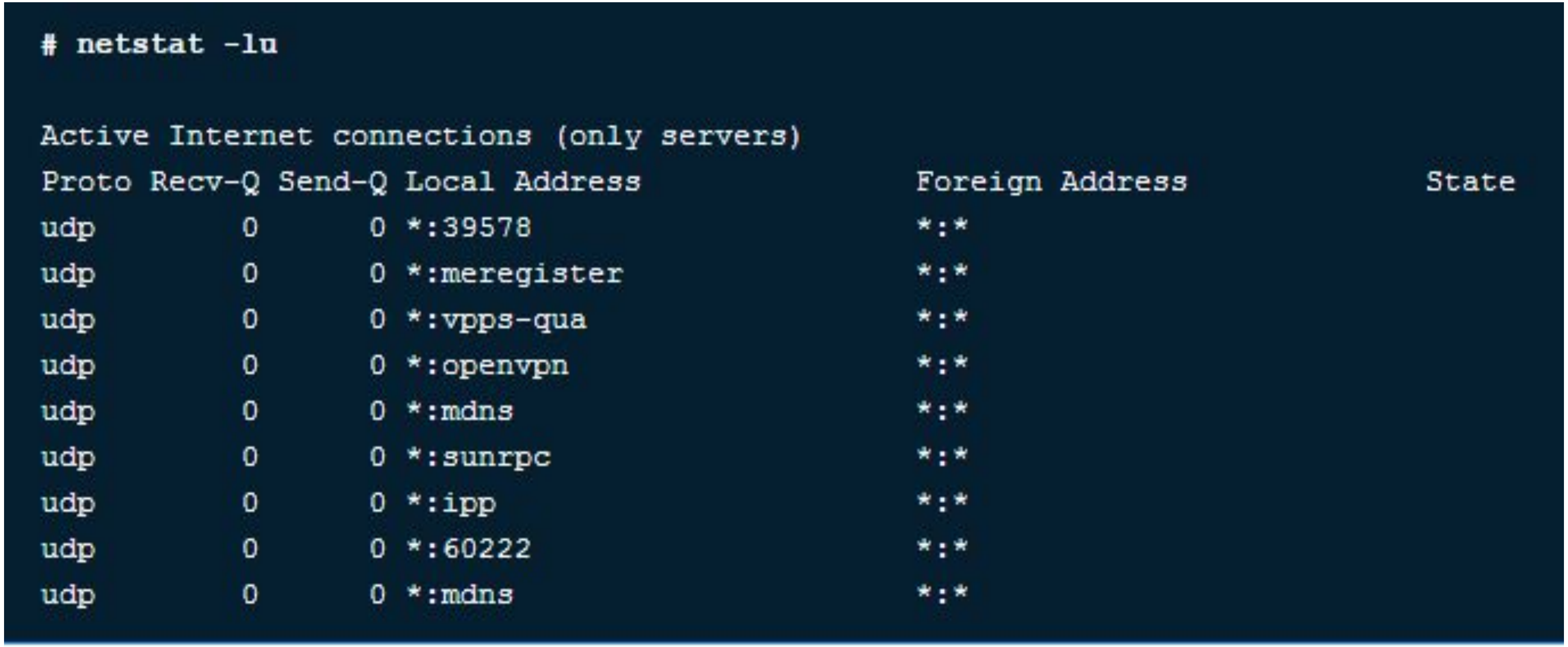
Listing all active listening TCP ports by using option netstat -lt.



```
# netstat -lt

Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address               Foreign Address             State
tcp        0      0 *:dctp                      *:*                         LISTEN
tcp        0      0 *:mysql                     *:*                         LISTEN
tcp        0      0 *:sunrpc                    *:*                         LISTEN
tcp        0      0 *:munin                     *:*                         LISTEN
tcp        0      0 *:ftp                       *:*                         LISTEN
tcp        0      0 localhost.localdomain:ipp   *:*                         LISTEN
tcp        0      0 localhost.localdomain:smtp  *:*                         LISTEN
tcp        0      0 *:http                      *:*                         LISTEN
tcp        0      0 *:ssh                       *:*                         LISTEN
tcp        0      0 *:https                     *:*                         LISTEN
```

Listing all UDP Listening Ports

Listing all active listening UDP ports by using option netstat -lu.



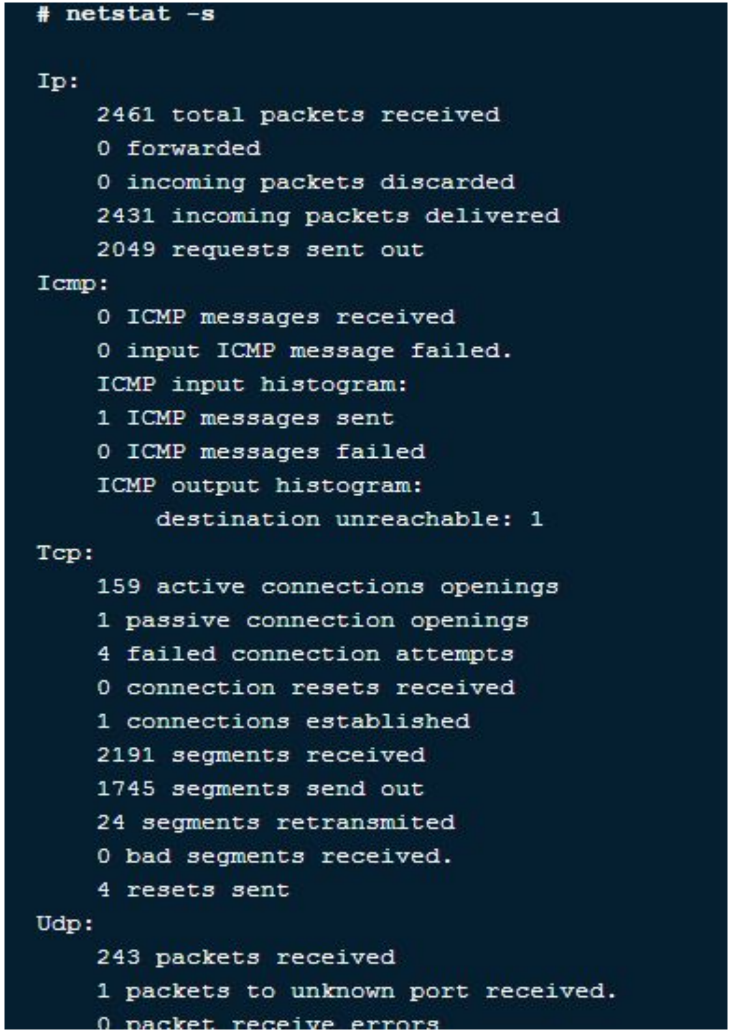
```
# netstat -lu

Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address               Foreign Address             State
udp        0      0 *:39578                     *:*
udp        0      0 *:meregister                *:*
udp        0      0 *:vpps-qua                  *:*
udp        0      0 *:openvpn                   *:*
udp        0      0 *:mdns                      *:*
udp        0      0 *:sunrpc                    *:*
udp        0      0 *:ipp                       *:*
udp        0      0 *:60222                     *:*
udp        0      0 *:mdns                      *:*
```

Showing Statistics by Protocol

Displays statistics by protocol. By default, statistics are shown for the TCP, UDP, ICMP, and IP protocols. **The -s parameter can be used to specify a set of protocols.**

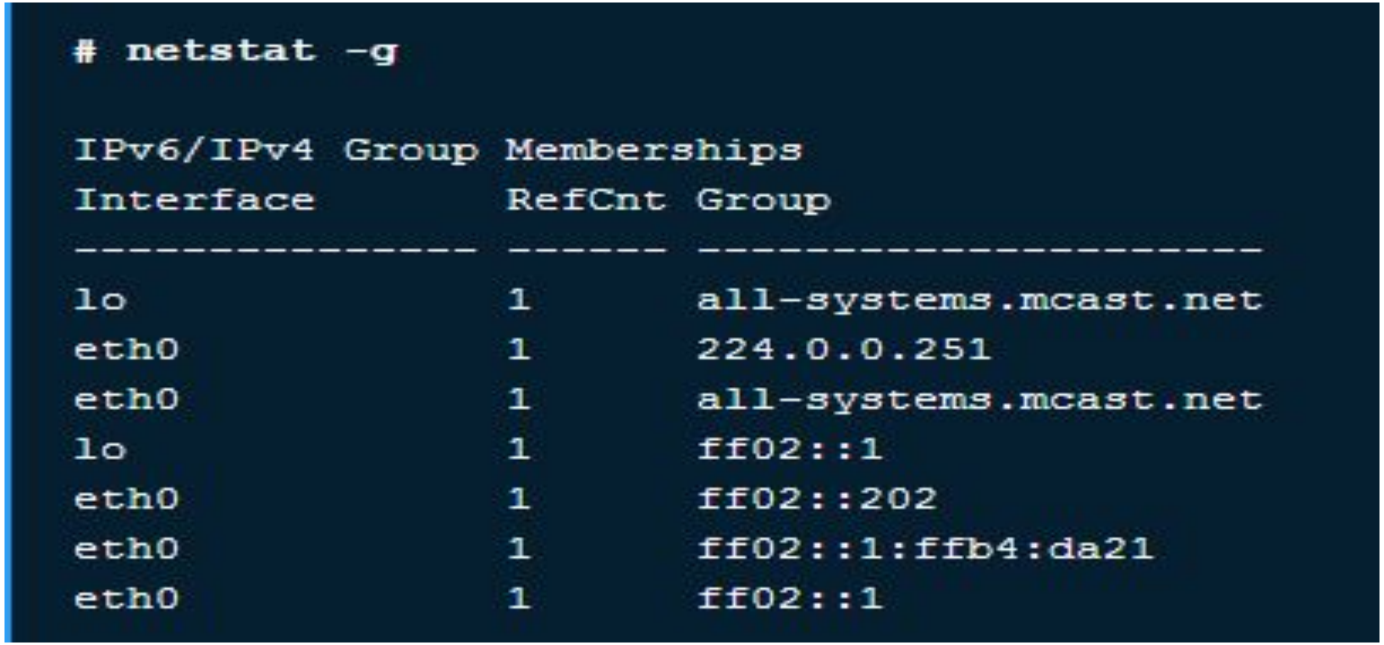
Showing statistics of only **TCP protocol by using option netstat -st.**

Showing statistics of only **UDP Protocol netstat -su**



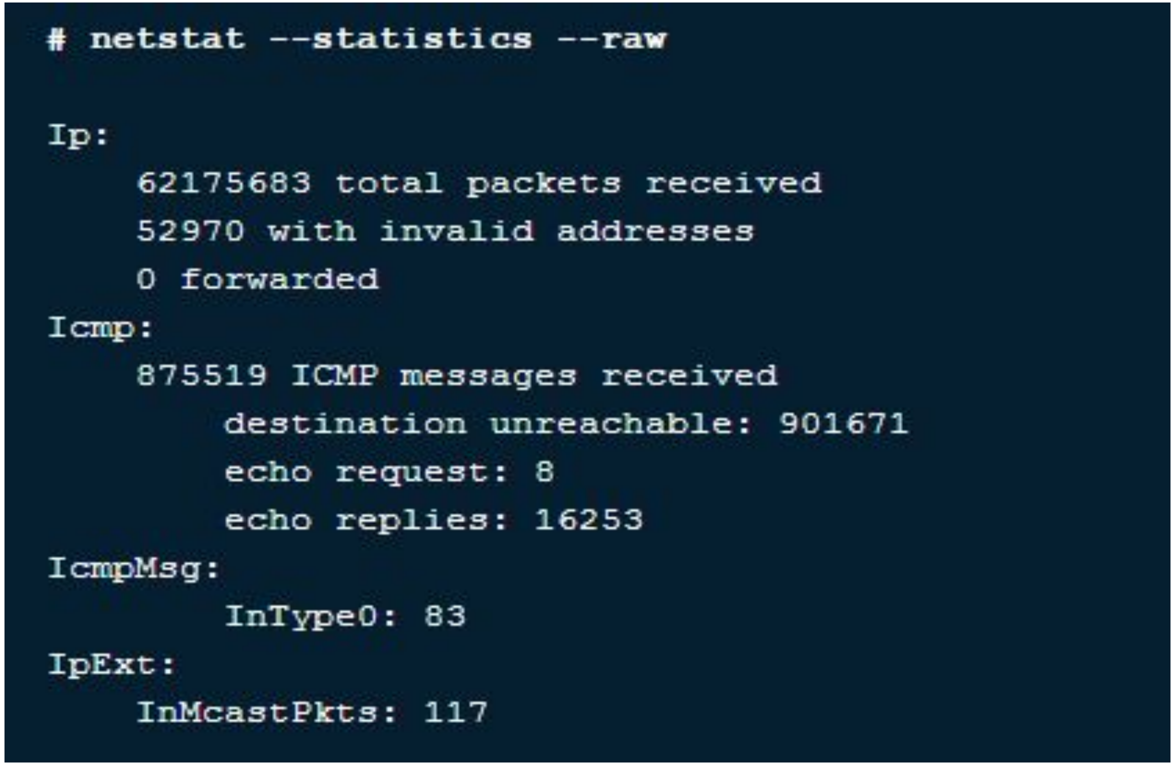
```
# netstat -s

Ip:
    2461 total packets received
    0 forwarded
    0 incoming packets discarded
    2431 incoming packets delivered
    2049 requests sent out
Icmp:
    0 ICMP messages received
    0 input ICMP message failed.
    ICMP input histogram:
    1 ICMP messages sent
    0 ICMP messages failed
    ICMP output histogram:
        destination unreachable: 1
Tcp:
    159 active connections openings
    1 passive connection openings
    4 failed connection attempts
    0 connection resets received
    1 connections established
    2191 segments received
    1745 segments send out
    24 segments retransmited
    0 bad segments received.
    4 resets sent
Udp:
    243 packets received
    1 packets to unknown port received.
    0 packet receive errors
```

Displaying IPv4 and IPv6 Information



```
# netstat -g

IPv6/IPv4 Group Memberships
Interface       RefCnt Group
--------------- ------ ---------------------
lo              1      all-systems.mcast.net
eth0            1      224.0.0.251
eth0            1      all-systems.mcast.net
lo              1      ff02::1
eth0            1      ff02::202
eth0            1      ff02::1:ffb4:da21
eth0            1      ff02::1
```

Displaying RAW Network Statistics



```
# netstat --statistics --raw

Ip:
    62175683 total packets received
    52970 with invalid addresses
    0 forwarded
Icmp:
    875519 ICMP messages received
        destination unreachable: 901671
        echo request: 8
        echo replies: 16253
IcmpMsg:
        InType0: 83
IpExt:
    InMcastPkts: 117
```

Ifconfig (Interface configuration)

ifconfig

The ifconfig command shows the details of the network interface(s) defined in the system.

configure, manage and query network interface parameters via command line

The most common option is -a , which shows all the interfaces.

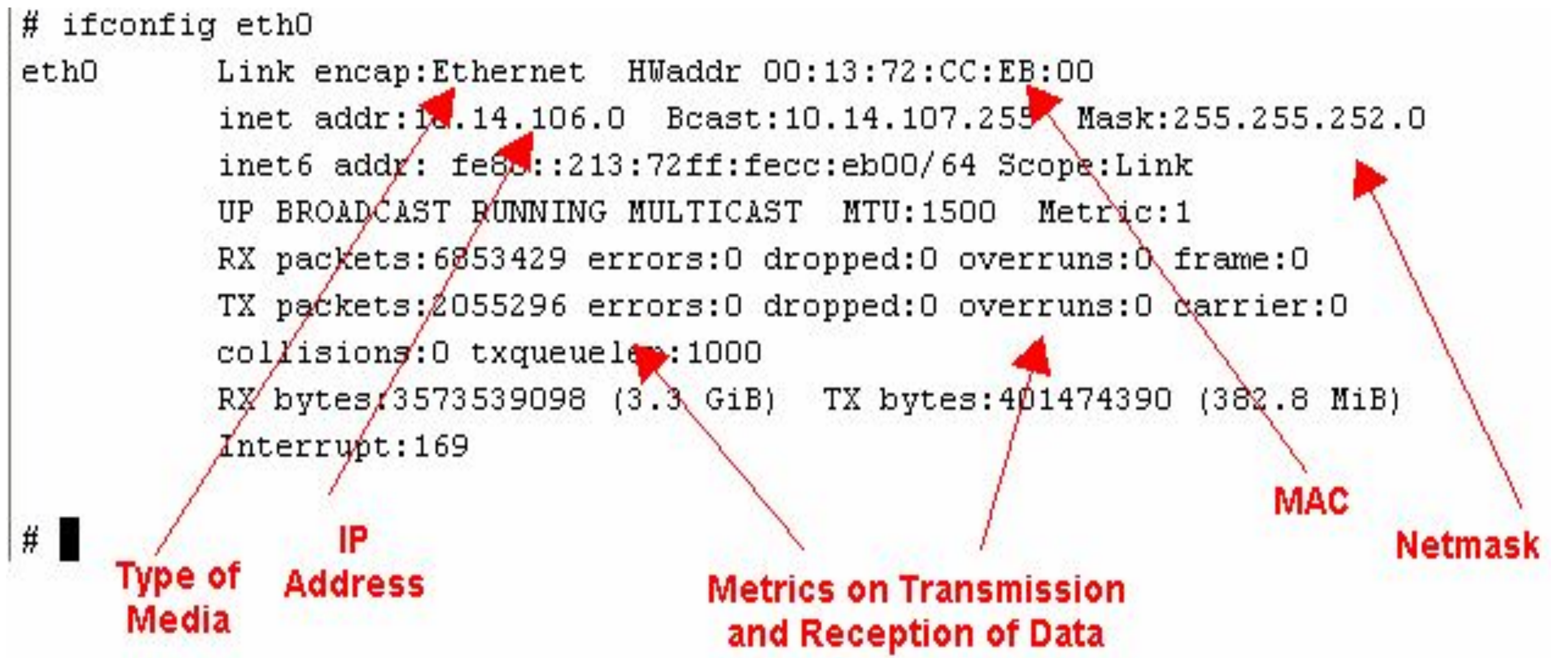
ifconfig -a

```
-a          display all interfaces which are currently available, even if down
```

The usual name of the primary Ethernet network interface is eth0. To find out the details of a specific interface, e.g. eth0, you can use:

ifconfig eth0

The output is show below



```
# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:13:72:CC:EB:00
          inet addr:10.14.106.0  Bcast:10.14.107.255  Mask:255.255.252.0
          inet6 addr: fe80::213:72ff:fecc:eb00/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:6853429 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2055296 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3573539098 (3.3 GiB)  TX bytes:401474390 (382.8 MiB)
          Interrupt:169

#
```

Here are some key parts of the output:

- **Link encap**: the type of the hardware physical medium supported by this interface (Ethernet, in this case)
- **HWaddr**: the unique identifier of the NIC card. Every NIC card has a unique identifier assigned by the manufacturer, called MAC or MAC address. The IP address is attached to the MAC to the server. If this IP address is changed, or this card is moved from this server to a different one, the MAC does not change.
- **Mask**: the netmask
- **inet addr**: the IP address attached to the interface
- **RX packets**: the number of packets received by this interface
- **TX packets**: the number of packets sent

The command is not just used to check the settings; it’s used to configure and manage the interface as well. Here is a short list of parameters and options for this command:

up/down – enables or disables a specific interface. You can use the down parameter to shutdown an interface (or disable it):

```
# ifconfig eth0 down
```

Similarly to bring it up (or enable) it, you would use:

```
# ifconfig eth0 up
```

add – sets a specific IP address for the interface. To set an IP address of 192.168.1.101 to the interface eth0, you would issue:

```
# ifconfig eth0 add  192.168.1.101
```

netmask – sets the netmask parameter of the interface. Here is an example where you can set the netmask of the eth0 interface to 255.255.255.0

```
# ifconfig eth0 netmask  255.255.255.0
```

How to Assign a Broadcast to Network Interface

Using the “broadcast” argument with an interface name will set the broadcast address for the given interface. For example, “ifconfig eth0 broadcast 172.16.25.63” command sets the broadcast address to an interface eth0.

```
Ifconfig eth0 boardcast 172.16.25.63
```

How to Assign a IP, Netmask and Broadcast to Network Interface

```
Ifconfig eth0 172.16.25.125 netmask 255.255.255.244 broadcast 172.16.25.63
```

How to Change MTU for an Network Interface

The “mtu” argument set the maximum transmission unit to an interface. The MTU allows you to set the limit size of packets that are transmitted on an interface.

```
Ifconfig eth0 mtu 1000
```

How to Enable Promiscuous Mode

```
Ifconfig eth0 promisc
```

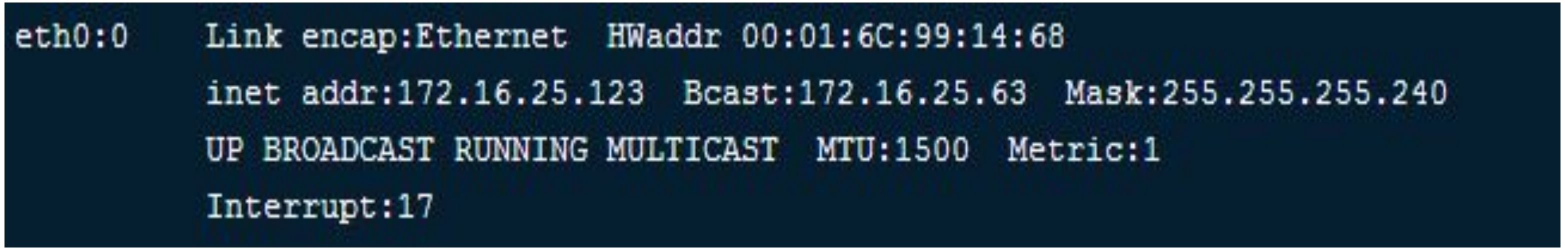
How to Disable Promiscuous Mode

```
Ifconfig eth0 -promisc
```

How to Add New Alias to Network Interface

Ifconfig eth0:0 172.16.25.127

Next, verify the newly created alias network interface address, by using “**ifconfig eth0:0**” command.



```
eth0:0    Link encap:Ethernet  HWaddr 00:01:6C:99:14:68
          inet addr:172.16.25.123  Bcast:172.16.25.63  Mask:255.255.255.240
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          Interrupt:17
```

How to Remove Alias to Network Interface

```
Ifconfig eth0:0 down
```

How to Change the MAC address of Network Interface

To change the MAC (Media Access Control) address of an eth0 network interface, use the following command with argument “hw ether“

```
Ifconfig eth0 hw ether AA:BB:CC:DD:EE:FF
```

IP Command

IP Command

The ip command is used to assign an address to a network interface and/or configure network interface parameters on Linux operating systems. This command replaces old good and now deprecated ifconfig command on modern Linux distributions.

Use this command to display and **configure the network parameters** for host interfaces.

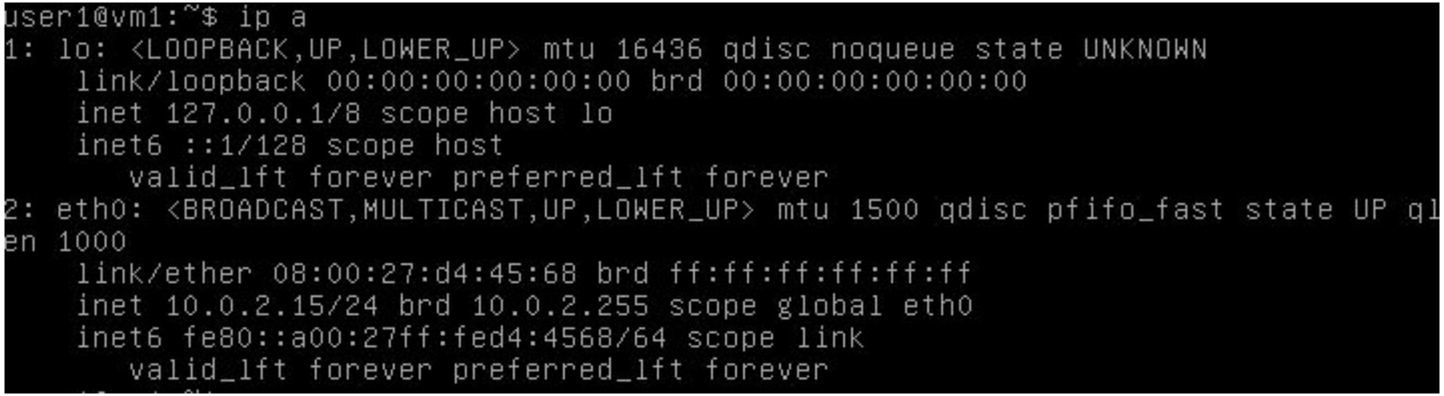
Displays info about all network interfaces

Type the following command to list and show all ip address associated on on all network interfaces:

```
ip a
```

OR

```
ip addr
```



```
user1@vm1:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP ql
en 1000
    link/ether 08:00:27:d4:45:68 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global eth0
    inet6 fe80::a00:27ff:fed4:4568/64 scope link
       valid_lft forever preferred_lft forever
```

You can select between IPv4 and IPv6 using the following syntax:

```
### Only show TCP/IP IPv4  ##
ip -4 a

### Only show TCP/IP IPv6  ###
ip -6 a
```

It is also possible to specify and list particular interface TCP/IP details:

```
### Only show eth0 interface ###
ip a show eth0
ip a list eth0
ip a show dev eth0

### Only show running interfaces ###
ip link ls up
```

Assigns the IP address to the interface

The syntax is as follows to add an IPv4/IPv6 address:

```
ip a add {ip_addr/mask} dev {interface}
```

To assign 192.168.1.200/255.255.255.0 to eth0, enter:

```
ip a add 192.168.1.200/255.255.255.0 dev eth0
```

OR

```
ip a add 192.168.1.200/24 dev eth0
```

Adding the broadcast address on the interface

By default, the ip command does not set any broadcast address unless explicitly requested. So syntax is as follows to set broadcast ADDRESS:

```
ip addr add brd {ADDDRESS-HERE} dev {interface}
```

```
ip addr add broadcast {ADDDRESS-HERE} dev {interface}
```

```
ip addr add broadcast 172.20.10.255 dev dummy0
```

Remove / Delete the IP address from the interface

The syntax is as follows to remove an IPv4/IPv6 address:

```
ip a del {ipv6_addr_OR_ipv4_addr} dev {interface}
```

To delete 192.168.1.200/24 from eth0, enter:

```
ip a del 192.168.1.200/24 dev eth0
```

Flush the IP address from the interface

You can delete or remove an IPv4/IPv6 address one-by-one as described above. However, the flush command can remove as flush the IP address as per given condition. For example, you can delete all the IP addresses from the private network 192.168.2.0/24 using the following command:

```
ip -s -s a f to 192.168.2.0/24
```

Sample outputs:

```
2: eth0    inet 192.168.2.201/24 scope global secondary eth0
2: eth0    inet 192.168.2.200/24 scope global eth0
*** Round 1, deleting 2 addresses ***
*** Flush is complete after 1 round ***
```

You can disable IP address on all the ppp (Point-to-Point) interfaces:

```
ip -4 addr flush label "ppp*"
```

change the state of the device to UP or DOWN?

The syntax is as follows:

```
ip link set dev {DEVICE} {up|down}
```

To make the state of the device eth1 down, enter:

```
ip link set dev eth1 down
```

To make the state of the device eth1 up, enter:

```
ip link set dev eth1 up
```

change the MTU of the device

```
ip link set mtu {NUMBER} dev {DEVICE}
```

To change the MTU of the device eth0 to 9000, enter:

```
ip link set mtu 9000 dev eth0
```

ip route: Routing table management commands

Use the following command to manage or manipulate the kernel routing table.

Show routing table

To display the contents of the routing tables:

```
ip r
```

```
ip r list
```

```
ip r list [options]
```

```
ip route
```

Sample output

```
192.168.1.0/24 dev eth1  proto kernel  scope link  src 192.168.1.10
```

Display routing for 192.168.1.0/24:

```
ip r list 192.168.1.0/24
```

Sample outputs:

```
192.168.1.0/24 dev eth1  proto kernel  scope link  src 192.168.1.10
```

Add a new route

The syntax is:

```
ip route add {NETWORK/MASK} via {GATEWAYIP}
```

```
ip route add {NETWORK/MASK} dev {DEVICE}
```

```
ip route add default {NETWORK/MASK} dev {DEVICE}
```

```
ip route add default {NETWORK/MASK} via {GATEWAYIP}
```

Add a plain route to network 192.168.1.0/24 via gateway 192.168.1.254:

```
ip route add 192.168.1.0/24 via 192.168.1.254
```

Delete a route

The syntax is as follows to delete default gateway:

```
ip route del default
```

```
ip route del 192.168.1.0/24 dev eth0
```

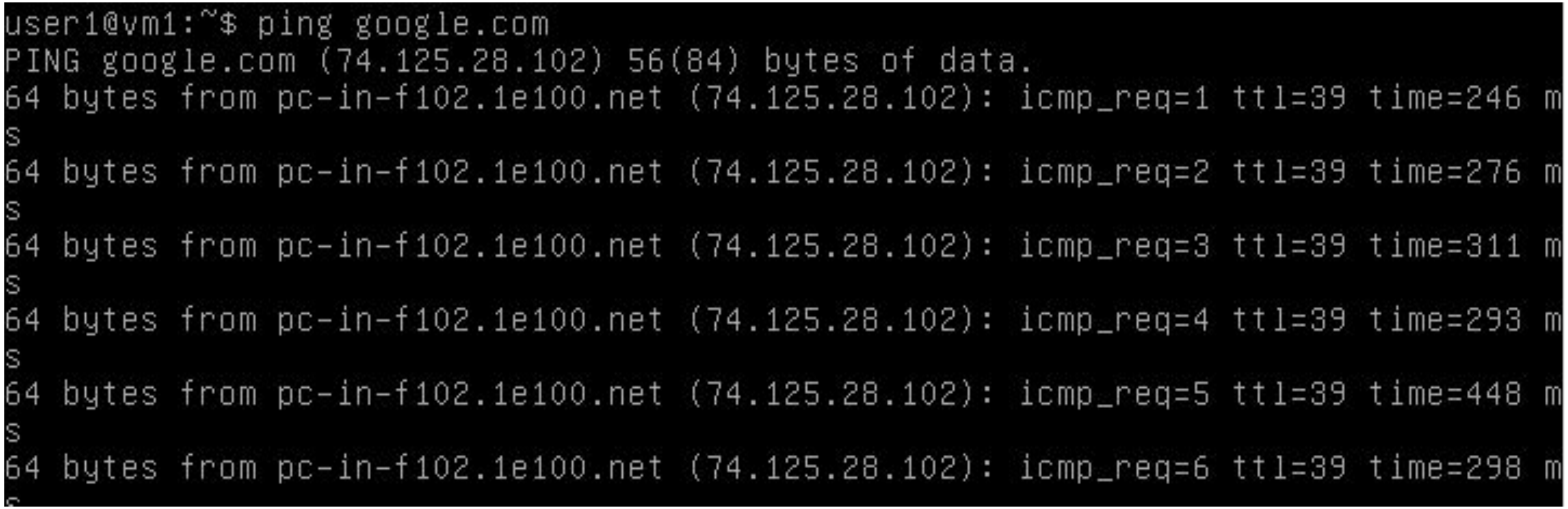
Ping

Ping

Test the connection between the local server/computer and a remote UNIX server. The ping command sends ICMP Echo Request (ECHO_REQUEST) packets to the host once per second. Each packet that is echoed back via an ICMP Echo Response packet is written to the standard output, including round-trip time. Use this command to find out:

1. If a remote server is up and running.
2. Test network problems.
3. Test network connectivity to your local or remote gateways.
4. Verify network problems.

ping google.com



```
user1@vm1:~$ ping google.com
PING google.com (74.125.28.102) 56(84) bytes of data.
64 bytes from pc-in-f102.1e100.net (74.125.28.102): icmp_req=1 ttl=39 time=246 m
s
64 bytes from pc-in-f102.1e100.net (74.125.28.102): icmp_req=2 ttl=39 time=276 m
s
64 bytes from pc-in-f102.1e100.net (74.125.28.102): icmp_req=3 ttl=39 time=311 m
s
64 bytes from pc-in-f102.1e100.net (74.125.28.102): icmp_req=4 ttl=39 time=293 m
s
64 bytes from pc-in-f102.1e100.net (74.125.28.102): icmp_req=5 ttl=39 time=448 m
s
64 bytes from pc-in-f102.1e100.net (74.125.28.102): icmp_req=6 ttl=39 time=298 m
```

time=69ms” This is the round trip time: the time between sending “Are you there?” and receiving “Yes I am!”.

TTL : (Time to live)
A timer value included in packets sent over TCP/IP-based networks that tells the recipients how long to hold or use the packet or any of its included data before expiring and discarding the packet or data.

mdev
It's the standard deviation, essentially an average of how far each ping RTT is from the mean RTT. The higher `mdev` is, the more variable the RTT is (over time).

Icmp_seq :sequence number
Icmp_req:request number

Send limited number of pings

You can stop after sending 4 ECHO_REQUEST packets with the -c option as follows:

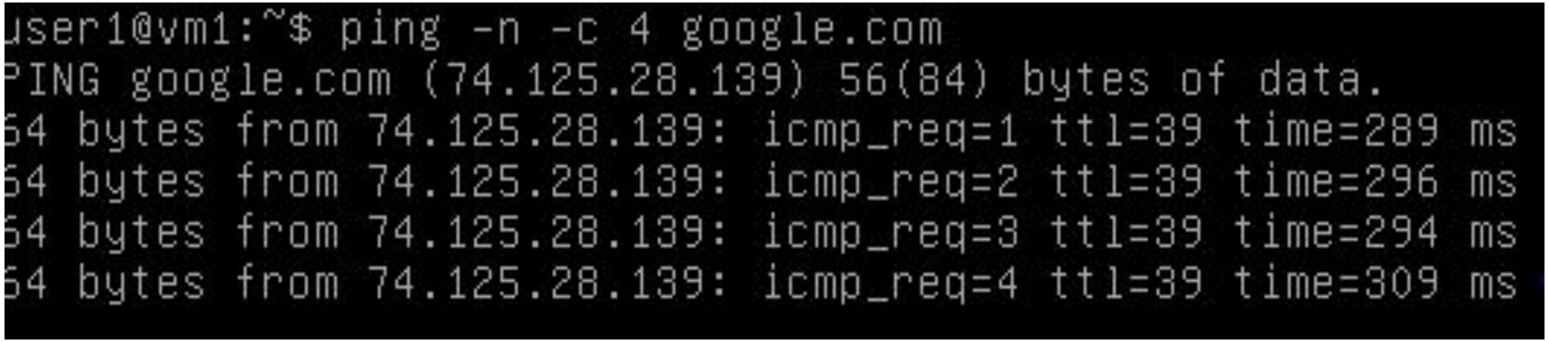
```
  ping  -c -4  yahoo.com
```

-c count -stop after sending count echo request packet,ping waits for count echo reply packets.

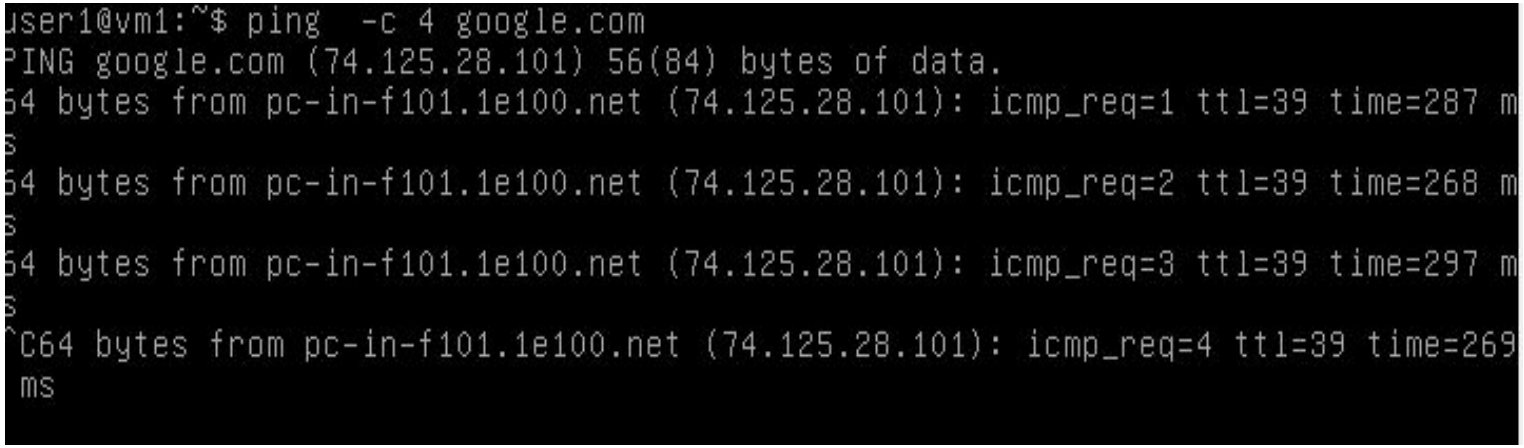
Avoid dns lookup

Pass the -n option to avoid to lookup symbolic names for host addresses i.e. numeric output only:

```
ping -n -c 4 google.com
```



```
user1@vm1:~$ ping -n -c 4 google.com
PING google.com (74.125.28.139) 56(84) bytes of data.
64 bytes from 74.125.28.139: icmp_req=1 ttl=39 time=289 ms
64 bytes from 74.125.28.139: icmp_req=2 ttl=39 time=296 ms
64 bytes from 74.125.28.139: icmp_req=3 ttl=39 time=294 ms
64 bytes from 74.125.28.139: icmp_req=4 ttl=39 time=309 ms
```



```
user1@vm1:~$ ping  -c 4 google.com
PING google.com (74.125.28.101) 56(84) bytes of data.
64 bytes from pc-in-f101.1e100.net (74.125.28.101): icmp_req=1 ttl=39 time=287 m
s
64 bytes from pc-in-f101.1e100.net (74.125.28.101): icmp_req=2 ttl=39 time=268 m
s
64 bytes from pc-in-f101.1e100.net (74.125.28.101): icmp_req=3 ttl=39 time=297 m
s
^C64 bytes from pc-in-f101.1e100.net (74.125.28.101): icmp_req=4 ttl=39 time=269
 ms
```

Increase or Decrease the Time Interval Between Packets

default ping waits for 1 second before sending the next packet.

Increase Ping Time Interval

Example: Wait for 5 seconds before sending the next packet.

```
$ ping -i 5 IP
```

Decrease Ping Time Interval

Example: Wait 0.1 seconds before sending the next packet.

```
ping -i 0.1 IP
```

Note: Only super user can specify interval less than 0.2 seconds.

Check whether the local network interface is up and running

```
ping 0
```

```
ping localhost
```

```
ping 127.0.0.1
```

Flood the network

Super users can send hundred or more packets per second using -f option.

```
ping -f localhost
```

```
PING localhost (127.0.0.1) 56(84) bytes of data.

.^C

--- localhost ping statistics ---

427412 packets transmitted, 427412 received, 0% packet loss, time 10941ms
```

Audible ping

```
ping -a IP
```

Give beep when the peer is reachable

Find out the IP address

```
ping -c 1 google.com
```

Specify path for ping to send the packet

```
ping hop1 hop2 hop3 .. hopN destination
```

```
ping 192.168.3.33 192.168.7.1 192.168.4.45
```

Record and print route of how ECHO_REQUEST sent and ECHO_REPLY received

It records, and prints the network route through which the packet is sent and received.

```
ping -R 192.168.1.63
```

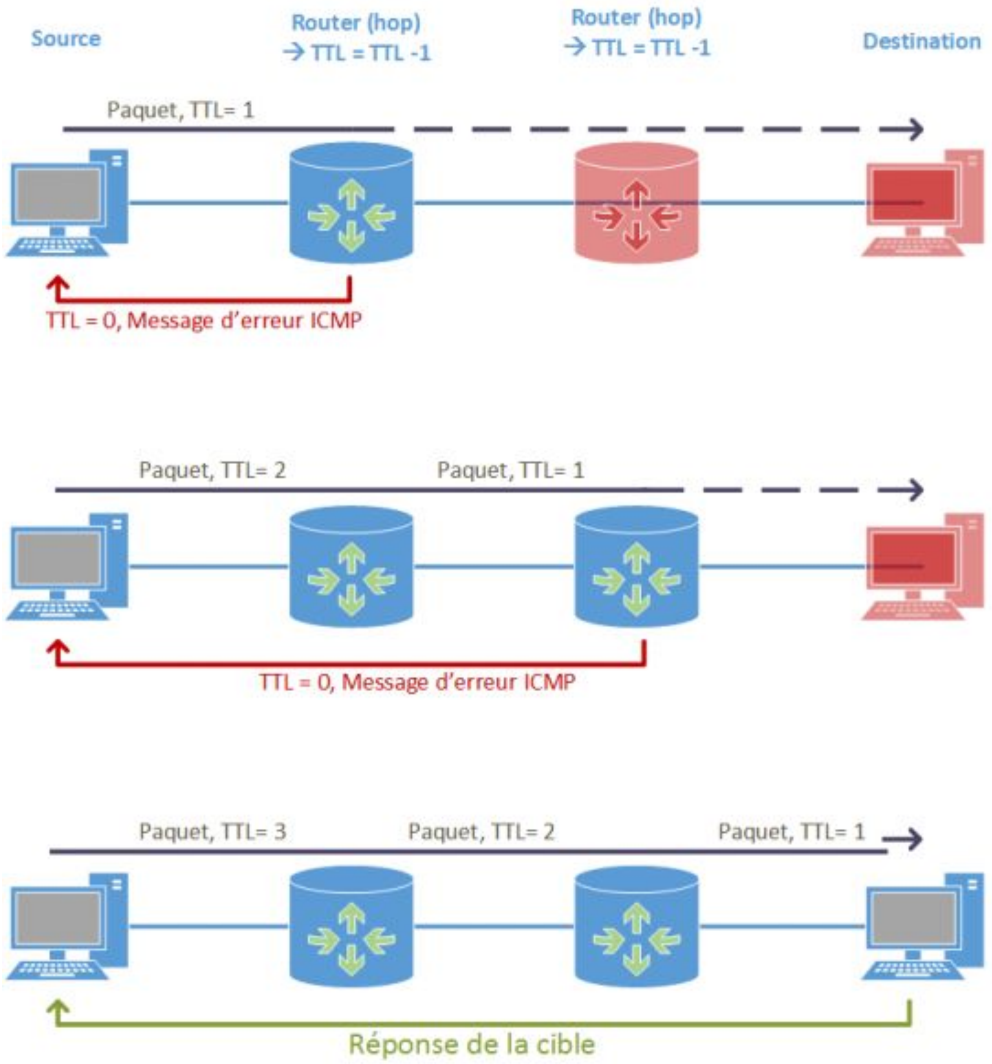
```
PING 192.168.1.63 (192.168.1.63) 56(84) bytes of data.

64 bytes from 192.168.1.63: icmp_seq=1 ttl=61 time=2.05 ms

RR:   192.168.9.118

        192.168.3.25

        192.168.10.35

        192.168.1.26

        192.168.1.63

        192.168.1.63

        192.168.10.4

        192.168.3.10

        192.168.4.25
```

Traceroute

traceroute utility prints out complete route to a particular destination.

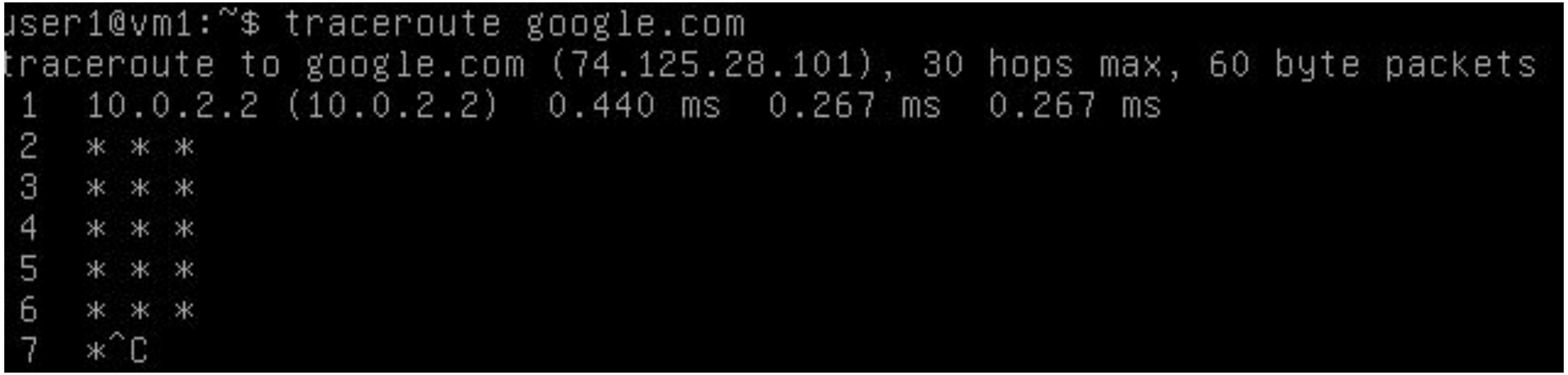
Traceroute utility uses the TTL field in the IP header to achieve its operation.
TTL field describes how much hops a particular packet will take while traveling on network.
this effectively outlines the lifetime of the packet on network. This field is usually set to 32 or 64.

- Each time the packet is held on an intermediate router.
- decreases the TTL value by 1
- When a router finds the TTL value of 1 in a received packet then that packet is not forwarded but instead discarded.
- After discarding the packet, router sends an ICMP error message of “Time exceeded” back to the source from where packet generated.
- The ICMP packet that is sent back contains the IP address of the router.
- So now it can be easily understood that traceroute operates by sending packets with TTL value starting from 1 and then incrementing by one each time
- So traceroute incrementally fetches the IP of all the routers between the source and the destination.


Source
Router (hop) → TTL = TTL -1
Router (hop) → TTL = TTL -1
Destination
Paquet, TTL= 1
TTL = 0, Message d'erreur ICMP
Paquet, TTL= 2
Paquet, TTL= 1
TTL = 0, Message d'erreur ICMP
Paquet, TTL= 3
Paquet, TTL= 2
Paquet, TTL= 1
Réponse de la cible

How to run traceroute?

traceroute google.com



```
user1@vm1:~$ traceroute google.com
traceroute to google.com (74.125.28.101), 30 hops max, 60 byte packets
 1  10.0.2.2 (10.0.2.2)  0.440 ms  0.267 ms  0.267 ms
 2  * * *
 3  * * *
 4  * * *
 5  * * *
 6  * * *
 7  *^C
```

How To Read Traceroute's Output

The first line tells us the conditions that traceroute is operating under:

traceroute to google.com (173.194.38.137), 30 hops max, 60 byte packets

- It gives the specified host
- IP address that DNS returns for that domain
- maximum number of hops to check
- size of the packet that will be used.

- Each lines gives the details of interaction with each router encountered.
- traceroute not only gives the IP addresses of the intermediate routers but also three round trip times for that particular router as for each router the traceroute commands fires three packets.
- The ‘*’ field in output
- This depicts that the required field could not be fetched.

- The maximum number of hops can be adjusted with the -m flag.
- host you are trying to route to is over 30 hops away.
- The maximum value you can set is 255.

traceroute -m 255 google.com

Adjust the size of the packet that is sent to each hop

traceroute google.com 70

```
traceroute to google.com (173.194.38.128), 30 hops max, 70 byte packets
 1  192.241.160.254 (192.241.160.254)  0.364 ms  0.330 ms  0.319 ms
 2  192.241.164.237 (192.241.164.237)  0.284 ms  0.343 ms  0.321 ms
```

After the first line, each subsequent line represents a "**hop**", or **intermediate host** that your traffic must pass through to reach the computer represented by the host you specified.

Each line has the following format:

hop_number   host_name   (IP_address) packet_round_trip_times

hop_number: A sequential count of the number of degrees of separation the host is from your computer.

IP_address: This field contains the IP address for this network hop.

Round**trip_times**: The remainder of the line gives the round-trip times for a packet to the host and back again

3 nyk-b6-link.telia.net (62.115.35.101) 0.311 ms 0.302 ms 0.293 ms

By default, three packets are sent to each host

Configure the TTL value to start with

Traceroute utility is **flexible enough** to **accept the TTL value** that the user wants to start the utility with. By **default** its **value is 1** which means it starts off with the first router in the path but using the **‘-f’ option** (which expects the new value of TTL) a new value of the TTL field can be set.

```
traceroute google.com

traceroute to google.com (74.125.236.132), 30 hops max, 60 byte packets

1  220.224.141.129 (220.224.141.129)  89.181 ms  101.540 ms  101.503 ms

2  115.255.239.65 (115.255.239.65)  101.468 ms  101.431 ms  101.324 ms

3  124.124.251.245 (124.124.251.245)  121.373 ms  121.350 ms  158.694 ms

4  115.255.239.45 (115.255.239.45)  101.223 ms  141.135 ms  123.932 ms

5  72.14.212.118 (72.14.212.118)  123.867 ms  123.832 ms  123.802 ms

6  72.14.232.202 (72.14.232.202)  123.773 ms  123.742 ms  587.812 ms

7  216.239.48.179 (216.239.48.179)  587.723 ms  587.681 ms  587.642 ms

8  bom03s02-in-f4.1e100.net (74.125.236.132)  577.548 ms  577.524 ms  587.512 ms


$ traceroute google.com -f 8

traceroute to google.com (74.125.236.129), 30 hops max, 60 byte packets

8  bom03s02-in-f1.1e100.net (74.125.236.129)  96.961 ms  96.886 ms  96.849 ms
```

Configure Number of Queries per Hop

- the traceroute utility sends 3 packets per hop to provide 3 round trip times
- This default value of 3 is configurable using the option ‘-q’.

```
$ traceroute google.com -q 5

traceroute to google.com (173.194.36.46), 30 hops max, 60 byte packets


1  220.224.141.129 (220.224.141.129)  91.579 ms  91.497 ms  91.458 ms  91.422 ms  91.385 ms

2  115.255.239.65 (115.255.239.65)  91.356 ms  91.325 ms  98.868 ms  98.848 ms  98.829 ms
```

Configure Response Wait Time

- The time for which traceroute utility waits after issuing a probe can also be configured.
- The -w option expects a value which the utility will take as the response time to wait for.
- the wait time is 0.1 seconds and the traceroute utility was unable to wait for any response and it printed all the *’s.

```
traceroute google.com -w 0.1

traceroute to google.com (74.125.236.101), 30 hops max, 60 byte packets

1  * * *

2  * * *

3  * * *

..

26  * * *

27  * * *

28  * * *

29  * * *

30  * * *
```

Disable IP address and host name mapping

Traceroute provides an option through which the **mapping of IP addresses with host name** (that traceroute tries) **is disabled**. The option for doing this is **‘-n’** . The following example illustrates this :

```
traceroute google.com -n

traceroute to google.com (173.194.36.7), 30 hops max, 60 byte packets

1  220.224.141.129  109.352 ms  109.280 ms  109.248 ms

2  115.255.239.65  131.633 ms  131.598 ms  131.573 ms

3  124.124.251.245  131.554 ms  131.529 ms  131.502 ms

4  115.255.239.45  131.478 ms  131.464 ms  199.741 ms
```

mtr(My Traceroute)

- It is a **powerful network diagnostic tool**
- combines the power of both **Ping** and **Traceroute** commands.
- diagnose and isolate network errors
- provide helpful **network status reports**.
- MTR works by sending ICMP packets by incrementally increasing the TTL value to find the route between the source and the given destination.
- MTR collects additional information regarding the **state, connection, and responsiveness of the intermediate hosts.**

Installation

On debian or Ubuntu systems use the following command:

```
$ sudo apt-get install mtr
```

On centos and fedora systems execute the following command:

```
$ yum install mtr
```

Execute mtr for a Domain

MTR works in two modes, a graphical mode (X11) and text based mode (ncurses). By default, mtr command runs in X11 mode.

```
mtr google.com
```

The above command will open up a GUI window, and display the results.

Launch Text Mode using –curses

Use the –curses option to run mtr in terminal mode.

```
mtr --curses google.com
```

Or

```
mtr -t google.com
```

```
                   Packets            Pings
 Host              Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. mblaze.hilink     0.0%    20    3.2   2.7   2.1   3.8   0.5
 2. 10.228.129.9      0.0%    20  187.7  61.8  41.6 187.7  31.3
 3. 10.228.149.14     0.0%    20  112.1  60.2  33.2 112.1  17.8
 4. 116.202.226.145   0.0%    20   57.9  63.2  35.2 147.6  24.4
 5. 116.202.226.21    0.0%    20   35.4  70.4  35.4 211.8  48.6
 6. 72.14.219.94      0.0%    20   58.9  74.6  43.4 231.2  44.2
 7. 72.14.233.204     0.0%    20   46.9  69.8  40.3 222.5  41.9
 8. 72.14.239.20      0.0%    20   94.1 259.2  68.8 3436. 748.2
 9. 209.85.244.111    0.0%    20   86.4  97.5  72.1 232.2  34.3
10. google.com        0.0%    19  387.9 132.5  71.8 387.9  84.9
```

The above will run continuously in interactive mode.

Generate MTR reports using the following syntax:

```
mtr -rw google.com
HOST: HIFX-33                              Loss%   Snt   Last   Avg  Best  Wrst StDev
  1.|-- 172.17.2.1                         0.0%  10     1.0   1.0   0.9   1.1   0.0
  2.|-- static-29.134.16.61-tataidc.co.in              0.0%  10     5.6   7.2   4.8  23.3   5.6
  3.|-- ???                                100.0 10     0.0   0.0   0.0   0.0   0.0
  4.|-- ???                                100.0 10     0.0   0.0   0.0   0.0   0.0
  5.|-- 10.117.227.50                      0.0%  10   29.7  30.4  29.2  31.8   0.5
  6.|-- static-98.110.93.111-tataidc.co.in             0.0%  10  124.2  84.4  34.9 124.2  33.5
  7.|-- 14.141.20.165.static-vsnl.net.in               0.0%  10  105.8  83.7  33.2 124.9  29.6
  8.|-- ???                                100.0 10     0.0   0.0   0.0   0.0   0.0
  9.|-- ???                                100.0 10     0.0   0.0   0.0   0.0   0.0
 10.|-- 115.114.142.137.static-Chennai.vsnl.net.in 10.0%      10  121.0 100.3  78.7 125.7  17.1
 11.|-- 121.240.1.50                       20.0%        10  116.5  93.1  67.9 116.5  18.4
 12.|-- 72.14.233.204                      10.0%        10   84.9  90.0  54.1 123.5  25.8
 13.|-- 209.85.255.43                             0.0%  9   57.2  66.9  28.7 119.3  31.8
 14.|-- maa03s18-in-f46.1e100.net                      0.0%  9   50.3  71.0  29.9 104.7  28.8
```

The r option flag generates the report (short for --report)

The w option flag uses the long-version of the hostname (short for --report-wide).

mtr --report google.com (with out using w flag)

```
Start: Thu Apr  7 10:35:05 2016
HOST: HIFX-33                     Loss%   Snt   Last   Avg  Best  Wrst StDev
  1.|-- 172.17.2.1                 0.0% 10     1.1   1.0   0.9   1.1   0.0
  2.|-- static-29.134.16.61-tatai  0.0%       10     6.2   9.9   5.2  44.7  12.2
  3.|-- ???                  100.0 10     0.0   0.0   0.0   0.0   0.0
  4.|-- ???                  100.0 10     0.0   0.0   0.0   0.0   0.0
  5.|-- 10.117.227.50              0.0% 10   31.9  30.8  30.5  31.9   0.3
  6.|-- static-98.110.93.111-tata  0.0%       10  114.0  94.4  57.4 116.6  17.4
  7.|-- 14.141.20.165.static-vsnl  0.0%       10  106.3  99.8  46.5 149.2  26.8
  8.|-- ???                  100.0 10     0.0   0.0   0.0   0.0   0.0
  9.|-- ???                  100.0 10     0.0   0.0   0.0   0.0   0.0
 10.|-- 115.114.142.137.static-Ch 20.0%  10  112.6 113.2 105.8 142.4  12.2
 11.|-- 121.240.1.50              20.0%       10  122.5 116.0 103.9 134.4   9.2
 12.|-- 72.14.233.204             20.0%       10  113.8 111.1  99.1 119.1   7.7
 13.|-- 209.85.255.43             10.0%       10  113.4 115.8 107.2 122.7   4.8
 14.|-- maa03s18-in-f14.1e100.net 20.0%        10  111.7 111.5 104.5 118.3   4.0
```

- report option which **sends 10 packets t**o the host google.com and generates the output.
- Without the --report option, mtr will run continuously in an interactive environment.

The names for the hosts are determined by reverse DNS lookups

mtr --no-dns --report google.com

```
HOST: deleuze                    Loss%   Snt   Last   Avg  Best  Wrst StDev
  1. 192.168.1.1                  0.0%    10    2.2   2.2   2.0   2.7   0.2
  2. 68.85.118.13                 0.0%    10    8.6  11.0   8.4  17.8   3.0
  3. 68.86.210.126                0.0%    10    9.1  12.1   8.5  24.3   5.2
  4. 68.86.208.22                 0.0%    10   12.2  15.1  11.7  23.4   4.4
  5. 68.85.192.86                 0.0%    10   17.2  14.8  13.2  17.2   1.3
  6. 68.86.90.25                  0.0%    10   14.2  16.4  14.2  20.3   1.9
```

MTR provides valuable statistics regarding the durability of that connection in the seven columns that follow.

The **Loss%** column shows the percentage of packet loss at each hop.

The **Snt** column counts the number of packets sent.

The next four columns **Last, Avg, Best, and Wrst** are all measurements of **latency in milliseconds**

- **Last** is the latency of the last packet sent.
- **Avg** is average latency of all packets.
- **Best** and **Wrst** display the best (shortest) and worst (longest) round trip time for a packet to this host.
- **StDev**, provides the standard deviation of the latencies to each host.
- The higher the standard deviation, the greater the difference is between measurements of latency

Verify Packet Loss

Rate Limit--common practice among the service provider to “Rate Limit” the ICMP traffic.
This can provide an illusion of packet loss
fact there is no loss.
to verify whether the loss is real or due to rate limiting, check the “Loss%” of the next hop.
If it shows 0.0%, then you can be sure that the “Loss%” reported is due to the ICMP rate limiting and not actual loss.

```
root@localhost:~# mtr --report www.google.com
HOST: example              Loss%   Snt   Last   Avg  Best  Wrst StDev
1. 63.247.74.43              0.0%    10    0.3   0.6   0.3   1.2   0.3
2. 63.247.64.157             50.0%    10    0.4   1.0   0.4   6.1   1.8
3. 209.51.130.213              0.0%    10    0.8   2.7   0.8  19.0   5.7
4. aix.pr1.atl.google.com       0.0%    10    6.7   6.8   6.7   6.9   0.1
5. 72.14.233.56              0.0%    10    7.2   8.3   7.1  16.4   2.9
6. 209.85.254.247              0.0%    10   39.1  39.4  39.1  39.7   0.2
7. 64.233.174.46              0.0%    10   39.6  40.4  39.4  46.9   2.3
8. gw-in-f147.1e100.net        0.0%    10   39.6  40.5  39.5  46.7   2.2
```

loss continues for more than 1 hop, then it is possible that there is some packet loss.
Rate limiting and Packet Loss can happen concurrently
In that case take the lowest Loss% in a sequence as actual loss.

Improper Destination Host Networking

```
root@localhost:~# mtr --report www.google.com
HOST: localhost                   Loss%   Snt   Last   Avg  Best  Wrst StDev
  1. 63.247.74.43                  0.0%    10    0.3   0.6   0.3   1.2   0.3
  2. 63.247.64.157                 0.0%    10    0.4   1.0   0.4   6.1   1.8
  3. 209.51.130.213                0.0%    10    0.8   2.7   0.8  19.0   5.7
  4. aix.pr1.atl.google.com        0.0%    10    6.7   6.8   6.7   6.9   0.1
  5. 72.14.233.56                  0.0%    10    7.2   8.3   7.1  16.4   2.9
  6. 209.85.254.247                0.0%    10   39.1  39.4  39.1  39.7   0.2
  7. 64.233.174.46                 0.0%    10   39.6  40.4  39.4  46.9   2.3
  8. gw-in-f147.1e100.net        100.0    10    0.0   0.0   0.0   0.0   0.0
```

From the above example, it may look like the packets didn’t reach the destination.
But it does reach the destination.
This may be a result of improperly configured host or firewall rules to drop ICMP packets.

Timeout and Return Route Issue

Sometimes, routers will discard the ICMP and it will be shown as ??? on the output.

Alternatively there can be a problem with Return route also.

```
9.|-- 209.85.244.25             0.0%    10  260.6 147.0  78.1 260.6  75.3
 10.|-- ???                       100.0    10    0.0   0.0   0.0   0.0   0.0
 11.|-- 74.125.200.100            0.0%    10   84.8 112.4  75.6 234.4  63.9
```

In the above example, the router at hop 10 is either not responding to ICMP or there is a problem in the return route of the packet.

Telnet(TELecommunication NETwork)

- network protocol used on the Internet or local area network (LAN) connections
- bidirectional interactive communications facility.
- enables a user to connect to a remote host or device using a telnet client, usually over port 23.

telnet hostname

- **telnet hostname** would connect a user to a host named **hostname**.
- **Telnet** enables a user to manage an account or device remotely.
- For example, a user may telnet into a computer that hosts their website to manage his or her files remotely.
- telnet provides access to a command-line interface on a remote host via a virtual terminal connection
- which consists of an 8-bit byte oriented data connection over the Transmission Control Protocol (TCP).

The problem with telnet

Telnet sends everything in clear text format including username and password. You can use tcpdump or snoop to see all information.

telnet examples

telnet myhost.com

Attempts to open a connection to the remote host **myhost.com**. If a connection is established, the host will prompt for a login name and password.

telnet -l myusername myhost.com:5555

- Attempts to open a connection to the remote host **myhost.com** on port **5555**
- using the login name **myusername**.
- using the login name **myusername**.

telnet

- Opens a local **telnet>** prompt
- where you can enter any of the commands listed above.

telnet> open myhost.com

tcpdump

- **Tcpdump is a commandline network analyzer tool**
- **packet sniffer**
- which is used to capture or filter **TCP/IP** packets that received or transferred over a network on a specific interface.
- It is available under most of the **Linux/Unix** based operating systems.
- tcpdump also gives us a option to save captured packets in a file for future analysis.
- It saves the file in a **pcap** format
- that can be viewed by tcpdump command or a open source GUI based tool called Wireshark (Network Protocol Analyzer) that reads tcpdump **pcap** format files.

Installing in ubuntu

sudo apt-get install tcpdump

Redhat
Yum install tcpdump

Tcpdump depends on **libpcap library** for sniffing packets

Basic snipping

When you execute tcpdump command without any option, it will capture all the packets flowing through all the interfaces.

Basic sniffing

The first simple command to use is tcpdump -n (Don't convert addresses (i.e., host addresses, port numbers, etc.) to names.)

```
sudo tcpdump -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
16:34:57.266865 IP 173.194.36.6 > 192.168.1.101: ICMP echo reply, id 19941, seq 1176, length 64
16:34:57.267226 IP 192.168.1.101.21271 > 218.248.255.163.53: 23380+ PTR? 6.36.194.173.in-addr.arpa. (43)
16:34:57.274549 IP 218.248.255.163.53 > 192.168.1.101.21271: 23380 1/4/2 PTR bom04s01-in-f6.1e100.net. (195)
16:34:57.297874 IP 192.168.1.101.56295 > 186.105.77.150.38213: UDP, length 105
```

- **Why sudo** ? Because tcpdump needs root privileges to be able to capture packets on network interfaces.
- The -n parameter is given to stop tcpdump from resolving ip addresses to hostnames
- 16:34:57.267226 IP 192.168.1.101.21271 > 218.248.255.163.53: 23380+ PTR? 6.36.194.173.in-addr.arpa. (43)
- "**16:34:57.267226**" is the timestamp with microsecond precision.
- **IP 192.168.1.101.21271** protocol of the packet called IP
- **218.248.255.163.53: 23380** source ip address joined with the source port
- Followed by destination ip
- **in-addr.arpa**-which is used to convert 32-bit numeric IP addresses back into domain names

verbose switch

```
sudo tcpdump -v -n

16:43:13.058660 IP (tos 0x20, ttl 54, id 50249, offset 0, flags [DF], proto TCP (6), length 40)
    64.41.140.209.5222 > 192.168.1.101.35783: Flags [.], cksum 0x6d32 (correct), ack 1617156745, win 9648, length 0
16:43:13.214621 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto ICMP (1), length 84)
    192.168.1.101 > 173.194.36.6: ICMP echo request, id 19941, seq 1659, length 64
16:43:13.355334 IP (tos 0x20, ttl 54, id 48656, offset 0, flags [none], proto ICMP (1), length 84)
    173.194.36.6 > 192.168.1.101: ICMP echo reply, id 19941, seq 1659, length 64
```

- verbose switch lots of additional details about the packet
- include the **ttl, id, tcp flags, packet length** etc.

Getting the ethernet header

- In the above examples details of the ethernet header are not printed.
- **-e** option to print the ethernet header details

sudo tcpdump -vv -n -e

```
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
17:57:27.218531 00:25:5e:1a:3d:f1 > 00:1c:c0:f8:79:ee, ethertype IPv4 (0x0800), length 98: (tos 0x20, ttl 54, id 53046, offset 0, flags [none],
proto ICMP (1), length 84)
    173.194.36.6 > 192.168.1.101: ICMP echo reply, id 19941, seq 6015, length 64
```

Now the first thing after the timestamp is the source and destination mac address.

List of available interface

Sudo tcpdump -D

```
1.eth0
2.usbmon1 (USB bus number 1)
3.usbmon2 (USB bus number 2)
4.any (Pseudo-device that captures on all interfaces)
5.lo
```

Sniffing a particular interface

```
sudo tcpdump -i 1
sudo tcpdump -i eth0
```

Filtering packets using expressions

Selecting protocols

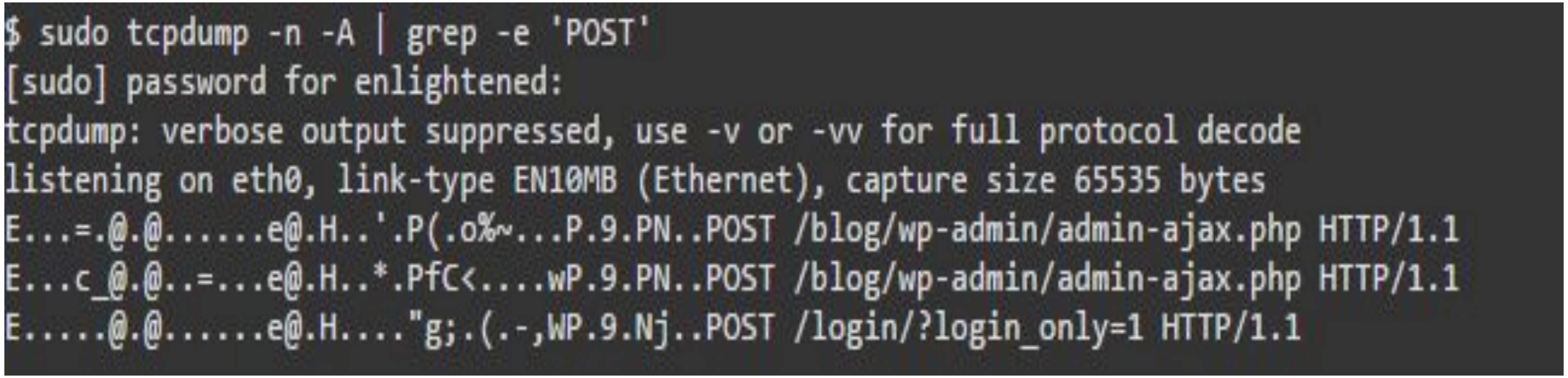
sudo tcpdump -n tcp

The above command will show only tcp packets. Similarly udp or icmp can be specified.

Particular host or port

sudo tcpdump -n ‘src 192.168.1.101’

Search the network traffic using grep



```
$ sudo tcpdump -n -A | grep -e 'POST'
[sudo] password for enlightened:
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
E...=.@.@......e@.H..'.P(.o%~...P.9.PN..POST /blog/wp-admin/admin-ajax.php HTTP/1.1
E...c_@.@..=...e@.H..*.PfC<....wP.9.PN..POST /blog/wp-admin/admin-ajax.php HTTP/1.1
E.....@.@......e@.H...."g;.(.-,WP.9.Nj..POST /login/?login_only=1 HTTP/1.1
```

-A   Print each packet in ASCII.

sniffing password

```
tcpdump port http or port ftp or port smtp or port imap or port pop3 -l -A | egrep -i
'pass=|pwd=|log=|login=|user=|username=|pw=|passw=|passwd=|password=|pass:|user:|username:|password:|login:|pass |user ' --color=auto
--line-buffered -B20
```

-l make stdout line buffered.useful if you want to see data while capturing it.

strace

- **strace** is a diagnostic, debugging and instructional userspace utility for Linux.
- It is used to monitor interactions between processes and the Linux kernel
- which include system calls, signal deliveries, and changes of process state
- The operation of strace is made possible by the kernel feature known as ptrace.
- strace uses the `ptrace` system call to attach to the target process.
- When the target process makes a system call, it will be stopped by the kernel with a `SIGTRAP` signal and strace will be notified.
- CPU registers are the main interface for exchanging data between a process in the user space and the kernel
- so they are used to implement system calls
- strace will inspect the registers and stack of the target process
- and what the arguments were. strace then resumes the process.
- When it returns from the system call, it is stopped, and strace is notified again
- so it can inspect the return value. strace prints some information for the user each time this happens.
-

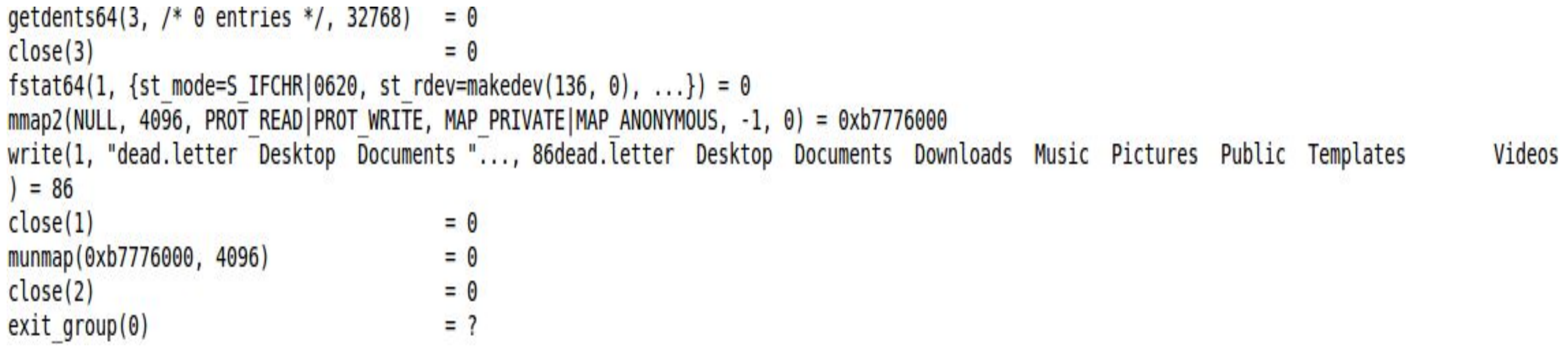
Usage

- The **most common usage is to start a program using strace, which prints a list of system calls made by the program.**
- This is useful if the program continually crashes, or does not behave as expected; for example using strace may reveal that the program is attempting to access a file which does not exist or cannot be read.
- An alternative application is to use the **-p** flag to attach to a running process.
- This is useful if a process has stopped responding, and might reveal, for example, that the process is blocking whilst attempting to make a network connection.
- As strace only details system calls, it cannot be used to detect as many problems as a code debugger

Trace the Execution of an Executable

strace command to trace the execution of any executable

```
strace ls

execve("/bin/ls", ["ls"], [/* 21 vars */]) = 0

brk(0)                                  = 0x8c31000

access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)

mmap2(NULL, 8192, PROT_READ, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb78c7000

access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)

open("/etc/ld.so.cache", O_RDONLY)      = 3

fstat64(3, {st_mode=S_IFREG|0644, st_size=65354, ...}) = 0

...
```



```
getdents64(3, /* 0 entries */, 32768)   = 0
close(3)                                = 0
fstat64(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 0), ...}) = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb7776000
write(1, "dead.letter  Desktop  Documents "..., 86dead.letter  Desktop  Documents  Downloads  Music  Pictures  Public  Templates        Videos
) = 86
close(1)                                = 0
munmap(0xb7776000, 4096)                = 0
close(2)                                = 0
exit_group(0)                           = ?
```

The above part of the output shows the write system call where it outputs to STDOUT the current directory's listing.

Trace a Specific System Calls in an Executable Using Option -e

strace -e open ls

```
open("/etc/ld.so.cache", O_RDONLY)      = 3
open("/lib/libselinux.so.1", O_RDONLY)  = 3
open("/lib/librt.so.1", O_RDONLY)       = 3
open("/lib/libacl.so.1", O_RDONLY)      = 3
open("/lib/libc.so.6", O_RDONLY)        = 3
open("/lib/libdl.so.2", O_RDONLY)       = 3
open("/lib/libpthread.so.0", O_RDONLY)  = 3
open("/lib/libattr.so.1", O_RDONLY)     = 3
open("/proc/filesystems", O_RDONLY|O_LARGEFILE) = 3
open("/usr/lib/locale/locale-archive", O_RDONLY|O_LARGEFILE) = 3
open(".", O_RDONLY|O_NONBLOCK|O_LARGEFILE|O_DIRECTORY|O_CLOEXEC) = 3
Desktop  Documents  Downloads  examples.desktop  libflashplayer.so
```

- The above output displays **only the open system call** of the ls command.
- At the end of the strace output, it also displays the output of the ls command.

trace multiple system calls use the “-e trace=” option

```
$ strace -e trace=open,read ls /home
```

```
open("/etc/ld.so.cache", O_RDONLY)      = 3

open("/lib/libselinux.so.1", O_RDONLY)  = 3

read(3, "\177ELF\1\1\1\3\3\1\260G004"..., 512) = 512

open("/lib/librt.so.1", O_RDONLY)       = 3

read(3, "\177ELF\1\1\1\3\3\1\300\30004"..., 512) = 512

..

open("/lib/libattr.so.1", O_RDONLY)     = 3

read(3, "\177ELF\1\1\1\3\3\1\360\r004"..., 512) = 512

open("/proc/filesystems", O_RDONLY|O_LARGEFILE) = 3

read(3, "nodev\tsysfs\nnodev\trootfs\nnodev\tb"..., 1024) = 315

read(3, "", 1024)                       = 0

open("/usr/lib/locale/locale-archive", O_RDONLY|O_LARGEFILE) = 3

open("/home", O_RDONLY|O_NONBLOCK|O_LARGEFILE|O_DIRECTORY|O_CLOEXEC) = 3
```

Save the Trace Execution to a File Using Option -o

```
strace -o output.txt ls
```

```
Desktop  Documents  Downloads  examples.desktop  libflashplayer.so
Music  output.txt  Pictures  Public  Templates  Ubuntu_OS  Videos
```

```
cat output.txt
```

```
execve("/bin/ls", ["ls"], [/* 37 vars */]) = 0
brk(0)                                  = 0x8637000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
mmap2(NULL, 8192, PROT_READ, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb7860000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY)      = 3
fstat64(3, {st_mode=S_IFREG|0644, st_size=67188, ...}) = 0
...
```

Execute Strace on a Running Linux Process Using Option -p

- execute strace on a program that is already running using the process id
- `ps -C firefox`

```
  PID TTY          TIME CMD
 1725 ?        00:40:50 firefox
```

```
sudo strace -p 1725 -o firefox_trace.txt
```

Print Timestamp for Each Trace Output Line Using Option -t

```
strace -t -e open ls /home
```

```
10:25:15 open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

10:25:15 open("/lib/x86_64-linux-gnu/libselinux.so.1", O_RDONLY|O_CLOEXEC) = 3

10:25:15 open("/lib/x86_64-linux-gnu/libacl.so.1", O_RDONLY|O_CLOEXEC) = 3

10:25:15 open("/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
```

Print Relative Time for System Calls Using Option -r

Strace also has the option to print the execution time for each system calls

```
strace -r ls
strace -r ls

     0.000000 execve("/bin/ls", ["ls"], [/* 37 vars */]) = 0

     0.000846 brk(0)                    = 0x8418000

     0.000143 access("/etc/ld.so.nohwcap", F_OK) = -1 ENOENT (No such file or directory)

     0.000163 mmap2(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb787b000

     0.000119 access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)

     0.000123 open("/etc/ld.so.cache", O_RDONLY) = 3
```

Generate Statistics Report of System Calls Using Option -c

Using option -c, strace provides useful statistical report for the execution trace.
“calls” column in the following output indicated how many times that particular system call was executed.

```
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 23.62    0.000128           6        23           mmap
 17.16    0.000093           7        14           mprotect
 14.02    0.000076           8         9           open
  7.56    0.000041           3        12           close
  7.38    0.000040           5         8           read
  7.38    0.000040           5         8         8 access
  5.35    0.000029          10         3           munmap
  4.80    0.000026           3        10           fstat
  3.32    0.000018           9         2           getdents
  3.32    0.000018           9         2         2 statfs
  1.48    0.000008           8         1           write
  1.11    0.000006           2         3           brk
  1.11    0.000006           3         2           ioctl
  0.92    0.000005           5         1           stat
  0.74    0.000004           4         1           openat
  0.37    0.000002           2         1           execve
  0.37    0.000002           2         1           arch_prctl
------ ----------- ----------- --------- --------- ----------------
100.00    0.000542                   101        10 total
```

ltrace

- ltrace lists all the library calls being called in an executable or a running process.
- Its name itself comes from **library-call tracing**.
- very useful for debugging user-space applications to **determine which library call is failing.**
- It is also capable of **receiving signals for segmentation faults**, etc.

Using ltrace

One does NOT need any root privileges to run ltrace.

Using ltrace to debug an application executable

ltrace ./executable <parameters>

Using ltrace to debug a running process

ltrace -p <PID>

ltrace time

```
__libc_start_main(0x8048a50, 1, 0xbfc806e4, 0x8049890, 0x8049880 <unfinished ...>
getenv("TIME")                                                     = NULL
getopt_long(1, 0xbfc806e4, "+af:o:pqvV", 0x0804c0a0, NULL)                          = -1
__fprintf_chk(0x7b6980, 1, 0x8049948, 0xbfc81dce, 0Usage: time [-apvV] [-f format] [-o file] [--append] [--verbose]
    [--portability] [--format=format] [--output=file] [--version]
    [--quiet] [--help] command [arg...]
)                               = 177
exit(1 <unfinished ...>
+++ exited (status 1) +++
```

Debugging with ltrace

- ltrace utility is most useful for debugging user-space application programs
- as the bugs could be surfaced out seeing the dynamic records of the library calls.

- Lets take a sample program.
- This program has a bug,

```
#include <stdio.h>
#include <unistd.h>

int main()
{
   FILE *fp = fopen("rfile.txt", "w+");
   fprintf(fp+1, "Invalid Write\n");
   fclose(fp);
   return 0;
}
```

Lets compile and run it.

```
gcc file.c -Wall -o file
./file
Segmentation fault (core dumped)
```

Lets ltrace the file and see whats happening.

ltrace ./file

```
__libc_start_main(0x8048454, 1, 0xbfc19db4, 0x80484c0, 0x8048530 <unfinished ...>
fopen("rfile.txt", "w+")                        = 0x9160008
fwrite("Invalid Write\n", 1, 14, 0x916009c <unfinished ...>
--- SIGSEGV (Segmentation fault) ---
+++ killed by SIGSEGV +++
```

- The first line states the libc function call 'main()', with its parameters and it terminated unfinished.
- Next line has listed the fopen() call to open 'rfile.txt' in write plus mode and returning a valid file pointer address.
- Moving on to the next library call, we have fwrite which is unfinished.
- Here is the bug because of which program crashed.
- notice the file pointer containing the address in fwrite() call doesnt match the one in fopen() call.
- The program is trying to write onto a file using file pointer which is invalid following a SIGSEGV signal.

- the bug is in line 7 of our program where the file pointer being passed to the fprintf() call is tampered. So, we just found the bug.

Corrected program

```
#include <stdio.h>
2.  #include <unistd.h>
3.
4.  int main()
5.  {
6.      FILE *fp = fopen("rfile.txt", "w+");
7.      fprintf(fp, "Invalid Write\n");
8.      fclose(fp);
9.      return 0;
10. }
```

trace output would had been much efficient to debug than re-checking each line of source or adding various printf's.

/proc

- It doesn't really exist
- Its zero-length files are neither binary nor text
- it is a virtual file system.
- the operating system creates them on the fly if you try to read them.
- This special directory holds all the details about your Linux system
- including its kernel, processes, and configuration parameters.
- Most virtual files always have a current timestamp, which indicates that they are constantly being kept up to date.
- The /proc directory itself is created every time you boot your box.
- Need to be root to examine the whole directory
- /proc directory, you’ll see two types of content — numbered directories, and system information files.
- Several Linux commands access the information from /proc, and displays in a certain format.

ls /proc

```
1 2432 3340 3715 3762 5441 815 devices modules
129 2474 3358 3716 3764 5445 acpi diskstats mounts
1290 248 3413 3717 3812 5459 asound dma mtrr
133 2486 3435 3718 3813 5479 bus execdomains partitions
1420 2489 3439 3728 3814 557 dri fb self
165 276 3450 3731 39 5842 driver filesystems slabinfo
166 280 36 3733 3973 5854 fs interrupts splash
2 2812 3602 3734 4 6 ide iomem stat
2267 3 3603 3735 40 6381 irq ioports swaps
2268 326 3614 3737 4083 6558 net kallsyms sysrq-trigger
2282 327 3696 3739 4868 6561 scsi kcore timer_list
2285 3284 3697 3742 4873 6961 sys keys timer_stats
2295 329 3700 3744 4878 7206 sysvipc key-users uptime
2335 3295 3701 3745 5 7207 tty kmsg version
2400 330 3706 3747 5109 7222 buddyinfo loadavg vmcore
2401 3318 3709 3749 5112 7225 cmdline locks vmstat
2427 3329 3710 3751 541 7244 config.gz meminfo zoneinfo
2428 3336 3714 3753 5440 752 cpuinfo misc
```

- lot of directories with just numbers.
- These numbers represents the process ids
- the files inside this numbered directory corresponds to the process with that particular PID

Some Important files under numbered directory

- cmdline – command line of the command.
- environ – environment variables.
- fd – Contains the file descriptors which is linked to the appropriate files
- limits – Contains the information about the specific limits to the process.
- mounts – mount related information

Important links under each numbered directory

- cwd – Link to current working directory of the process.
- exe – Link to executable of the process.
- root – Link to the root directory of the process.
-

/proc Files about the system information

- /proc/cpuinfo – information about CPU,
- /proc/meminfo – information about memory,
- /proc/loadvg – load average,
- /proc/partitions – partition related information,
- /proc/version – linux version

Some Linux commands read the information from this /proc files and displays it. For example, **free command**, reads the memory information from **/proc/meminfo file**, formats it, and displays it.

Example cd /proc/12

cat /proc/12/status

```
Name:	xenwatch
State:	S (sleeping)
Tgid:	12
Pid:	12
PPid:	2
TracerPid:	0
Uid:	0	0	0	0
Gid:	0	0	0	0
FDSize:	64
Groups:
Threads:	1
SigQ:	1/4592
SigPnd:	0000000000000000
ShdPnd:	0000000000000000
SigBlk:	0000000000000000
SigIgn:	ffffffffffffffff
SigCgt:	0000000000000000
CapInh:	0000000000000000
CapPrm:	ffffffffffffffff
CapEff:	ffffffffffffffff
CapBnd:	ffffffffffffffff
Cpus_allowed:	1
Cpus_allowed_list:	0
Mems_allowed:	00000000,00000001
Mems_allowed_list:	0
voluntary_ctxt_switches:	84
```

- this process belongs to **xenwatch**
- Its current state is **sleeping**
- its process **ID** is **12**
- We also can see who is running this, as **UID** and **GID** are **0**, indicating that this process belongs to the **root** user.
-

free

- **'free' shows you information about the machine's memory.**
- **This includes**
  physical memory (RAM), swap as well as the shared memory and buffers
  used by the kernel.
- **All measurements are in Kb.**

```
 free
         total          used          free          shared        buffers       cached
Mem:  7993580         4765408       3228172       640200        189588        2242860
-/+ buffers/cache:    2332960       5660620
Swap:           0             0             0
```

total: total ram.
used: This is a **mix of application used memory and other 'temporarily' (buffer + cache) used memory** that is actually available if needed.
The 'temporarily' used memory is borrowed if available by the linux system to help speed up system performance,
Much of this type of memory is shown under the 'cached' column.
This memory is given up by the linux system if an application need memory.

free: free and untouched memory.

shared: Memory specifically **allocated** for use by **multiple processes**

buffers: Temporary memory that is set aside to help some processes

cache: Memory that is available and 'borrowed' by the operating system to help speed up many linux OS operations. This memory is given up by the system if an application need it.

-/+ buffers/cache - If you're wanting to get a good idea on how **much free memory is available**, the **free** section in the **buffers/cache** is what you should be reading.

-/+ buffers/cache is typically more helpful than the first **Mem** line.

The **intersection** of **free** and **-/+ buffers/cache** is essentially what you have for **'available' memory**.

Swap - memory management involving swapping regions of memory to and from storage.

Display Memory in Bytes

free -b

```
             total       used       free     shared    buffers     cached
Mem:    1046147072  934420480  111726592          0  123256832  671281152
-/+ buffers/cache:  139882496  906264576
Swap:   4294959104          0 4294959104
```

Display Memory in Kilo Bytes

free -k

```
             total       used       free     shared    buffers     cached
Mem:       1021628     912520     109108          0     120368     655548
-/+ buffers/cache:     136604     885024
Swap:      4194296          0    4194296
```

Display Memory in Megabytes

free -m

```
             total       used       free     shared    buffers     cached
Mem:           997        891        106          0        117        640
-/+ buffers/cache:        133        864
Swap:         4095          0       4095
```

Display Total Line

Free command with **-t** option, will list the total line at the end.

```
free -t
                  total       used        free        shared      buffers     cached
Mem:              7993580     5188684     2804896     624840      200980      2372464
-/+ buffers/cache:2615240     5378340
Swap:             0        0        0
Total:            7993580     5188684     2804896
```

Disable Display of Buffer Adjusted Line

free -o

```
          total       used       free     shared    buffers     cached
Mem:       1021628     912520     109108          0     120368     655548
Swap:      4194296          0    4194296
```

Display Memory Status for Regular Intervals

the below command will update free command every **3 seconds**.

free -s 3

```
             total     used      free      shared    buffers   cached
Mem:         7993580   5169700   2823880   639732    201472    2387352
-/+ buffers/cache:  2580876   5412704
Swap:        0    0    0

             total     used      free      shared    buffers   cached
Mem:         7993580   5165896   2827684   639404    201484    2387024
-/+ buffers/cache:  2577388   5416192
Swap:        0    0    0

             total     used      free      shared    buffers   cached
Mem:         7993580   5165468   2828112   639404    201484    2387024
-/+ buffers/cache:  2576960   5416620
Swap:        0    0    0
```

Show Low and High Memory Statistics

The **-l** switch displays detailed high and low memory size statistics.

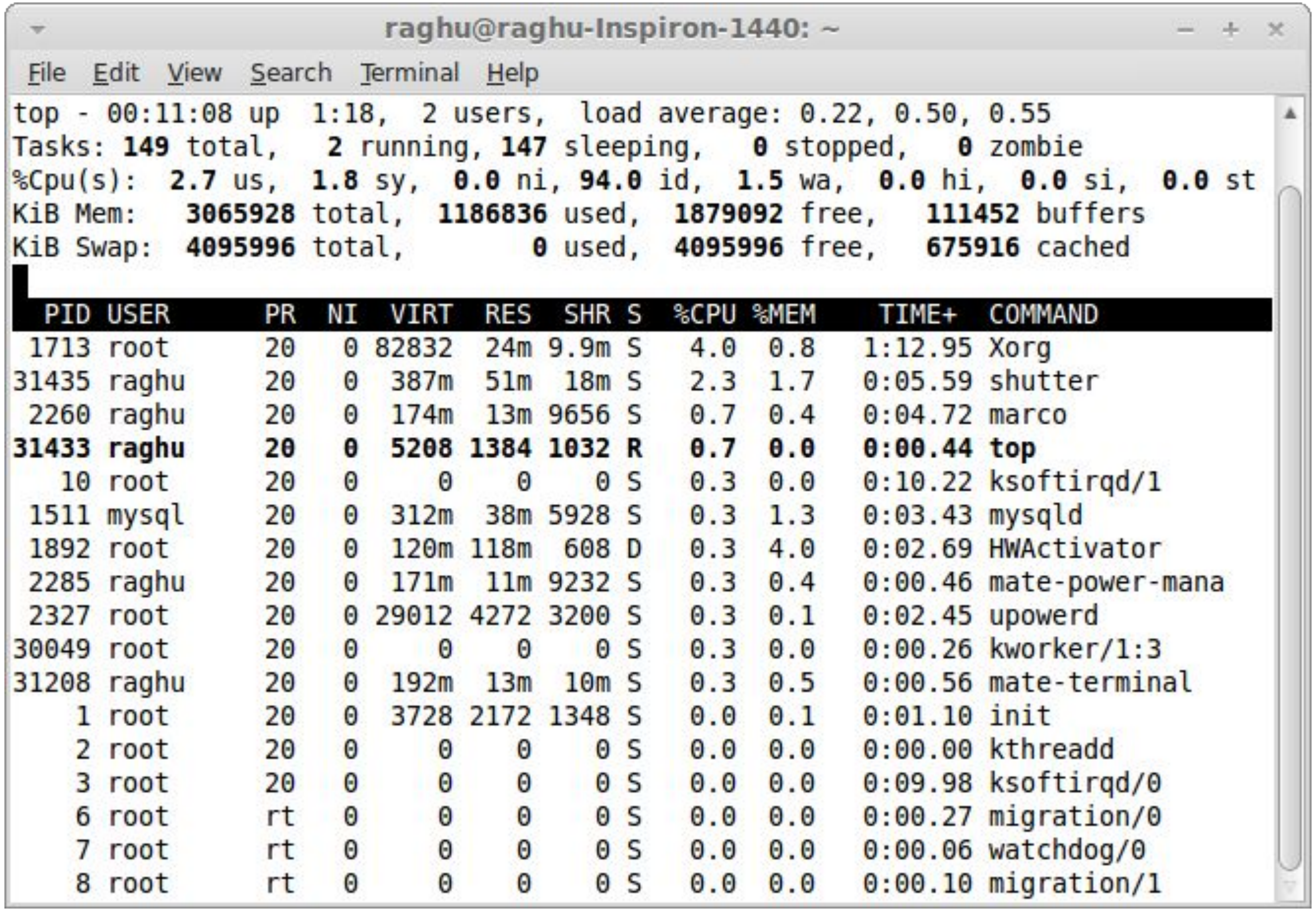
free -l

```
             total       used       free     shared    buffers     cached
Mem:       1021628     912368     109260          0     120368     655548
Low:        890036     789064     100972
High:       131592     123304       8288
-/+ buffers/cache:     136452     885176
Swap:      4194296          0    4194296
```

top

- The top command in Linux **displays the running processes on the system.**
- It is used extensively for **monitoring the load on a server**.
- The top command is an interactive command
- **top** provides an ongoing look at **processor activity in real time**.
- displays the top 30 processes on the system and periodically updates this information.

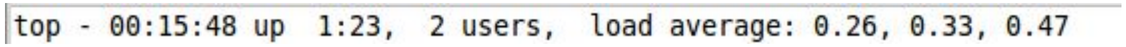
$ **top**


raghu@raghu-Inspiron-1440: ~

File Edit View Search Terminal Help

```
top - 00:11:08 up  1:18,  2 users,  load average: 0.22, 0.50, 0.55
Tasks: 149 total,   2 running, 147 sleeping,   0 stopped,   0 zombie
%Cpu(s):  2.7 us,  1.8 sy,  0.0 ni, 94.0 id,  1.5 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:   3065928 total,  1186836 used,  1879092 free,   111452 buffers
KiB Swap:  4095996 total,        0 used,  4095996 free,   675916 cached

  PID USER      PR  NI  VIRT  RES  SHR S  %CPU %MEM    TIME+  COMMAND
 1713 root      20   0 82832  24m 9.9m S   4.0  0.8   1:12.95 Xorg
31435 raghu     20   0  387m  51m  18m S   2.3  1.7   0:05.59 shutter
 2260 raghu     20   0  174m  13m 9656 S   0.7  0.4   0:04.72 marco
31433 raghu     20   0  5208 1384 1032 R   0.7  0.0   0:00.44 top
   10 root      20   0     0    0    0 S   0.3  0.0   0:10.22 ksoftirqd/1
 1511 mysql     20   0  312m  38m 5928 S   0.3  1.3   0:03.43 mysqld
 1892 root      20   0  120m 118m  608 D   0.3  4.0   0:02.69 HWActivator
 2285 raghu     20   0  171m  11m 9232 S   0.3  0.4   0:00.46 mate-power-mana
 2327 root      20   0 29012 4272 3200 S   0.3  0.1   0:02.45 upowerd
30049 root      20   0     0    0    0 S   0.3  0.0   0:00.26 kworker/1:3
31208 raghu     20   0  192m  13m  10m S   0.3  0.5   0:00.56 mate-terminal
    1 root      20   0  3728 2172 1348 S   0.0  0.1   0:01.10 init
    2 root      20   0     0    0    0 S   0.0  0.0   0:00.00 kthreadd
    3 root      20   0     0    0    0 S   0.0  0.0   0:09.98 ksoftirqd/0
    6 root      rt   0     0    0    0 S   0.0  0.0   0:00.27 migration/0
    7 root      rt   0     0    0    0 S   0.0  0.0   0:00.06 watchdog/0
    8 root      rt   0     0    0    0 S   0.0  0.0   0:00.10 migration/1
```

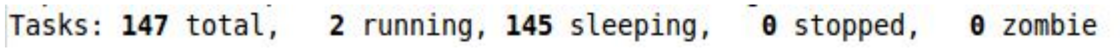
Uptime and Load Averages:



```
top - 00:15:48 up  1:23,  2 users,  load average: 0.26, 0.33, 0.47
```

- output similar to uptime command.
- The fields display:
- current time
- the time your system is been up
- number of users logged in
- load average of 5, 10 and 15 minutes respectively.

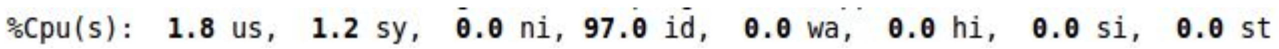
Tasks:



```
Tasks: 147 total,   2 running, 145 sleeping,   0 stopped,   0 zombie
```

- The second line shows summary of tasks or processes. The processes can be in different states.
- It shows total number of the processes
- Out of these, the processes can be running, sleeping, stopped or in zombie

CPU States:

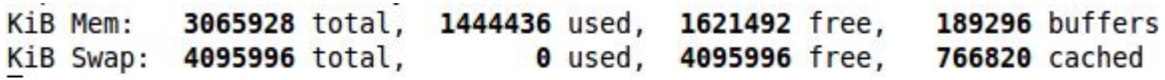


```
%Cpu(s):  1.8 us,  1.2 sy,  0.0 ni, 97.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
```

% of CPU time in different modes is shown. The meaning of different CPU times are:

***us**, user: CPU time in running (un-niced) user processes
* **sy**, system: CPU time in running kernel processes
* **ni**, niced: CPU time in running niced user processes
* **wa**, IO wait: CPU time waiting for IO completion
* **hi**: CPU time serving hardware interrupts
* **si**: CPU time serving software interrupts
* **st**: CPU time stolen for this vm by the hypervisor.

Memory Usage:

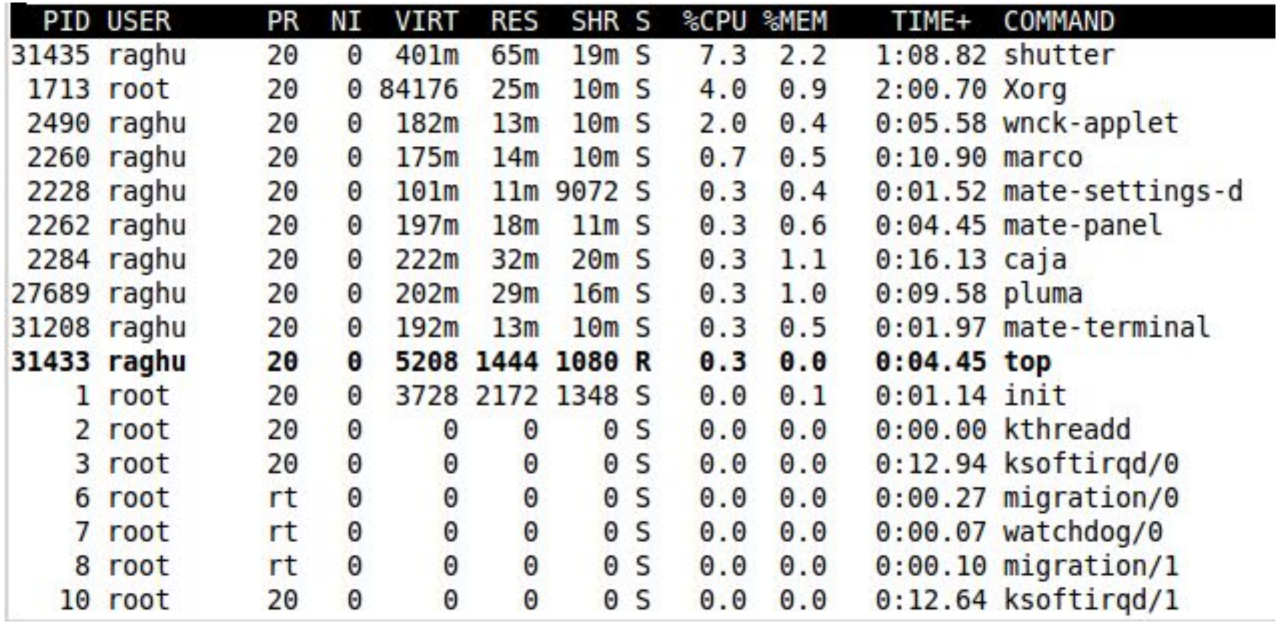


```
KiB Mem:   3065928 total,  1444436 used,  1621492 free,   189296 buffers
KiB Swap:  4095996 total,        0 used,  4095996 free,   766820 cached
```

- two lines show memory usage
- somewhat like 'free' command.
- 1st of these lines is for **physical memory** and the second for **virtual memory** (swap space).

- The physical memory is displayed as: total available memory, used memory, free memory, and memory used for buffers
- Similarly, swap reflects: total, used, free and cached swap space.

Fields/Columns:



```
  PID USER      PR  NI  VIRT  RES  SHR S  %CPU %MEM    TIME+  COMMAND
31435 raghu     20   0  401m  65m  19m S   7.3  2.2   1:08.82 shutter
 1713 root      20   0 84176  25m  10m S   4.0  0.9   2:00.70 Xorg
 2490 raghu     20   0  182m  13m  10m S   2.0  0.4   0:05.58 wnck-applet
 2260 raghu     20   0  175m  14m  10m S   0.7  0.5   0:10.90 marco
 2228 raghu     20   0  101m  11m 9072 S   0.3  0.4   0:01.52 mate-settings-d
 2262 raghu     20   0  197m  18m  11m S   0.3  0.6   0:04.45 mate-panel
 2284 raghu     20   0  222m  32m  20m S   0.3  1.1   0:16.13 caja
27689 raghu     20   0  202m  29m  16m S   0.3  1.0   0:09.58 pluma
31208 raghu     20   0  192m  13m  10m S   0.3  0.5   0:01.97 mate-terminal
31433 raghu     20   0  5208 1444 1080 R   0.3  0.0   0:04.45 top
    1 root      20   0  3728 2172 1348 S   0.0  0.1   0:01.14 init
    2 root      20   0     0    0    0 S   0.0  0.0   0:00.00 kthreadd
    3 root      20   0     0    0    0 S   0.0  0.0   0:12.94 ksoftirqd/0
    6 root      rt   0     0    0    0 S   0.0  0.0   0:00.27 migration/0
    7 root      rt   0     0    0    0 S   0.0  0.0   0:00.07 watchdog/0
    8 root      rt   0     0    0    0 S   0.0  0.0   0:00.10 migration/1
   10 root      20   0     0    0    0 S   0.0  0.0   0:12.64 ksoftirqd/1
```

PID

The Process ID, to uniquely identify a processes.

USER
The effective user name of the owner of the processes.

PR
The **scheduling priority of the process**. Some values in this field are '**rt**'. It means that the **process is running under real-time**.

NI
The **nice value of the process**. Lower values mean higher priority.

VIRT
The amount of **virtual memory used by the process**.

RES
The **resident memory size**. Resident memory is the amount of non-swapped **physical memory a task is using**.

SHR
SHR is the **shared memory** used by the process.

S
This is the process status. It can have one of the following values:
D - uninterruptible sleep
R - running
S - sleeping
T - traced or stopped
Z - zombie

%CPU
It is the **percentage of CPU time the task has used** since last update.

%MEM
Percentage of **available physical memory used by the process.**

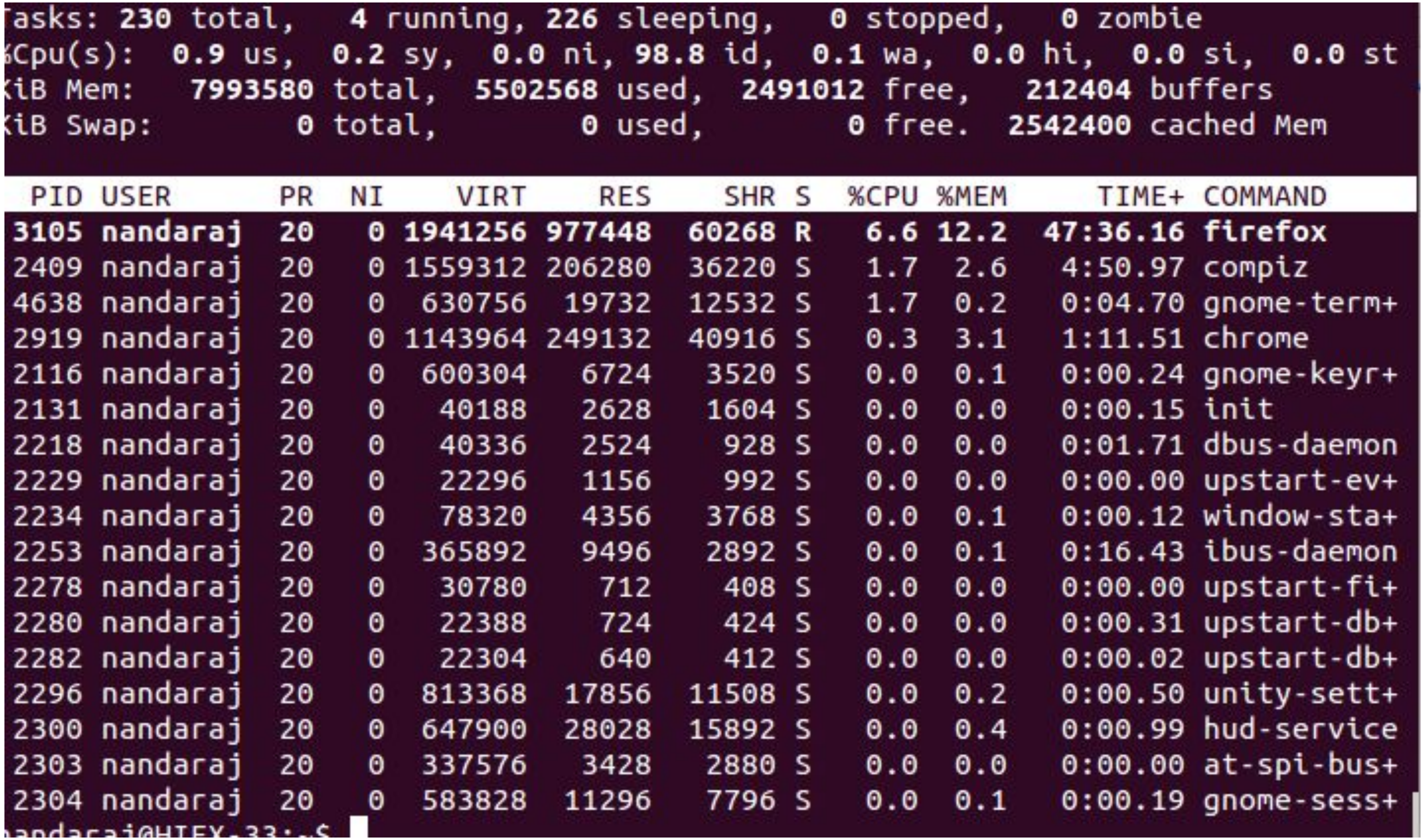
TIME+
The **total CPU time the task has used since it started**, with precision upto hundredth of a second.

COMMAND
The command which was used to start the process.

Display Specific User Process

Use top command with ‘**u**‘ option will display specific **User** process details.

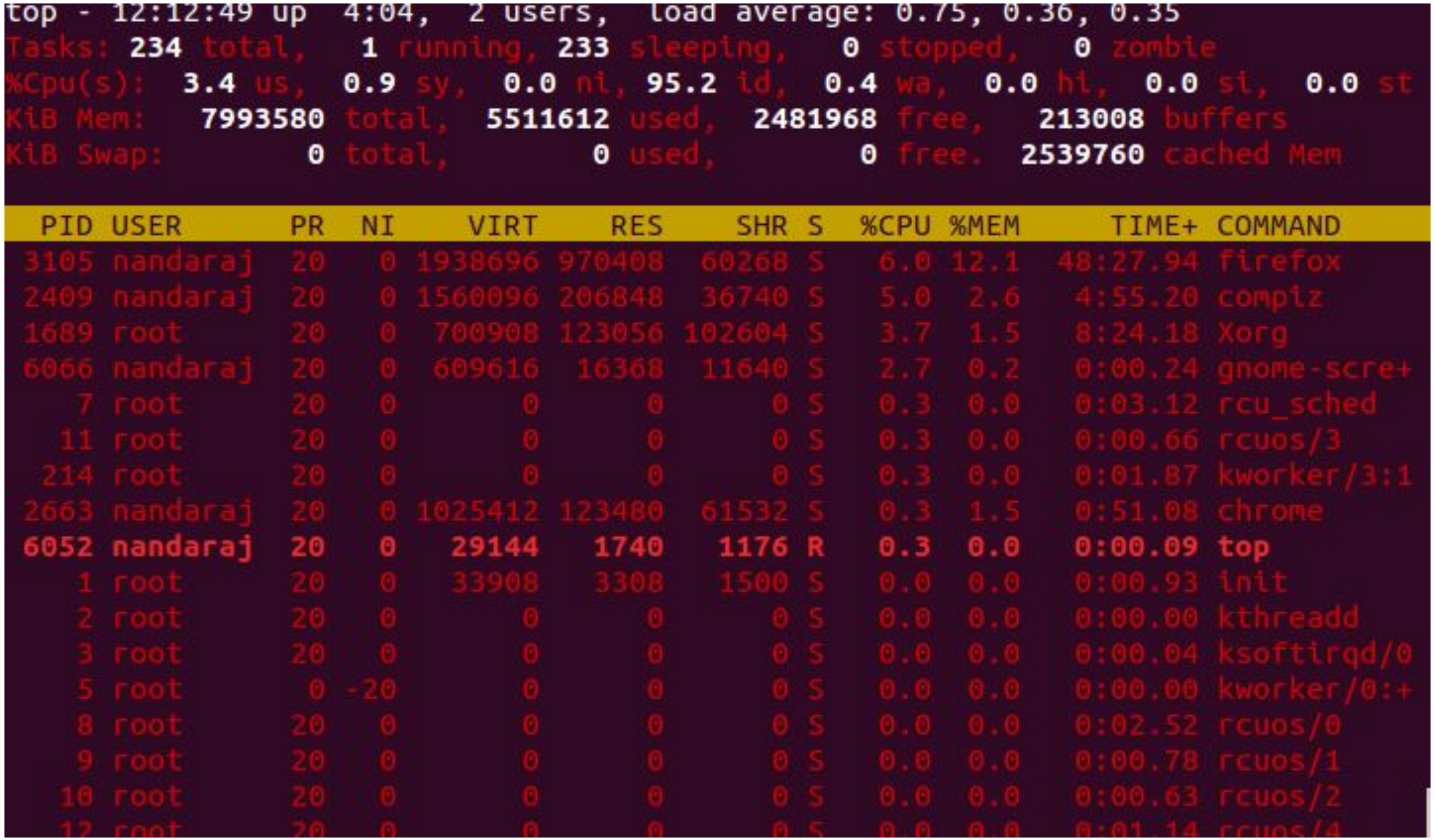
top -u nandaraj



```
Tasks: 230 total,   4 running, 226 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.9 us,  0.2 sy,  0.0 ni, 98.8 id,  0.1 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:   7993580 total,  5502568 used,  2491012 free,   212404 buffers
KiB Swap:        0 total,        0 used,        0 free.  2542400 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 3105 nandaraj  20   0 1941256 977448  60268 R   6.6 12.2  47:36.16 firefox
 2409 nandaraj  20   0 1559312 206280  36220 S   1.7  2.6   4:50.97 compiz
 4638 nandaraj  20   0  630756  19732  12532 S   1.7  0.2   0:04.70 gnome-term+
 2919 nandaraj  20   0 1143964 249132  40916 S   0.3  3.1   1:11.51 chrome
 2116 nandaraj  20   0  600304   6724   3520 S   0.0  0.1   0:00.24 gnome-keyr+
 2131 nandaraj  20   0   40188   2628   1604 S   0.0  0.0   0:00.15 init
 2218 nandaraj  20   0   40336   2524    928 S   0.0  0.0   0:01.71 dbus-daemon
 2229 nandaraj  20   0   22296   1156    992 S   0.0  0.0   0:00.00 upstart-ev+
 2234 nandaraj  20   0   78320   4356   3768 S   0.0  0.1   0:00.12 window-sta+
 2253 nandaraj  20   0  365892   9496   2892 S   0.0  0.1   0:16.43 ibus-daemon
 2278 nandaraj  20   0   30780    712    408 S   0.0  0.0   0:00.00 upstart-fi+
 2280 nandaraj  20   0   22388    724    424 S   0.0  0.0   0:00.31 upstart-db+
 2282 nandaraj  20   0   22304    640    412 S   0.0  0.0   0:00.02 upstart-db+
 2296 nandaraj  20   0  813368  17856  11508 S   0.0  0.2   0:00.50 unity-sett+
 2300 nandaraj  20   0  647900  28028  15892 S   0.0  0.4   0:00.99 hud-service
 2303 nandaraj  20   0  337576   3428   2880 S   0.0  0.0   0:00.00 at-spi-bus+
 2304 nandaraj  20   0  583828  11296   7796 S   0.0  0.1   0:00.19 gnome-sess+
nandaraj@HIFX-33:~$
```

Highlight Running Process in Top

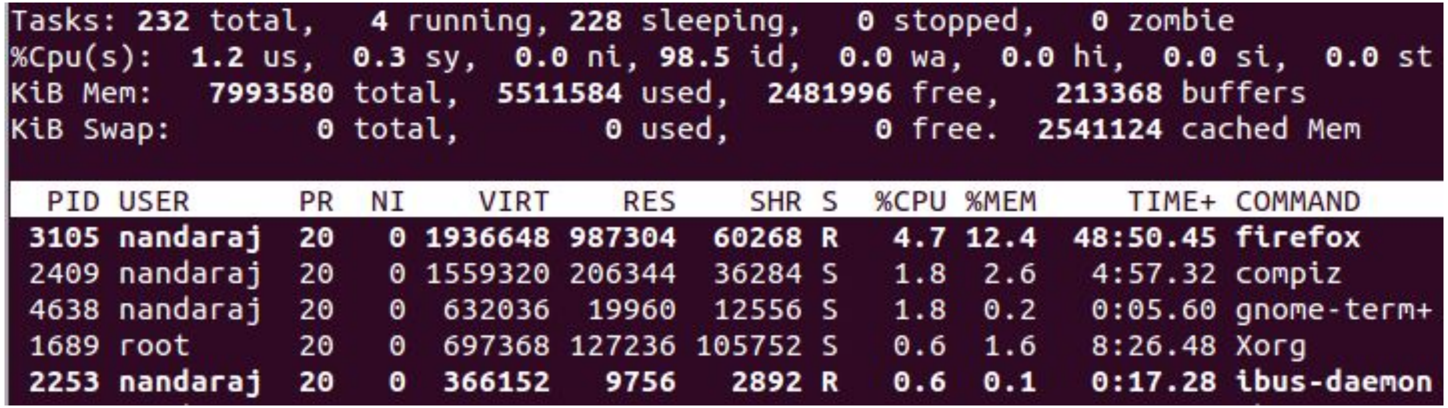
Press ‘**z**‘ option in running top command will display running process in color which may help you to identified running process easily.



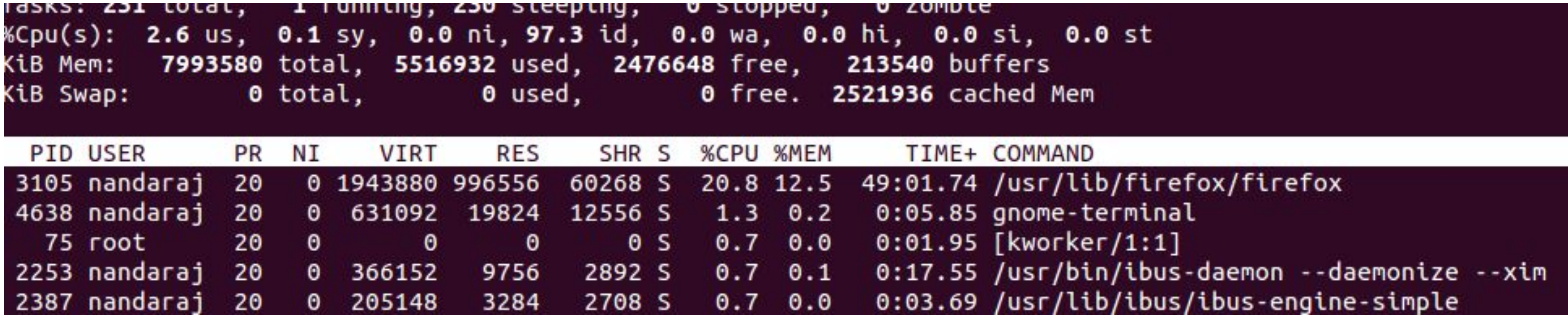
```
top - 12:12:49 up  4:04,  2 users,  load average: 0.75, 0.36, 0.35
Tasks: 234 total,   1 running, 233 sleeping,   0 stopped,   0 zombie
%Cpu(s):  3.4 us,  0.9 sy,  0.0 ni, 95.2 id,  0.4 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:   7993580 total,  5511612 used,  2481968 free,   213008 buffers
KiB Swap:        0 total,        0 used,        0 free.  2539760 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 3105 nandaraj  20   0 1938696 970408  60268 S   6.0 12.1  48:27.94 firefox
 2409 nandaraj  20   0 1560096 206848  36740 S   5.0  2.6   4:55.20 compiz
 1689 root      20   0  700908 123056 102604 S   3.7  1.5   8:24.18 Xorg
 6066 nandaraj  20   0  609616  16368  11640 S   2.7  0.2   0:00.24 gnome-scre+
    7 root      20   0       0      0      0 S   0.3  0.0   0:03.12 rcu_sched
   11 root      20   0       0      0      0 S   0.3  0.0   0:00.66 rcuos/3
  214 root      20   0       0      0      0 S   0.3  0.0   0:01.87 kworker/3:1
 2663 nandaraj  20   0 1025412 123480  61532 S   0.3  1.5   0:51.08 chrome
 6052 nandaraj  20   0   29144   1740   1176 R   0.3  0.0   0:00.09 top
    1 root      20   0   33908   3308   1500 S   0.0  0.0   0:00.93 init
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.00 kthreadd
    3 root      20   0       0      0      0 S   0.0  0.0   0:00.04 ksoftirqd/0
    5 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:+
    8 root      20   0       0      0      0 S   0.0  0.0   0:02.52 rcuos/0
    9 root      20   0       0      0      0 S   0.0  0.0   0:00.78 rcuos/1
   10 root      20   0       0      0      0 S   0.0  0.0   0:00.63 rcuos/2
   12 root      20   0       0      0      0 S   0.0  0.0   0:01.14 rcuos/4
```

Shows Absolute Path of Processes

Press ‘**c**‘ option in running top command, it will display absolute path of running process.



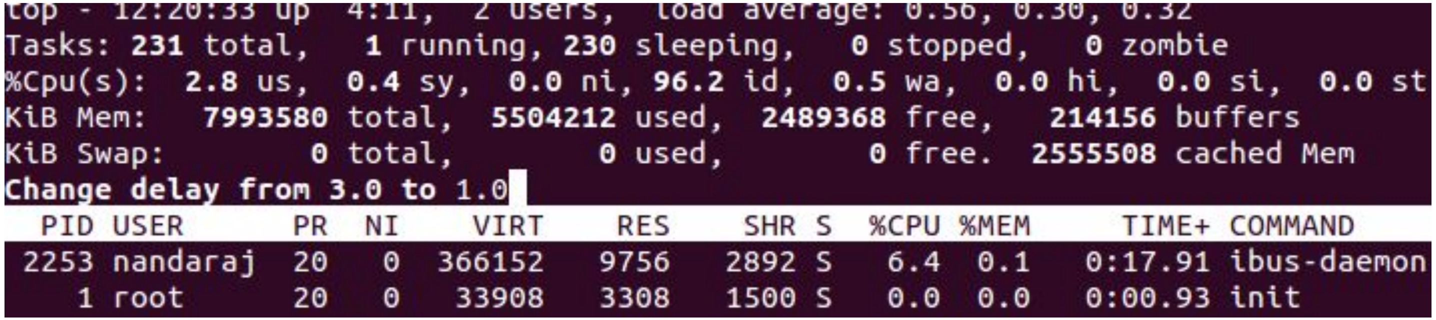
```
Tasks: 232 total,   4 running, 228 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1.2 us,  0.3 sy,  0.0 ni, 98.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:   7993580 total,  5511584 used,  2481996 free,   213368 buffers
KiB Swap:        0 total,        0 used,        0 free.  2541124 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 3105 nandaraj  20   0 1936648 987304  60268 R   4.7 12.4  48:50.45 firefox
 2409 nandaraj  20   0 1559320 206344  36284 S   1.8  2.6   4:57.32 compiz
 4638 nandaraj  20   0  632036  19960  12556 S   1.8  0.2   0:05.60 gnome-term+
 1689 root      20   0  697368 127236 105752 S   0.6  1.6   8:26.48 Xorg
 2253 nandaraj  20   0  366152   9756   2892 R   0.6  0.1   0:17.28 ibus-daemon
```



```
Tasks: 231 total,   1 running, 230 sleeping,   0 stopped,   0 zombie
%Cpu(s):  2.6 us,  0.1 sy,  0.0 ni, 97.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:   7993580 total,  5516932 used,  2476648 free,   213540 buffers
KiB Swap:        0 total,        0 used,        0 free.  2521936 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 3105 nandaraj  20   0 1943880 996556  60268 S  20.8 12.5  49:01.74 /usr/lib/firefox/firefox
 4638 nandaraj  20   0  631092  19824  12556 S   1.3  0.2   0:05.85 gnome-terminal
   75 root      20   0       0      0      0 S   0.7  0.0   0:01.95 [kworker/1:1]
 2253 nandaraj  20   0  366152   9756   2892 S   0.7  0.1   0:17.55 /usr/bin/ibus-daemon --daemonize --xim
 2387 nandaraj  20   0  205148   3284   2708 S   0.7  0.0   0:03.69 /usr/lib/ibus/ibus-engine-simple
```

Set ‘Screen Refresh Interval’ in Top

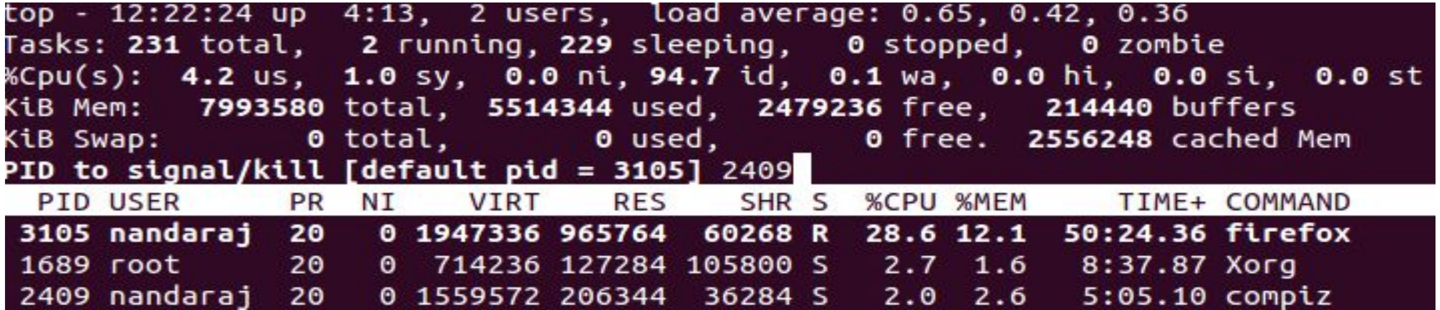
By default screen refresh interval is **3.0** seconds, same can be changed by pressing ‘**d**‘ option in running top command



```
top - 12:20:33 up  4:11,  2 users,  load average: 0.56, 0.30, 0.32
Tasks: 231 total,   1 running, 230 sleeping,   0 stopped,   0 zombie
%Cpu(s):  2.8 us,  0.4 sy,  0.0 ni, 96.2 id,  0.5 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:   7993580 total,  5504212 used,  2489368 free,   214156 buffers
KiB Swap:        0 total,        0 used,        0 free.  2555508 cached Mem
Change delay from 3.0 to 1.0
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 2253 nandaraj  20   0  366152   9756   2892 S   6.4  0.1   0:17.91 ibus-daemon
    1 root      20   0   33908   3308   1500 S   0.0  0.0   0:00.93 init
```

Kill running process with argument ‘k’

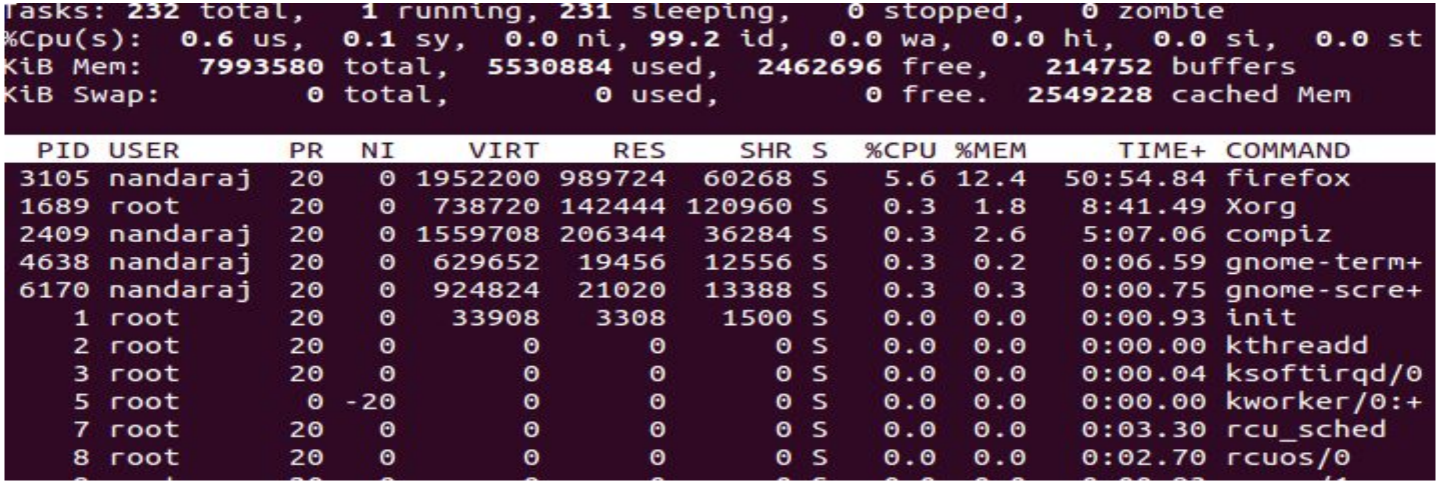
You can kill a process after finding **PID** of process by pressing ‘**k**‘ option in running top command without exiting from top window as shown below.



```
top - 12:22:24 up  4:13,  2 users,  load average: 0.65, 0.42, 0.36
Tasks: 231 total,   2 running, 229 sleeping,   0 stopped,   0 zombie
%Cpu(s):  4.2 us,  1.0 sy,  0.0 ni, 94.7 id,  0.1 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:   7993580 total,  5514344 used,  2479236 free,   214440 buffers
KiB Swap:        0 total,        0 used,        0 free.  2556248 cached Mem
PID to signal/kill [default pid = 3105] 2409
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 3105 nandaraj  20   0 1947336 965764  60268 R  28.6 12.1  50:24.36 firefox
 1689 root      20   0  714236 127284 105800 S   2.7  1.6   8:37.87 Xorg
 2409 nandaraj  20   0 1559572 206344  36284 S   2.0  2.6   5:05.10 compiz
```

Sort by CPU Utilisation

Press (**Shift+P**) to sort processes as per **CPU** utilization.



```
Tasks: 232 total,   1 running, 231 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.6 us,  0.1 sy,  0.0 ni, 99.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:   7993580 total,  5530884 used,  2462696 free,   214752 buffers
KiB Swap:        0 total,        0 used,        0 free.  2549228 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 3105 nandaraj  20   0 1952200 989724  60268 S   5.6 12.4  50:54.84 firefox
 1689 root      20   0  738720 142444 120960 S   0.3  1.8   8:41.49 Xorg
 2409 nandaraj  20   0 1559708 206344  36284 S   0.3  2.6   5:07.06 compiz
 4638 nandaraj  20   0  629652  19456  12556 S   0.3  0.2   0:06.59 gnome-term+
 6170 nandaraj  20   0  924824  21020  13388 S   0.3  0.3   0:00.75 gnome-scre+
    1 root      20   0   33908   3308   1500 S   0.0  0.0   0:00.93 init
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.00 kthreadd
    3 root      20   0       0      0      0 S   0.0  0.0   0:00.04 ksoftirqd/0
    5 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:+
    7 root      20   0       0      0      0 S   0.0  0.0   0:03.30 rcu_sched
    8 root      20   0       0      0      0 S   0.0  0.0   0:02.70 rcuos/0
```

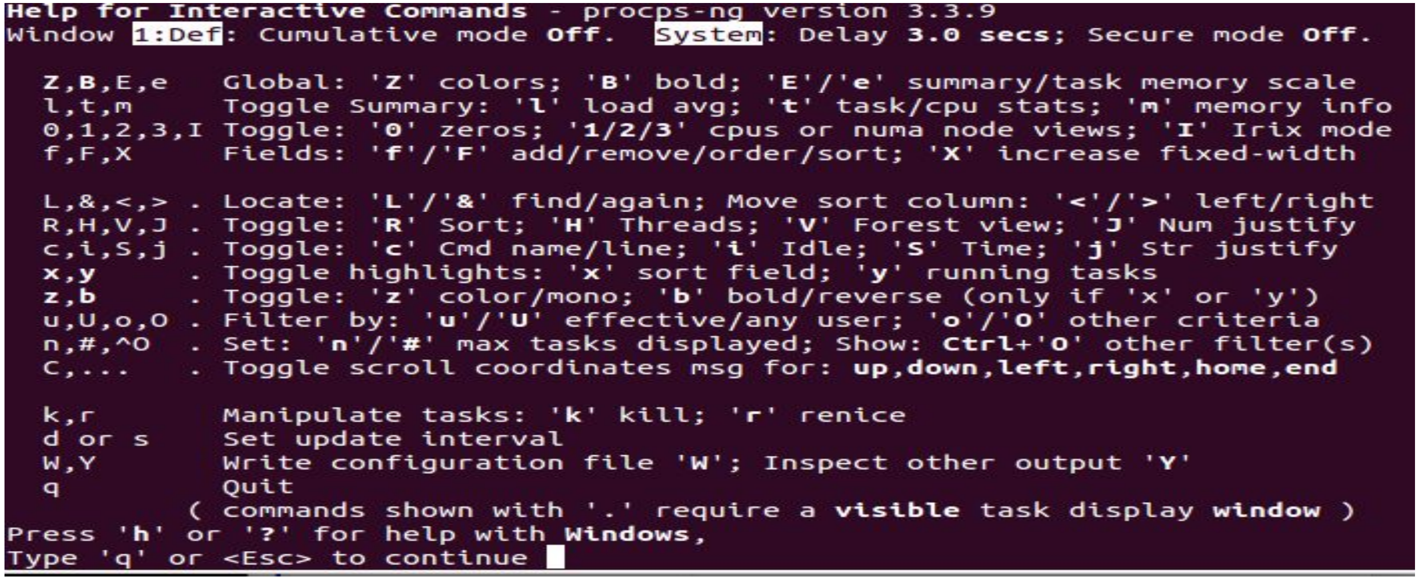
Exit Top Command After Specific repetition

Top output keep refreshing until you press ‘**q**‘. With below command top command will automatically exit after 10 number of repetition.

# top -n 10

Getting Top Command Help

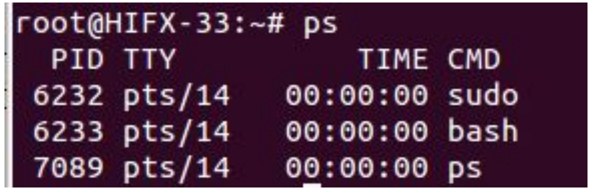
Press ‘**h**‘ option to obtain the top command help.



```
Help for Interactive Commands - procps-ng version 3.3.9
Window 1:Def: Cumulative mode Off.  System: Delay 3.0 secs; Secure mode Off.

  Z,B,E,e   Global: 'Z' colors; 'B' bold; 'E'/'e' summary/task memory scale
  l,t,m     Toggle Summary: 'l' load avg; 't' task/cpu stats; 'm' memory info
  0,1,2,3,I Toggle: '0' zeros; '1/2/3' cpus or numa node views; 'I' Irix mode
  f,F,X     Fields: 'f'/'F' add/remove/order/sort; 'X' increase fixed-width

  L,&,<,> . Locate: 'L'/'&' find/again; Move sort column: '<'/'>' left/right
  R,H,V,J . Toggle: 'R' Sort; 'H' Threads; 'V' Forest view; 'J' Num justify
  c,i,S,j . Toggle: 'c' Cmd name/line; 'i' Idle; 'S' Time; 'j' Str justify
  x,y     . Toggle highlights: 'x' sort field; 'y' running tasks
  z,b     . Toggle: 'z' color/mono; 'b' bold/reverse (only if 'x' or 'y')
  u,U,o,O . Filter by: 'u'/'U' effective/any user; 'o'/'O' other criteria
  n,#,^O  . Set: 'n'/'#' max tasks displayed; Show: Ctrl+'O' other filter(s)
  C,...   . Toggle scroll coordinates msg for: up,down,left,right,home,end

  k,r       Manipulate tasks: 'k' kill; 'r' renice
  d or s    Set update interval
  W,Y       Write configuration file 'W'; Inspect other output 'Y'
  q         Quit
          ( commands shown with '.' require a visible task display window )
Press 'h' or '?' for help with Windows,
Type 'q' or <Esc> to continue
```

ps

ps

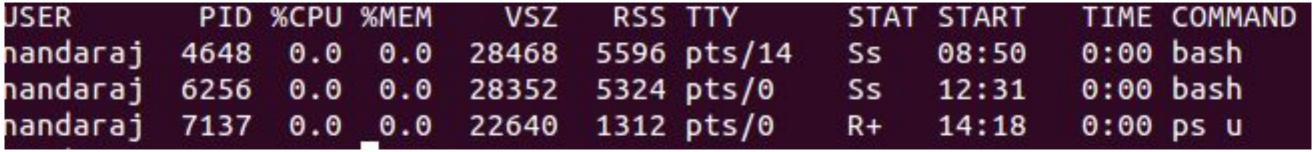
- **ps** program (short for "**p**rocess **s**tatus") **displays the currently-running processes**
- ps
- 

```
root@HIFX-33:~# ps
  PID TTY          TIME CMD
 6232 pts/14   00:00:00 sudo
 6233 pts/14   00:00:00 bash
 7089 pts/14   00:00:00 ps
```

PID -   The process ID of the process.
TTY   The **controlling terminal** for the process.
TIME - The **CPU time the process has used since the process started**. The format for this time is [dd-]hh:mm:ss.
CMD - The name of the command that is running

Display User-Oriented Output (u)

ps u



```
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
handaraj   4648  0.0  0.0  28468  5596 pts/14   Ss   08:50   0:00 bash
handaraj   6256  0.0  0.0  28352  5324 pts/0    Ss   12:31   0:00 bash
handaraj   7137  0.0  0.0  22640  1312 pts/0    R+   14:18   0:00 ps u
```

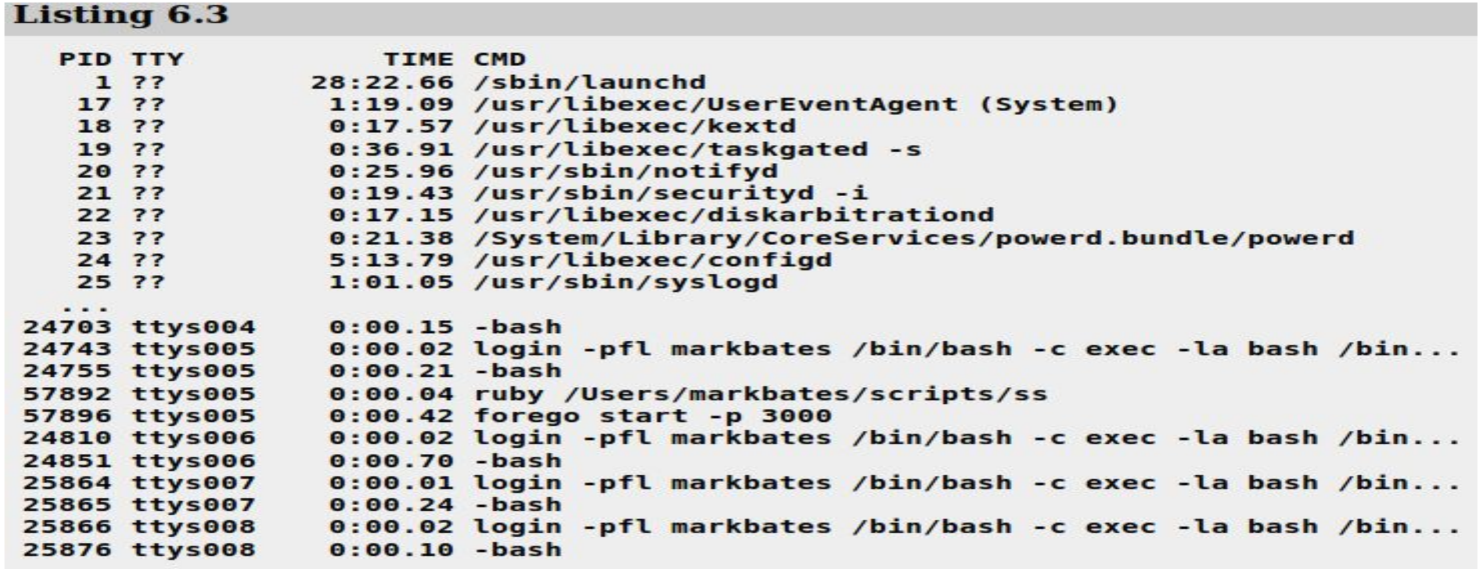
USER-The user that the process is running under.
%CPU-The **percentage of time the process has used the CPU** since the process started.
%MEM-The percentage of **real memory used** by this process.
VSZ-Indicates, as a decimal integer, the **size in kilobytes of the process in virtual memory**.
RSS-The **real-memory size of the process** (in 1 KB units).
STAT-The status of the process.
STARTED-When the process was started.

Display All Processes (-e)

Display all processes in the system irrespective of user.

mix of “system” processes as well as “user” processes.

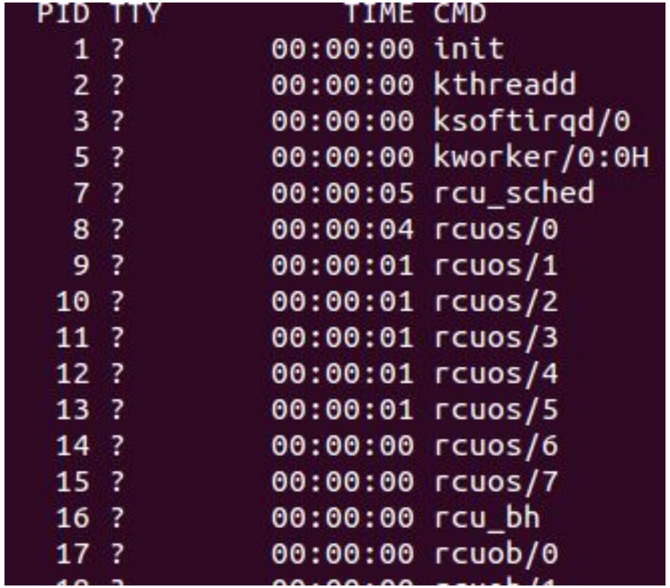
ps -e



Listing 6.3

```
  PID TTY           TIME CMD
    1 ??        28:22.66 /sbin/launchd
   17 ??         1:19.09 /usr/libexec/UserEventAgent (System)
   18 ??         0:17.57 /usr/libexec/kextd
   19 ??         0:36.91 /usr/libexec/taskgated -s
   20 ??         0:25.96 /usr/sbin/notifyd
   21 ??         0:19.43 /usr/sbin/securityd -i
   22 ??         0:17.15 /usr/libexec/diskarbitrationd
   23 ??         0:21.38 /System/Library/CoreServices/powerd.bundle/powerd
   24 ??         5:13.79 /usr/libexec/configd
   25 ??         1:01.05 /usr/sbin/syslogd
  ...
24703 ttys004    0:00.15 -bash
24743 ttys005    0:00.02 login -pfl markbates /bin/bash -c exec -la bash /bin...
24755 ttys005    0:00.21 -bash
57892 ttys005    0:00.04 ruby /Users/markbates/scripts/ss
57896 ttys005    0:00.42 forego start -p 3000
24810 ttys006    0:00.02 login -pfl markbates /bin/bash -c exec -la bash /bin...
24851 ttys006    0:00.70 -bash
25864 ttys007    0:00.01 login -pfl markbates /bin/bash -c exec -la bash /bin...
25865 ttys007    0:00.24 -bash
25866 ttys008    0:00.02 login -pfl markbates /bin/bash -c exec -la bash /bin...
25876 ttys008    0:00.10 -bash
```

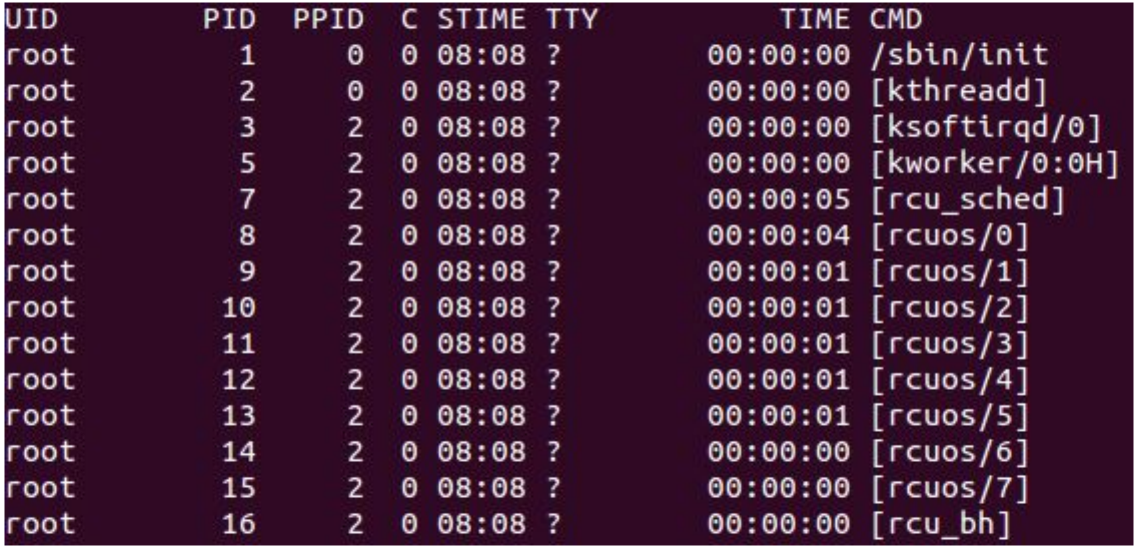
Display Processes by User (-U)

- filter processes by the user that is running them.
- ps -U root
- 

```
  PID TTY          TIME CMD
    1 ?        00:00:00 init
    2 ?        00:00:00 kthreadd
    3 ?        00:00:00 ksoftirqd/0
    5 ?        00:00:00 kworker/0:0H
    7 ?        00:00:05 rcu_sched
    8 ?        00:00:04 rcuos/0
    9 ?        00:00:01 rcuos/1
   10 ?        00:00:01 rcuos/2
   11 ?        00:00:01 rcuos/3
   12 ?        00:00:01 rcuos/4
   13 ?        00:00:01 rcuos/5
   14 ?        00:00:00 rcuos/6
   15 ?        00:00:00 rcuos/7
   16 ?        00:00:00 rcu_bh
   17 ?        00:00:00 rcuob/0
```

ps -ef

To find out information about *all* the processes in the system, the command is,

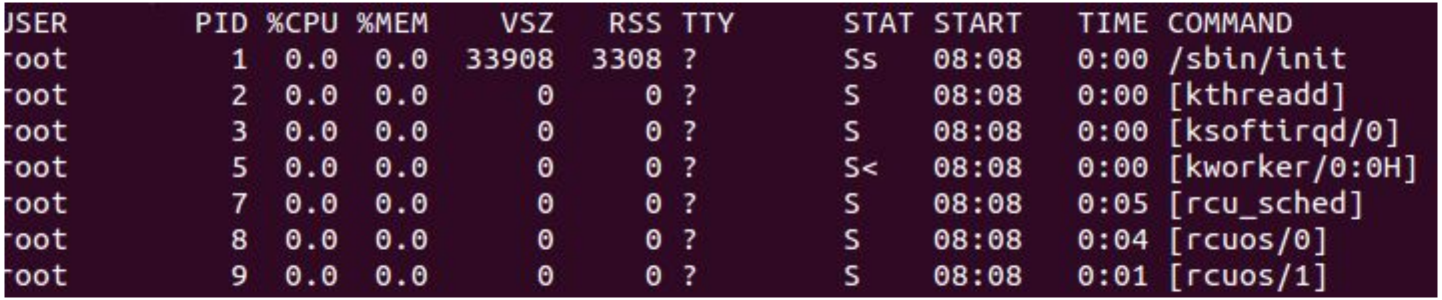


```
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 08:08 ?        00:00:00 /sbin/init
root         2     0  0 08:08 ?        00:00:00 [kthreadd]
root         3     2  0 08:08 ?        00:00:00 [ksoftirqd/0]
root         5     2  0 08:08 ?        00:00:00 [kworker/0:0H]
root         7     2  0 08:08 ?        00:00:05 [rcu_sched]
root         8     2  0 08:08 ?        00:00:04 [rcuos/0]
root         9     2  0 08:08 ?        00:00:01 [rcuos/1]
root        10     2  0 08:08 ?        00:00:01 [rcuos/2]
root        11     2  0 08:08 ?        00:00:01 [rcuos/3]
root        12     2  0 08:08 ?        00:00:01 [rcuos/4]
root        13     2  0 08:08 ?        00:00:01 [rcuos/5]
root        14     2  0 08:08 ?        00:00:00 [rcuos/6]
root        15     2  0 08:08 ?        00:00:00 [rcuos/7]
root        16     2  0 08:08 ?        00:00:00 [rcu_bh]
```

- The -e option selects all processes in the system.
- Without this, ps gives the processes belonging to the same effective user id
- The **-f option gives the full listing**.
- **UID -** gives the process owner's user-id
- PID - process-id
- PPID - parent process id

- **C -** represents CPU utilization**.**
- A high value for C indicates a CPU intensive process.
- **STIME-** process start time
- TTY - terminal from which the process has been started
- TIME -cumulative **CPU time** used by the process in [DD-]hh:mm:ss format
- CMD - command, using which, the process was created

- **ps aux**

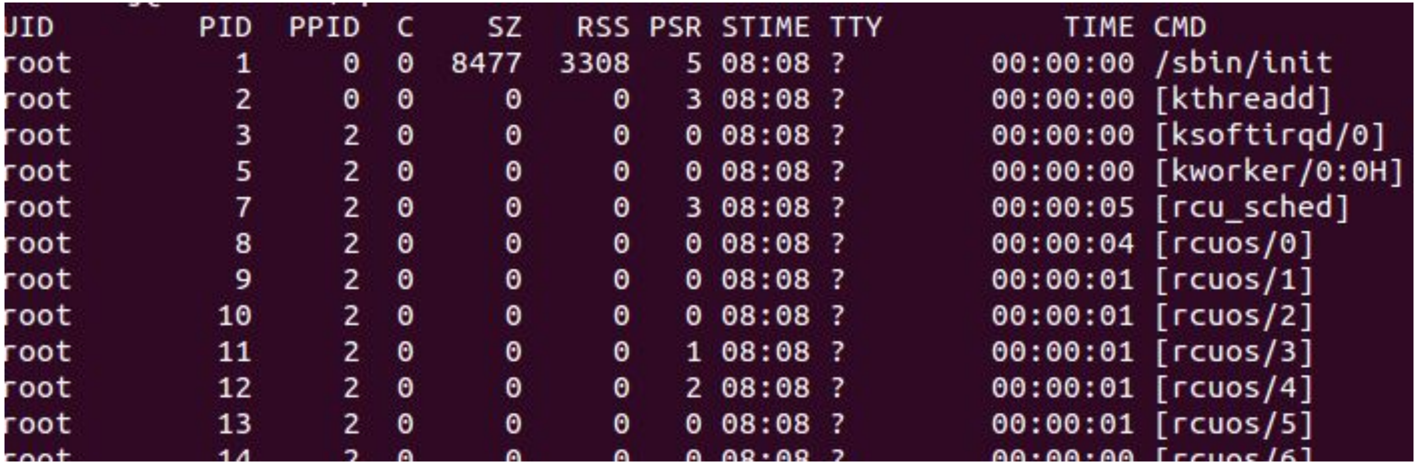


```
JSER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root          1  0.0  0.0  33908  3308 ?        Ss   08:08   0:00 /sbin/init
root          2  0.0  0.0      0     0 ?        S    08:08   0:00 [kthreadd]
root          3  0.0  0.0      0     0 ?        S    08:08   0:00 [ksoftirqd/0]
root          5  0.0  0.0      0     0 ?        S<   08:08   0:00 [kworker/0:0H]
root          7  0.0  0.0      0     0 ?        S    08:08   0:05 [rcu_sched]
root          8  0.0  0.0      0     0 ?        S    08:08   0:04 [rcuos/0]
root          9  0.0  0.0      0     0 ?        S    08:08   0:01 [rcuos/1]
```

- **USER** = user owning the process
- **PID** = process ID of the process
- **%CPU** = It is the CPU time used divided by the time the process has been running.
- **%MEM** = ratio of the process's resident set size to the physical memory on the machine
- **VSZ** = virtual memory usage of entire process (in KiB)
- **RSS** = resident set size, the non-swapped physical memory that a task has used (in KiB)
- **TTY** = controlling tty (terminal)
- **STAT** = multi-character process state
- **START** = starting time or date of the process
- **TIME** = cumulative CPU time
- **COMMAND** = command with all its arguments

ps -eF

-F option, which is the ***extra full* format**.



```
UID          PID  PPID  C    SZ   RSS PSR STIME TTY          TIME CMD
root           1     0  0  8477  3308   5 08:08 ?        00:00:00 /sbin/init
root           2     0  0     0     0   3 08:08 ?        00:00:00 [kthreadd]
root           3     2  0     0     0   0 08:08 ?        00:00:00 [ksoftirqd/0]
root           5     2  0     0     0   0 08:08 ?        00:00:00 [kworker/0:0H]
root           7     2  0     0     0   3 08:08 ?        00:00:05 [rcu_sched]
root           8     2  0     0     0   0 08:08 ?        00:00:04 [rcuos/0]
root           9     2  0     0     0   0 08:08 ?        00:00:01 [rcuos/1]
root          10     2  0     0     0   0 08:08 ?        00:00:01 [rcuos/2]
root          11     2  0     0     0   1 08:08 ?        00:00:01 [rcuos/3]
root          12     2  0     0     0   2 08:08 ?        00:00:01 [rcuos/4]
root          13     2  0     0     0   0 08:08 ?        00:00:01 [rcuos/5]
```

-F option gives three additional columns of output.

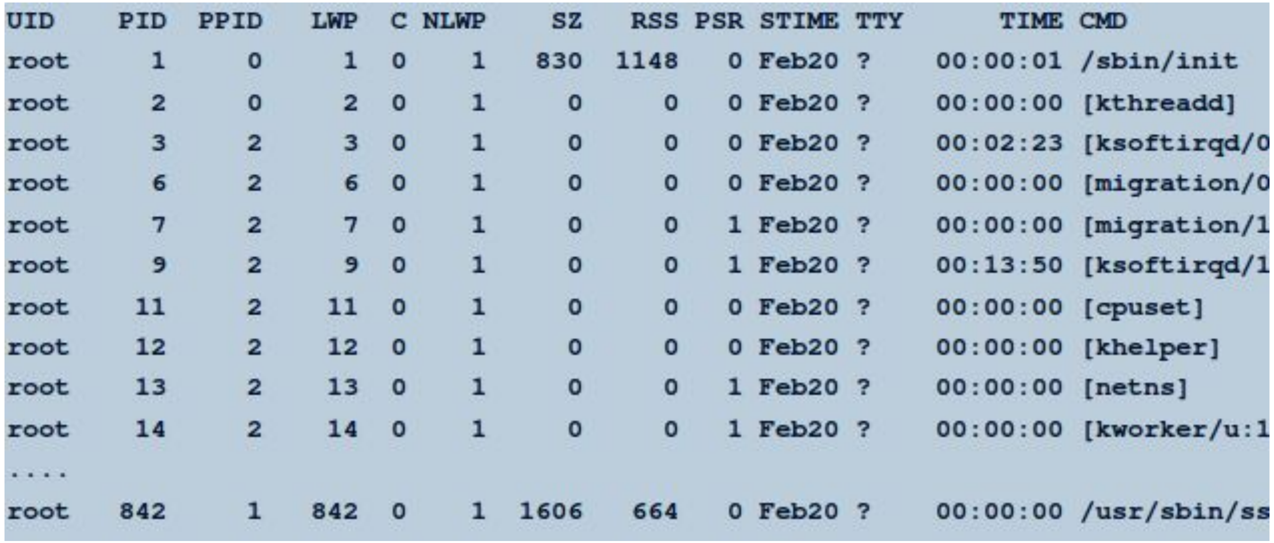
SZ column gives the **virtual size of the process**.

RSS is the **resident set size**, the **non-swapped physical memory that the process has used** in kilobytes.

PSR indicates the **processor that the process is currently assigned to**.

ps -eLF

The -L option is for getting information about the threads



```
UID      PID  PPID   LWP  C NLWP    SZ   RSS PSR STIME TTY          TIME CMD
root       1     0     1  0    1   830  1148   0 Feb20 ?        00:00:01 /sbin/init
root       2     0     2  0    1     0     0   0 Feb20 ?        00:00:00 [kthreadd]
root       3     2     3  0    1     0     0   0 Feb20 ?        00:02:23 [ksoftirqd/0
root       6     2     6  0    1     0     0   0 Feb20 ?        00:00:00 [migration/0
root       7     2     7  0    1     0     0   1 Feb20 ?        00:00:00 [migration/1
root       9     2     9  0    1     0     0   1 Feb20 ?        00:13:50 [ksoftirqd/1
root      11     2    11  0    1     0     0   0 Feb20 ?        00:00:00 [cpuset]
root      12     2    12  0    1     0     0   0 Feb20 ?        00:00:00 [khelper]
root      13     2    13  0    1     0     0   1 Feb20 ?        00:00:00 [netns]
root      14     2    14  0    1     0     0   1 Feb20 ?        00:00:00 [kworker/u:1
....
root     842     1   842  0    1  1606   664   0 Feb20 ?        00:00:00 /usr/sbin/ss
```

This gives one line of information per thread

LWP gives the **thread id of each thread**.

- NLWP is the n**umber of threads in the system** for the underlying process.
- The -L option gives an idea about the **total number of concurrent threads** in the system.

lsof

lsof stands for List Open Files.

- It is easy to remember lsof command if you think of it as **“ls + of”**, **where ls stands for list,** and **of stands for open files.**
- list the information about the files that are opened by various processes.
- In linux everything is file , using lsof you can get the information about any opened files.
- lsof

```
COMMAND  PID      USER   FD      TYPE     DEVICE  SIZE/OFF       NODE NAME

init       1       root  cwd       DIR       8,1      4096          2 /

init       1       root  txt      REG       8,1    124704     917562 /sbin/init

init       1       root    0u      CHR       1,3       0t0       4369 /dev/null

init       1       root    1u      CHR       1,3       0t0       4369 /dev/null

init       1       root    2u      CHR       1,3       0t0       4369 /dev/null

init       1       root    3r     FIFO       0,8       0t0       6323 pipe

...
```

- By default One file per line is displayed

FD – Represents the file descriptor. Some of the values of FDs are,

- cwd – Current Working Directory
- txt – Text file
- mem – Memory mapped file
- mmap – Memory mapped device
- NUMBER – Represent the actual file descriptor. The character after the number i.e ‘1u’, represents the mode in which the file is opened. r for read, w for write, u for read and write.

TYPE – Specifies the type of the file. Some of the values of TYPEs are,

- REG – Regular File
- DIR – Directory
- FIFO – First In First Out
- CHR – Character special file

List processes which opened a specific file

list only the processes which opened a specific file by providing the filename as arguments.

```
lsof /var/log/syslog
```

```
COMMAND  PID   USER   FD   TYPE DEVICE SIZE/OFF   NODE NAME
rsyslogd 488 syslog    1w   REG    8,1     1151 268940 /var/log/syslog
```

```
 FD is followed by one of these characters, describing the mode under which the file is open:
```

- `r for read access;`
- `w for write access;`
- `u for read and write access;`

List files opened by a specific user

```
lsof -u lakshmanan
```

```
COMMAND    PID       USER   FD   TYPE     DEVICE SIZE/OFF       NODE NAME
update-no 1892 lakshmanan   20r  FIFO        0,8      0t0      14536 pipe
update-no 1892 lakshmanan   21w  FIFO        0,8      0t0      14536 pipe
bash      1995 lakshmanan  cwd    DIR        8,1     4096     393218 /home/lakshmanan
```

list files opened by all users, expect some 1 or 2.

```
lsof -u ^lakshmanan
```

```
COMMAND    PID       USER   FD      TYPE     DEVICE  SIZE/OFF       NODE NAME
rtkit-dae 1380      rtkit    7u     0000        0,9         0       4360 anon_inode
udisks-da 1584       root  cwd       DIR        8,1      4096          2 /
```

List opened files under a directory

```
lsof +D /var/log/
```

```
COMMAND   PID   USER   FD   TYPE DEVICE SIZE/OFF   NODE NAME
rsyslogd  488 syslog   1w   REG    8,1     1151 268940 /var/log/syslog
rsyslogd  488 syslog   2w   REG    8,1     2405 269616 /var/log/auth.log
console-k 144   root   9w   REG    8,1    10871 269369 /var/log/ConsoleKit/history
```

+D will recurse the sub directories also. If you don’t want lsof to recurse, then use ‘+d’ option.

List opened files based on process names starting with

```
lsof -c ssh -c init
```

```
COMMAND    PID   USER   FD   TYPE DEVICE SIZE/OFF   NODE NAME
init         1       root  txt    REG        8,1   124704  917562 /sbin/init
init         1       root  mem    REG        8,1  1434180 1442625 /lib/i386-linux-gnu/libc-2.13.so
init         1       root  mem    REG        8,1    30684 1442694 /lib/i386-linux-gnu/librt-2.13.so
...
ssh-agent 1528 lakshmanan    1u   CHR        1,3      0t0    4369 /dev/null
ssh-agent 1528 lakshmanan    2u   CHR        1,3      0t0    4369 /dev/null
```

List all open files by a specific process

```
lsof -p 1753
```

```
COMMAND  PID       USER   FD   TYPE DEVICE SIZE/OFF    NODE NAME
bash    1753 lakshmanan  cwd    DIR    8,1     4096  393571 /home/lakshmanan/test.txt
bash    1753 lakshmanan  rtd    DIR    8,1     4096       2 /
bash    1753 lakshmanan  255u   CHR  136,0      0t0       3 /dev/pts/0
...
```

ss

- used to show socket statistics
- It allows showing information similar to netstat command.
- The ss command gets its information directly from kernel space.
- The **ss command** is fast. Information is very fast when you are searching.
- gain more useful information about TCP and state information with the **ss command**.

ss command can provide information about:

- All TCP sockets.
- All UDP sockets.
- All established ssh / ftp / http / https connections.
- Filtering by state (such as connected, synchronized), addresses and ports.

List all connection

ss | more

```
Netid  State      Recv-Q Send-Q   Local Address:Port                       Peer Address:Port
u_str  ESTAB      0              0                   * 219827                             * 220601
u_str  ESTAB      0             0     @/tmp/.X11-unix/X0 223495                           * 223494
u_str  ESTAB      0             0                  * 220663                               * 220044
u_str  ESTAB      0             0     @/tmp/.X11-unix/X0 220453                           *219708
u_str  ESTAB      0             0     @/tmp/dbus-elWbPTCWlq 223497                       * 223496
............................
tcp    ESTAB     0             0        192.168.1.2:33643                                 68.232.35.139:https
tcp    ESTAB     0             0        192.168.1.2:56529                                 54.236.180.90:9999
```

- **Netid**, it denotes socket type and transport protocol
- t**cp, udp, raw, u_str** is abbreviation for **unix_stream**, **u_dgr** for **UNIX datagram sockets**, **nl** for **netlink**, **p_raw** and **p_dgr** for **raw and datagram packet sockets**.
- **State**-Socket state ,the names are standard TCP names, except for UNCONN, which cannot happen for TCP, but normal for not connected sockets of another types.
- **Recv-Q and Send-Q** showing amount of data queued for receive and transmit.
- last two columns display **local address and port of the socket** and **its peer address**, if the socket is connected.

List all TCP connection

To list all TCP connection use -t option:

ss -t

```
State     Recv-Q Send-Q        Local Address:Port        Peer Address:Port

ESTAB          0        0        172.17.7.47:38272        172.17.5.2:ssh

ESTAB          0        0        172.17.7.47:57092        74.125.204.189:https

ESTAB          0        0        172.17.7.47:38091        204.11.109.67:http

ESTAB          0        0        172.17.7.47:53200        74.125.130.109:imaps
```

“**-t**” options report only “ESTABLISHED” or “CONNECTED” connections. If use the “**-ta**” option report show all TCP connections (connected and listening).

ss -ta

```
State      Recv-Q Send-Q   Local Address:Port          Peer Address:Port

LISTEN          0       128                *:ssh               *:*

LISTEN          0       128      127.0.0.1:ipp             *:*

LISTEN          0       50       127.0.0.1:mysql           *:*

ESTAB           0       0        172.17.7.47:53700         216.58.197.34:https

ESTAB           0       0        172.17.7.47:38272         172.17.5.2:ssh
```

List all UDP connection

```
ss -ua
```

```
State      Recv-Q Send-Q   Local Address:Port           Peer Address:Port

UNCONN          0       0                   *:ipp                 *:*

UNCONN          0       0                   *:mdns                *:*

UNCONN          0       0                   *:mdns                *:*

UNCONN          0       0                   *:35057               *:*

UNCONN          0       0                   :::46110              :::*

UNCONN          0       0                   :::mdns               :::*
```

Do not resolve hostname

- use the "n" option to prevent ss from resolving ip addresses to hostnames.

ss -ua

```
●   State    Recv-Q Send-Q        Local Address:Port     Peer Address:Port
●   UNCONN       0        0                *:ipp                    *:*
●   UNCONN       0        0                *:mdns                   *:*
●   UNCONN       0        0                *:mdns                   *:*
●   UNCONN       0        0                *:35057                  *:*
●   UNCONN       0        0                :::46110                 :::*
●   UNCONN       0        0                :::mdns                  :::*
```

ss -nua

```
●   State    Recv-Q Send-Q        Local Address:Port     Peer Address:Port
●   UNCONN       0        0                *:631                    *:*
●   UNCONN       0        0                *:5353                   *:*
●   UNCONN       0        0                *:5353                   *:*
●   UNCONN       0        0                *:35057                  *:*
●   UNCONN       0        0                :::46110                 :::*
●   UNCONN       0        0                :::5353                  :::*
```

Show only listening sockets

list out all the listening sockets. For example apache web server opens a socket connection on port 80 to listen for incoming connections.

ss -ltn (listening tcp connections)

```
●    State   Recv-Q Send-Q       Local Address:Port     Peer Address:Port
●    LISTEN       0      128                   *:22                   *:*
●    LISTEN       0      128           127.0.0.1:631                  *:*
●    LISTEN       0      50            127.0.0.1:3306                 *:*
●    LISTEN       0      128           :::22               :::*
●    LISTEN       0      128           ::1:631             :::*
```

ss -lun (listening udp connections)

```
●    State   Recv-Q Send-Q       Local Address:Port     Peer Address:Port
●    UNCONN       0      0             *:631               *:*
●    UNCONN       0      0             *:5353              *:*
●    UNCONN       0      0             *:5353              *:*
●    UNCONN       0      0             *:35057             *:*
●    UNCONN       0      0             :::46110            :::*
●    UNCONN       0      0             :::5353             :::*
```

Print process name and pid

To print out the process name/pid which owns the connection use the p option

```
ss -ltp
State      Recv-Q Send-Q    Local Address:Port        Peer Address:Port
LISTEN     0      100          127.0.0.1:smtp               *:*
LISTEN     0      128          127.0.0.1:9050               *:*
LISTEN     0      128                  *:90                 *:*
LISTEN     0      128                  *:db-lsp            *:*       users:(("dropbox",3566,32))
LISTEN     0      5           127.0.0.1:6600                *:*
LISTEN     0      128          127.0.0.1:9000               *:*       users:(("php5-fpm",1620,0),("php5-fpm",1619,0))
```

Print summary statistics

The s option prints out the statistics.

```
 ss -s
Total: 913 (kernel 0)
TCP:   31 (estab 24, closed 0, orphaned 0, synrecv 0, timewait 0/0), ports 0
Transport Total          IP        IPv6
*      0            -         -
RAW      0      0         0
UDP      6      4         2
TCP      31     28        3
INET      37    32        5
FRAG      0     0         0
```

Display timer information

With the '-o' option, the time information of each connection would be displayed.

```
$ ss -tn -o
State      Recv-Q Send-Q      Local Address:Port        Peer Address:Port
ESTAB      0      0           192.168.1.2:43839     108.160.162.37:80
ESTAB      0      0           192.168.1.2:36335     204.144.140.26:80     timer:(keepalive,26sec,0)
ESTAB      0      0           192.168.1.2:33141      83.170.73.249:6667
ESTAB      0      0           192.168.1.2:58857      74.121.141.84:80     timer:(keepalive,23sec,0)
ESTAB      0      0           192.168.1.2:42794     173.194.40.239:80     timer:(keepalive,32sec,0)
```

Display only IPv4 or IPv6 socket connections

ss -tl4

```
State    Recv-Q Send-Q         Local Address:Port     Peer Address:Port
LISTEN         0         128               *:ssh                      *:*
LISTEN         0         128               127.0.0.1:ipp                        *:*
LISTEN         0         50                127.0.0.1:mysql                      *:*
```

ss -tl6

```
State    Recv-Q Send-Q         Local Address:Port     Peer Address:Port
LISTEN         0         128               :::ssh                     :::*
LISTEN         0         128               ::1:ipp                    :::*
```

Filtering connections by tcp state

ss command supports filters that can be use to display only specific connections.

Display sockets with state time-wait

```
ss -t4 state time-wait
Recv-Q          Send-Q        Local Address:Port            Peer Address:Port
0                  0             192.168.1.2:42261          199.59.150.39:https
0                  0               127.0.0.1:43541              127.0.0.1:2633
```

Filter connections by address and port number

Filter by destination port

```
ss -nt dst :443 or dst :80
```

Filter by destination address

```
ss -nt dst 74.125.236.178
```

Address and Port combined

```
$ ss -nt dst 74.125.236.178:80
```

Iotop

used to monitor the Disk I/O

- Iotop is an open source and free utility
- used to monitor the Disk I/O and to trace the exact process or high used user disk read/writes.
- Iotop tool is based on Python programming
- Need sudo privilege to run iotop

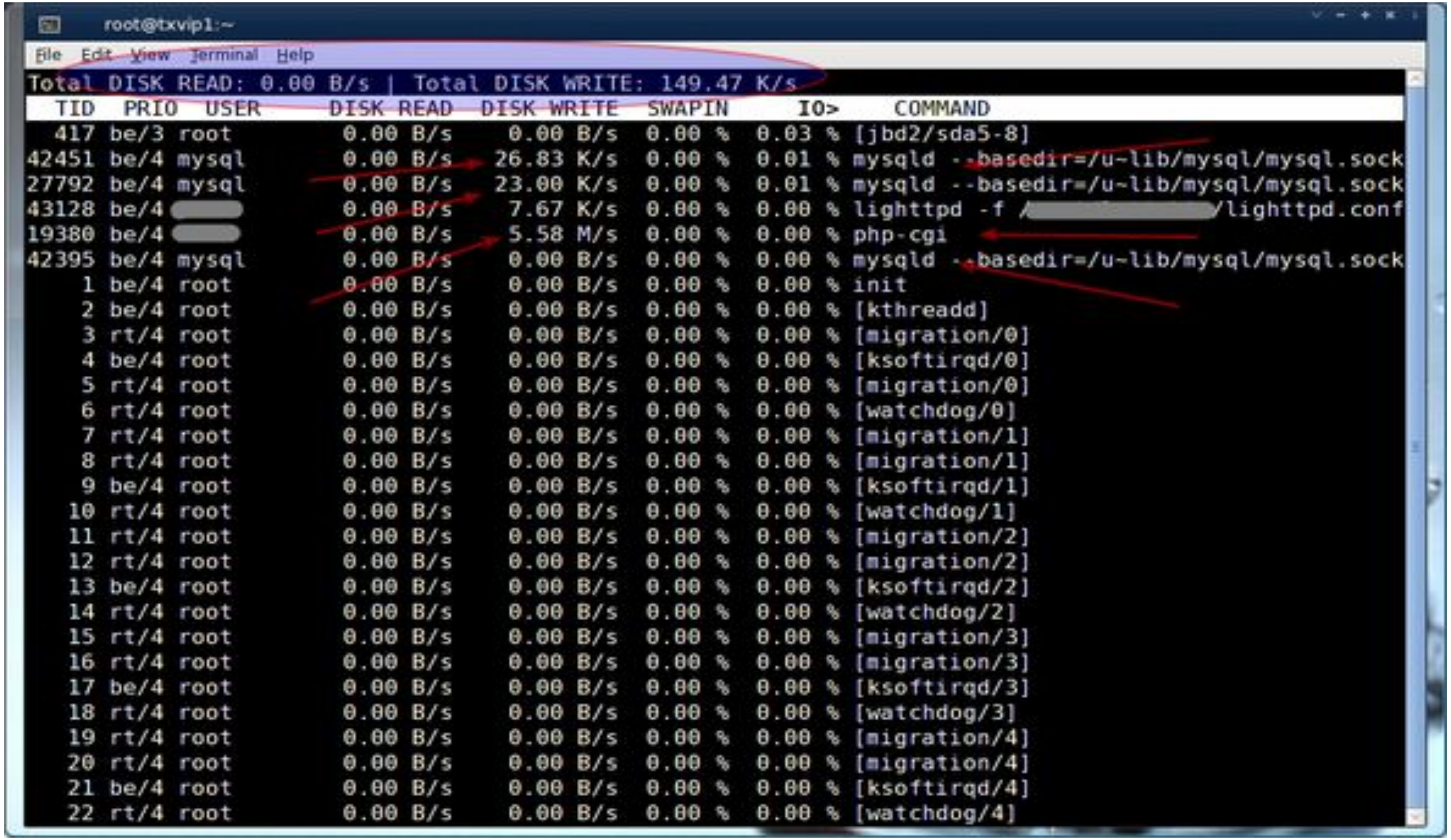
Install iotop

Centos ,redhat

```
yum install iotop
```

Debian ,ubuntu

```
sudo apt-get install iotop
```

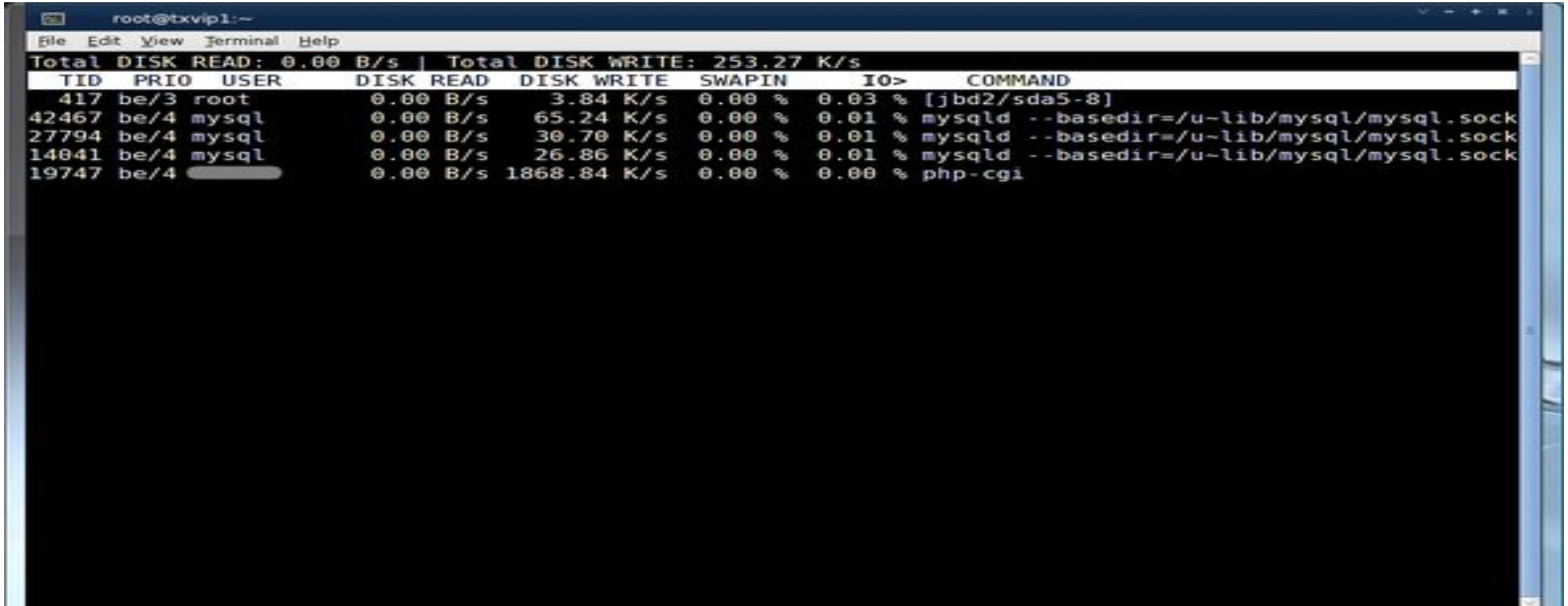


```
Total DISK READ: 0.00 B/s | Total DISK WRITE: 149.47 K/s
  TID  PRIO  USER     DISK READ  DISK WRITE  SWAPIN     IO>    COMMAND
  417 be/3 root        0.00 B/s    0.00 B/s  0.00 %  0.03 % [jbd2/sda5-8]
42451 be/4 mysql       0.00 B/s   26.83 K/s  0.00 %  0.01 % mysqld --basedir=/u-lib/mysql/mysql.sock
27792 be/4 mysql       0.00 B/s   23.00 K/s  0.00 %  0.01 % mysqld --basedir=/u-lib/mysql/mysql.sock
43128 be/4 [redacted]  0.00 B/s    7.67 K/s  0.00 %  0.00 % lighttpd -f /[redacted]/lighttpd.conf
19380 be/4 [redacted]  0.00 B/s    5.58 M/s  0.00 %  0.00 % php-cgi
42395 be/4 mysql       0.00 B/s    0.00 B/s  0.00 %  0.00 % mysqld --basedir=/u-lib/mysql/mysql.sock
    1 be/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % init
    2 be/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [kthreadd]
    3 rt/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [migration/0]
    4 be/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [ksoftirqd/0]
    5 rt/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [migration/0]
    6 rt/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [watchdog/0]
    7 rt/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [migration/1]
    8 rt/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [migration/1]
    9 be/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [ksoftirqd/1]
   10 rt/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [watchdog/1]
   11 rt/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [migration/2]
   12 rt/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [migration/2]
   13 be/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [ksoftirqd/2]
   14 rt/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [watchdog/2]
   15 rt/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [migration/3]
   16 rt/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [migration/3]
   17 be/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [ksoftirqd/3]
   18 rt/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [watchdog/3]
   19 rt/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [migration/4]
   20 rt/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [migration/4]
   21 be/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [ksoftirqd/4]
   22 rt/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [watchdog/4]
```

- The "IO" column lists total I/O for each process
- The "SwapIn" column lists **swap activity for each process**.
- TID stands for Thread ID

iotop --only

to see only **processes or threads actually doing I/O**, instead of showing all processes or threads



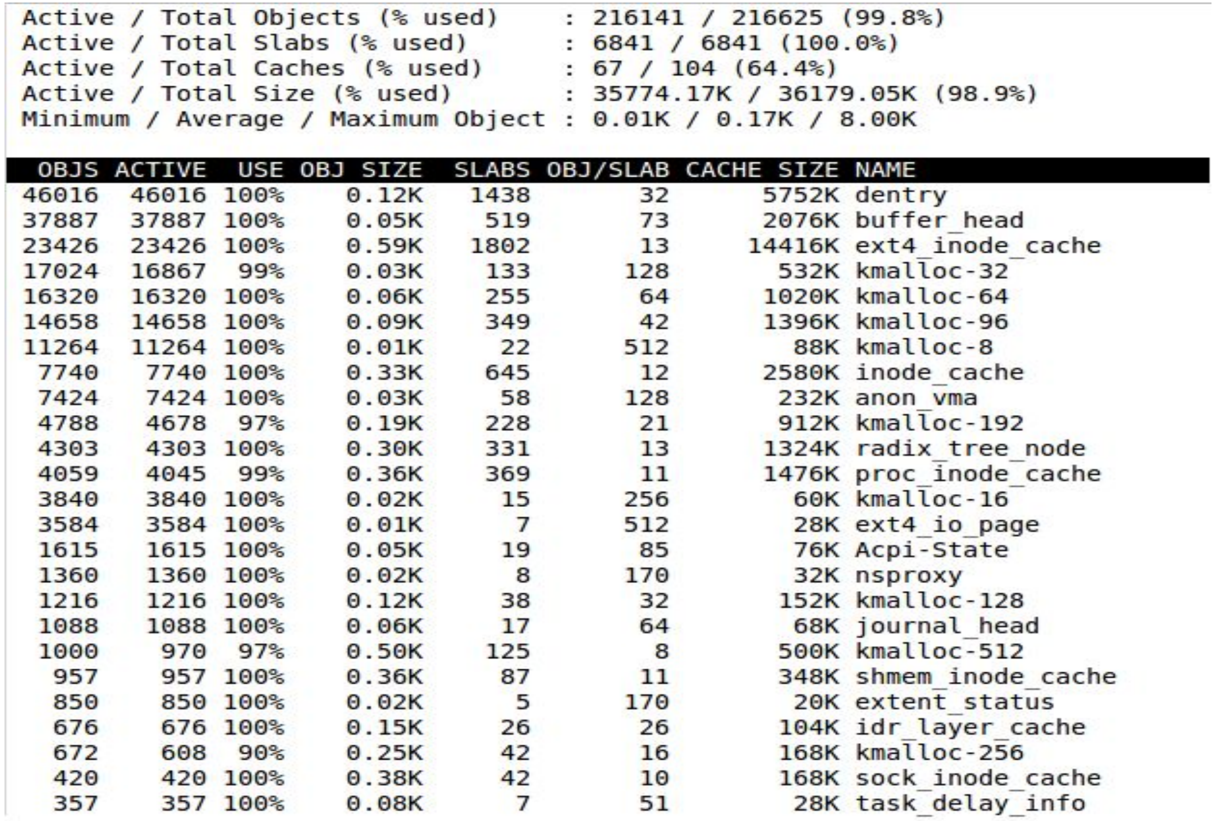
root@txvip1:~

File Edit View Terminal Help

```
Total DISK READ: 0.00 B/s | Total DISK WRITE: 253.27 K/s
  TID  PRIO  USER     DISK READ  DISK WRITE  SWAPIN     IO>    COMMAND
  417 be/3 root        0.00 B/s    3.84 K/s  0.00 %  0.03 % [jbd2/sda5-8]
42467 be/4 mysql       0.00 B/s   65.24 K/s  0.00 %  0.01 % mysqld --basedir=/u-lib/mysql/mysql.sock
27794 be/4 mysql       0.00 B/s   30.70 K/s  0.00 %  0.01 % mysqld --basedir=/u-lib/mysql/mysql.sock
14041 be/4 mysql       0.00 B/s   26.86 K/s  0.00 %  0.01 % mysqld --basedir=/u-lib/mysql/mysql.sock
19747 be/4 [redacted]  0.00 B/s 1868.84 K/s  0.00 %  0.00 % php-cgi
```

1. Hit the **left** and **right** arrow keys to change the sorting.
2. Hit **r** to reverse the sorting order.
3. Hit **o** only to see processes or threads actually doing I/O, instead of showing all processes or threads.
4. Hit **p** only show processes. Normally iotop shows all threads.

slabtop

- slab - **memory management mechanism** intended for the **efficient memory allocation of kernel objects**
- slabtop command displays detailed **kernel slab cache information** in real time
- 

```
 Active / Total Objects (% used)    : 216141 / 216625 (99.8%)
 Active / Total Slabs (% used)      : 6841 / 6841 (100.0%)
 Active / Total Caches (% used)     : 67 / 104 (64.4%)
 Active / Total Size (% used)       : 35774.17K / 36179.05K (98.9%)
 Minimum / Average / Maximum Object : 0.01K / 0.17K / 8.00K

  OBJS ACTIVE  USE OBJ SIZE  SLABS OBJ/SLAB CACHE SIZE NAME
 46016  46016 100%    0.12K   1438       32      5752K dentry
 37887  37887 100%    0.05K    519       73      2076K buffer_head
 23426  23426 100%    0.59K   1802       13     14416K ext4_inode_cache
 17024  16867  99%    0.03K    133      128       532K kmalloc-32
 16320  16320 100%    0.06K    255       64      1020K kmalloc-64
 14658  14658 100%    0.09K    349       42      1396K kmalloc-96
 11264  11264 100%    0.01K     22      512        88K kmalloc-8
  7740   7740 100%    0.33K    645       12      2580K inode_cache
  7424   7424 100%    0.03K     58      128       232K anon_vma
  4788   4678  97%    0.19K    228       21       912K kmalloc-192
  4303   4303 100%    0.30K    331       13      1324K radix_tree_node
  4059   4045  99%    0.36K    369       11      1476K proc_inode_cache
  3840   3840 100%    0.02K     15      256        60K kmalloc-16
  3584   3584 100%    0.01K      7      512        28K ext4_io_page
  1615   1615 100%    0.05K     19       85        76K Acpi-State
  1360   1360 100%    0.02K      8      170        32K nsproxy
  1216   1216 100%    0.12K     38       32       152K kmalloc-128
  1088   1088 100%    0.06K     17       64        68K journal_head
  1000    970  97%    0.50K    125        8       500K kmalloc-512
   957    957 100%    0.36K     87       11       348K shmem_inode_cache
   850    850 100%    0.02K      5      170        20K extent_status
   676    676 100%    0.15K     26       26       104K idr_layer_cache
   672    608  90%    0.25K     42       16       168K kmalloc-256
   420    420 100%    0.38K     42       10       168K sock_inode_cache
   357    357 100%    0.08K      7       51        28K task_delay_info
```

vmstat (*virtual memory statistics*)

- vmstat is a tool in Unix/Linux which is used to **Report virtual memory statistics**.
- It shows how much virtual memory there is
- how much is free and paging activity.
- page-ins and page-outs as they happen.

```
vmstat

procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----

 r  b   swpd   free   buff      cache          si       so         bi        bo   in   cs  us  sy id  wa  st

 2  0    0    509288 218380 3525292        0         0         17        48   60  259  2  0  98  1   0
```

- **r**: The number of processes waiting for run time
- **b**: The number of processes in uninterruptible sleep. (b=blocked queue, waiting for resource
- **swpd**: shows how **many blocks are swapped out to disk** (paged). The **amount of Virtual memory used.**
- **free**: The amount of Idle Memory
- **buff**: Memory used as buffers, like before/after I/O operations
- **cache**: Memory used as cache by the Operating System

si: Amount of memory swapped in from disk (/s). This shows page-ins

so: Amount of memory swapped to disk (/s). This shows page-outs

Under IO we have:

bi: Blocks received from block device - Read

bo: Blocks sent to a block device – Write

Under System we have:

in: The number of interrupts per second,

cs: The number of context switches per second

Under CPU we have:

These are **percentages of total CPU time**.

us: % of CPU time spent in user mode

sy: % of CPU time spent running kernel code. (system time)

id: % of CPU idle time

wa: % of CPU time spent waiting for IO.

/proc/meminfo

Active — The total amount of buffer or page cache memory, in kilobytes, that is in active use. This is memory that has been recently used and is usually **not reclaimed** for other purposes.

Inactive — The total amount of buffer or page cache memory, in kilobytes, that are free and available. This is memory that has no been recently used and can be **reclaimed for other purposes**.

Dirty — The **total amount of memory**, in kilobytes, **waiting to be written back to the disk**.

Slab — The total amount of memory, in kilobytes, used by the kernel to cache data structures for its own use.

- **memory management mechanism** intended for the **efficient memory allocation of kernel objects**

SwapCached — The amount of swap, in kilobytes, used as cache memory.

Buffer - data buffer (or just buffer) is a region of a physical memory storage used to temporarily store data while it is being moved from one place to another.
The data is stored in a buffer as it is retrieved from an input device (such as a microphone) or just before it is sent to an output device (such as speakers).

Cache-to reduce the average time to access data from the main memory. The cache is a smaller, faster memory which stores copies of the data from frequently used main memory locations. Most CPUs have different independent caches, including instruction and data caches, where the data cache is usually organized as a hierarchy of more cache levels (L1, L2, etc.).

When the processor needs to read from or write to a location in main memory, it first checks whether a copy of that data is in the cache. If so, the processor immediately reads from or writes to the cache, which is much faster than reading from or writing to main memory.

instruction cache to speed up executable instruction fetch

data cache to speed up data fetch and store,

translation lookaside buffer (TLB) used to speed up virtual-to-physical address translation for both executable instructions and data

/proc/partitions

This **file contains partition block allocation information.**

Block devices are disk devices for which the kernel provides caching.

```
major minor        #blocks         name
  3    0           19531250        hda
  3   1            104391           hda1
  3   2            19422585        hda2
253         0           22708224        dm-0
253    1           524288          dm-1
```

major – **The major number of the device with this partition**. The major number in the /proc/partitions, (3), corresponds with the block device ide0, in /proc/devices.

minor — **The minor number of the device with this partition**. This serves to separate the partitions into different physical devices and relates to the number at the end of the name of the partition.

#blocks — **Lists the number of physical disk blocks contained in a particular partition**.

name — **The name of the partition**.

/proc/devices

This file **displays the various character and block devices currently configured**

```
Character devices:
  1 mem
  4 /dev/vc/0
  4 tty
  4 ttyS
  5 /dev/tty
  5 /dev/console
  5 /dev/ptmx
  7 vcs
  10 misc
  13 input
  29 fb
  36 netlink
  128 ptm
  136 pts
```

```
180 usb

Block devices:
  1 ramdisk
  3 ide0
  9 md
  22 ide1
  253 device-mapper
  254 mdp
```

The **output from /proc/devices includes the major number and name of the device**, and is broken into two major sections:**Character devices and Block devices.**

A Character ('c') Device is one with which the **Driver communicates by sending and receiving single characters** (bytes, octets). A **Block ('b')Device** is one with which the **Driver communicates by sending entire blocks of data**.

Clearing swap space

free

```
total       used       free     shared    buffers     cached
Mem:       7987492    7298164     689328          0      30416     457936
-/+ buffers/cache:    6809812    1177680
Swap:      5963772     609452    5354320
```

Use swapoff -a command

```
free
```

```
total         used          free       shared    buffers      cached

Mem:          7987492     7777912      209580           0       39332      489864

-/+ buffers/cache:     7248716      738776

Swap:                0           0           0
```

And to re-enable it:

```
$ swapon -a
```

Context switch

A **context switch** occurs when a computer's CPU switches from one process or thread to a different process or thread. Context switching allows for one CPU to handle numerous processes or threads without the need for additional processors.

Interrupt

interrupt is a signal to the processor emitted by hardware or software indicating an event that needs immediate attention.

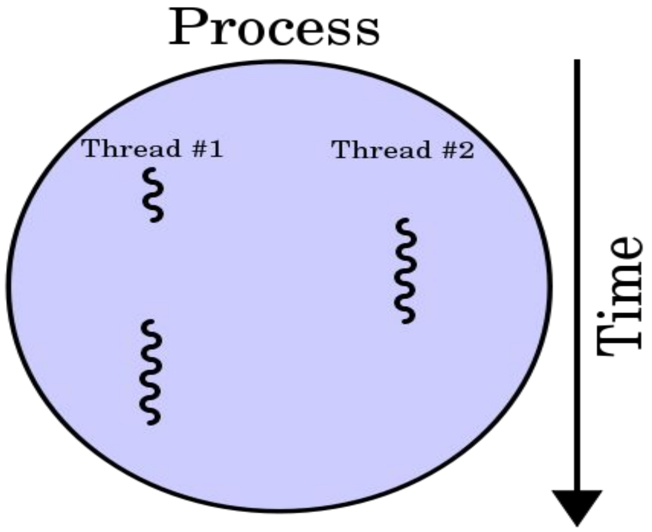
An interrupt alerts the processor to a high-priority condition requiring the interruption of the current code the processor is executing.

Hardware interrupts are used by devices to communicate that they require attention from the operating system.

A **software interrupt** is caused either by an exceptional condition in the processor itself, or a special instruction in the instruction set which

Thread

thread of execution is the smallest sequence of programmed instructions that can be managed independently,which is typically a part of the operating system.



A process with two threads of execution, running on one processor.

Clearing cache in linux

To free pagecache:

# **echo 1 > /proc/sys/vm/drop_caches**

To free dentries and inodes:

# **echo 2 > /proc/sys/vm/drop_caches**

To free pagecache, dentries and inodes:

# **echo 3 > /proc/sys/vm/drop_caches**

An **inode i**n your context is a **data structure that represents a file**. A **dentries i**s a **data structure that represents a directory**. These structures could be **used to build a memory cache that represents the file structure on a disk**. To get a directory listing, the OS could go to the dentries--if the directory is there--list its contents (a series of inodes). If not there, go to the disk and read it into memory so that it can be used again.

Page Cache accelerates many accesses to files.

This happens because, when it first reads from or writes to data media like hard drives, Linux also stores data in **unused areas of memory**, which acts as a cache. If this data is read again later, it can be quickly read from this cache in memory

Writing

If data is written, it is first written to the Page Cache and managed as one of its ***dirty pages***. *Dirty* means that the data is stored in the Page Cache, but needs to be written to the underlying storage device first. The content of these *dirty pages* is periodically transferred to the underlying storage device.

Reading

File blocks are written to the Page Cache not just during writing, but also when reading files. For example, when you read a 100-megabyte file twice, once after the other, the second access will be quicker, because the file blocks come directly from the Page Cache in memory and do not have to be read from the hard disk again.