EXP NO: 1

DATE: 9/7/25

Basic commands in Windows or Linux OS

AIM:

To Study of various Network commands used in Linux and Windows

COMMANDS:

Exploring the IP configuration of your Linux host with the "ip" command!

So there you are, a network engineer sitting at the console of a Linux workstation, and you need to explore or change the network configuration. Let's walk through a bit of "networking 101" with the "ip" command.

First up, let's see what happens when we just run "ip."

```
(main) expert@expert-cws:~$ ip
Usage: ip [ OPTIONS ] OBJECT { COMMAND | help }
       ip [ -force ] -batch filename
where  OBJECT := { link | address | addrlabel | route | rule | neigh | ntable |
                   tunnel | tuntap | maddress | mroute | mrule | monitor | xfrm |
                   netns | l2tp | fou | macsec | tcp_metrics | token | netconf | ila |
                   vrf | sr | nexthop }
       OPTIONS := { -V[ersion] | -s[tatistics] | -d[etails] | -r[esolve] |
                    -h[uman-readable] | -iec | -j[son] | -p[retty] |
                    -f[amily] { inet | inet6 | mpls | bridge | link } |
                    -4 | -6 | -I | -D | -M | -B | -0 |
                    -l[oops] { maximum-addr-flush-attempts } | -br[ief] |
                    -o[neline] | -t[imestamp] | -ts[hort] | -b[atch] [filename] |
                    -rc[vbuf] [size] | -n[etns] name | -N[umeric] | -a[ll] |
                    -c[olor]}
```

There's some interesting info just in this help/usage message. It looks like "ip" requires an OBJECT on which a COMMAND is executed. And the possible objects include several that jump out at the network engineer inside of me.

- link – I'm curious what "link" means in this context, but it catches my eye for sure
- address – This is really promising. The ip "addresses" assigned to a host is high on the list of things I know I'll want to understand.
- route – I wasn't fully expecting "route" to be listed here if I'm thinking in terms of the "ipconfig" or "ifconfig" command. But the routes configured on a host is something I'll be interested in.
- neigh – Neighbors? What kind of neighbors?
- tunnel – Oooo… tunnel interfaces are definitely interesting to see here.

- maddress, mroute, mrule – My initial thought when I saw "maddress" was "MAC address," but then I looked at the next two objects and thought maybe it's "multicast address." We'll leave "multicast" for another blog post. ?

The other objects in the list are interesting to see. Having "netconf" in the list was a happy surprise for me. But for this blog post, we'll stick with the basic objects of link, address, route, and neigh.

Where in the network are we? Exploring "ip address"

First up in our exploration will be the "ip address" object. Rather than just go through the full command help or man page line (ensuring *no one ever* reads another post of mine), I'm going to look at some common things I might want to know about the network configuration on a host. As you are exploring on your own, I would highly recommend exploring "ip address help" as well as "man ip address" for more details. These commands are very powerful and flexible.
What is my IP address?

```
(main) expert@expert-cws:~$ ip address show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group
default qlen 1000
    link/ether 00:0c:29:75:99:27 brd ff:ff:ff:ff:ff:ff
    inet 172.16.211.128/24 brd 172.16.211.255 scope global dynamic ens160
       valid_lft 1344sec preferred_lft 1344sec
    inet6 fe80::20c:29ff:fe75:9927/64 scope link
       valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state
DOWN group default
    link/ether 02:42:9a:0c:8a:ee brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
```

Running "ip address show" will display the address configuration for all interfaces on the Linux workstation. My workstation has 3 interfaces configured, a loopback address, the ethernet interface, and docker interface. Some of the Linux hosts I work on have dozens of interfaces, particularly if the host happens to be running lots of Docker containers as each container generates network interfaces. I plan to dive into Docker networking in future blog posts, so we'll leave the "docker0" interface alone for now.

We can focus our exploration by providing a specific network device name as part of our command.

```
(main) expert@expert-cws:~$ ip add show dev ens160
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group
default qlen 1000
   link/ether 00:0c:29:75:99:27 brd ff:ff:ff:ff:ff:ff
   inet 172.16.211.128/24 brd 172.16.211.255 scope global dynamic ens160
     valid_lft 1740sec preferred_lft 1740sec
   inet6 fe80::20c:29ff:fe75:9927/64 scope link
     valid_lft forever preferred_lft forever
```

Okay, that's really what I was interested in looking at when I wanted to know what my IP address was. But there is a lot more info in that output than just the IP address. For a long time, I just skimmed over the output. I would ignore most output and simply look at the address and for state info like "UP" or "DOWN." Eventually, I wanted to know what all that output meant, so in case you're interested in how to decode the output above…

- Physical interface details
  - "ens160" – The name of the interface from the operating system's perspective. This depends a lot on the specific distribution of Linux you are running, whether it is a virtual or physical machine, and the type of interface. If you're more used to seeing "eth0" interface names (like I was) it is time to become comfortable with the new interface naming scheme.
  - "<BROADCAST,MULTICAST,UP,LOWER_UP>" – Between the angle brackets are a series of flags that provide details about the interface state. This shows that my interface is both broadcast and multicast capable and that the interface is enabled (UP) and that the physical layer is connected (LOWER_UP)
  - "mtu 1500" – The maximum transmission unit (MTU) for the interface. This interface is configured for the default 1500 bytes
  - "qdisc mq" – This indicates the queueing approach being used by the interface. Things to look for here are values of "noqueue" (send immediately) or "noop" (drop all). There are several other options for queuing a system might be running.
  - "state UP"- Another indication of the operational state of an interface. "UP" and "DOWN" are pretty clear, but you might also see "UNKNOWN" like in the loopback interface above. "UNKNOWN" indicates that the interface is up and operational, but nothing is connected. Which is pretty valid for a loopback address.
  - "group default" – Interfaces can be grouped together on Linux to allow common attributes or commands. Having all interfaces connected to "group default" is the most common setup, but there are some handy things you can do if you group interfaces together. For example, imagine a VM host system with 2 interfaces for management and 8 for data traffic. You could group them into "mgmt" and "data" groups and then control all interfaces of a type together.
  - "qlen 1000" – The interface has a 1000 packet queue. The 1001st packet would be dropped.
- "link/ether" – The layer 2 address (MAC address) of the interface

- “inet” – The IPv4 interface configuration
  - “scope global” – This address is globally reachable. Other options include link and host
  - “dynamic” – This IP address was assigned by DHCP. The lease length is listed in the next line under “valid_lft”
  - “ens160” – A reference back to the interface this IP address is associated with
- “inet6” – The IPv6 interface configuration. Only the link local address is configured on the host. This shows that while IPv6 is enabled, the network doesn’t look to have it configured more widely

Network engineers link the world together one device at a time. Exploring the “ip link” command.

Now that we’ve gotten our feet wet, let’s circle back to the “link” object. The output of “ip address show” command gave a bit of a hint at what “link” is referring to. “Links” are the network devices configured on a host, and the “ip link” command provides engineers options for exploring and managing these devices.

What networking interfaces are configured on my host?

```
(main) expert@expert-cws:~$ ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode
DEFAULT group default qlen 1000
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode
DEFAULT group default qlen 1000
   link/ether 00:0c:29:75:99:27 brd ff:ff:ff:ff:ff:ff
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state
DOWN mode DEFAULT group default
   link/ether 02:42:9a:0c:8a:ee brd ff:ff:ff:ff:ff:ff
```

After exploring the output of “ip address show,” it shouldn’t come as a surprise that there are 3 network interfaces/devices configured on my host. And a quick look will show the output from this command is all included in the output for “ip address show.” For this reason, I almost always just use “ip address show” when looking to explore the network state of a host.

However, the “ip link” object is quite useful when you are looking to configure new interfaces on a host or change the configuration on an existing interface. For example, “ip link set” can change the MTU on an interface.

```
root@expert-cws:~# ip link set ens160 mtu 9000

root@expert-cws:~# ip link show dev ens160
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc mq state UP mode
DEFAULT group default qlen 1000
   link/ether 00:0c:29:75:99:27 brd ff:ff:ff:ff:ff:ff
```

Note 1: Changing network configuration settings requires administrative or “root” privileges.

Note 2: The changes made using the "set" command on an object are typically NOT maintained across system or service restarts. This is the equivalent of changing the "running-configuration" of a network device. In order to change the "startup-configuration" you need to edit the network configuration files for the Linux host. Check the details for network configuration for your distribution of Linux (ie Ubuntu, RedHat, Debian, Raspbian, etc.)

Is anyone else out there? Exploring the "ip neigh" command

Networks are most useful when other devices are connected and reachable through the network. The "ip neigh" command gives engineers a view at the other hosts connected to the same network. Specifically, it offers a look at, and control of, the ARP table for the host.

Do I have an ARP entry for the host that I'm having trouble connecting to?

A common problem network engineers are called on to support is when one host can't talk to another host. If I had a nickel for every help desk ticket I've worked on like this one, I'd have an awful lot of nickels. Suppose my attempts to ping a host on my same local network with IP address 172.16.211.30 are failing. The first step I might take would be to see if I've been able to learn an ARP entry for this host.

```
(main) expert@expert-cws:~$ ping 172.16.211.30
PING 172.16.211.30 (172.16.211.30) 56(84) bytes of data.
^C
--- 172.16.211.30 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2039ms

(main) expert@expert-cws:~$ ip neigh show
172.16.211.30 dev ens160  FAILED
172.16.211.254 dev ens160 lladdr 00:50:56:f0:11:04 STALE
172.16.211.2 dev ens160 lladdr 00:50:56:e1:f7:8a STALE
172.16.211.1 dev ens160 lladdr 8a:66:5a:b5:3f:65 REACHABLE
```

And the answer is no. The attempt to ARP for 172.16.211.30 "FAILED." However, I can see that ARP in general is working on my network, as I have other "REACHABLE" addresses in the table.

Another common use of the "ip neigh" command involves clearing out an ARP entry after changing the IP address configuration of another host (or hosts). For example, if you replace the router on a network, a host won't be able to communicate with it until the old ARP entry ages out and the system tries ARPing again for a new address. Depending on the operating system, this can take minutes — which can feel like *years* when waiting for a system to start responding again. The "ip neigh flush" command can clear an entry from the table immediately.

How do I get from here to there? Exploring the "ip route" command

Most of the traffic from a host is destined somewhere on another layer 3 network, and the host needs to know how to "route" that traffic correctly. After looking at the IP address(es) configured on a host, I will often take a look at the routing table to see if it looks like I'd expect. For that, the "ip route" command is the first place I look.

What routes does this host have configured?

```
(main) expert@expert-cws:~$ ip route show
default via 172.16.211.2 dev ens160 proto dhcp src 172.16.211.128 metric 100
10.233.44.0/23 via 172.16.211.130 dev ens160
172.16.211.0/24 dev ens160 proto kernel scope link src 172.16.211.128
172.17.0.0/16 dev docker0 proto kernel scope link src 172.17.0.1 linkdown
```

It may not look exactly like the output of "show ip route" on a router, but this command provides very usable output.

- My default gateway is 172.16.211.2 through the "ens160" device. This route was learned from DHCP and will use the IP address configured on my "ens160" interface.
- There is a static route configured to network 10.233.44.0/23 through address 172.16.211.130
- And there are 2 routes that were added by the kernel for the local network of the two configured IP addresses on the interfaces. But the "docker0" route shows "linkdown" — matching the state of the "docker0" interface we saw earlier.

The "ip route" command can also be used to add or delete routes from the table, but with the same notes as when we used "ip link" to change the MTU of an interface. You'll need admin rights to run the command, and any changes made will not be maintained after a restart. But this can still be very handy when troubleshooting or working in the lab.

And done… or am I?

So that's is my "brief" look at the "ip" command for Linux. Oh wait, that bad pun attempt reminded me of one more tip I meant to include. There is a "–brief" option you can add to any of the commands that reformats the data in a nice table that is often quite handy. Here are a few examples.

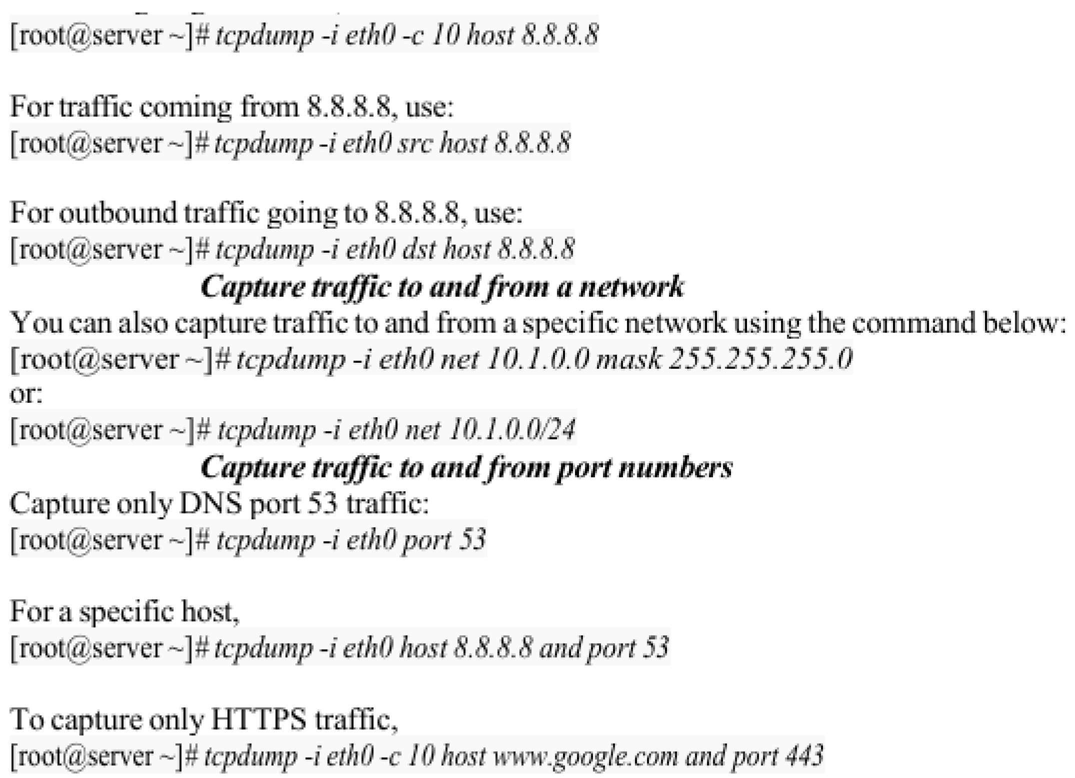
```
(main) expert@expert-cws:~$ ip --brief address show
lo               UNKNOWN        127.0.0.1/8 ::1/128
ens160           UP             172.16.211.128/24 fe80::20c:29ff:fe75:9927/64
docker0          DOWN           172.17.0.1/16

(main) expert@expert-cws:~$ ip --brief link show
lo               UNKNOWN        00:00:00:00:00:00 <LOOPBACK,UP,LOWER_UP>
ens160           UP             00:0c:29:75:99:27 <BROADCAST,MULTICAST,UP,LOWER_UP>
docker0          DOWN           02:42:9a:0c:8a:ee <NO-
CARRIER,BROADCAST,MULTICAST,UP>
```

Not all commands have a "brief" output version, but several do, and they are worth checking out.

There is quite a bit more I could go into on how you can use the “ip” command as part of your Linux network administration skillset. (Checkout the “–json” flag for another great option).

OUTPUT:



```
[root@server ~]# tcpdump -i eth0 -c 10 host 8.8.8.8
```

For traffic coming from 8.8.8.8, use:

```
[root@server ~]# tcpdump -i eth0 src host 8.8.8.8
```

For outbound traffic going to 8.8.8.8, use:

```
[root@server ~]# tcpdump -i eth0 dst host 8.8.8.8
```

Capture traffic to and from a network

You can also capture traffic to and from a specific network using the command below:

```
[root@server ~]# tcpdump -i eth0 net 10.1.0.0 mask 255.255.255.0
```

or:

```
[root@server ~]# tcpdump -i eth0 net 10.1.0.0/24
```

Capture traffic to and from port numbers

Capture only DNS port 53 traffic:

```
[root@server ~]# tcpdump -i eth0 port 53
```

For a specific host,

```
[root@server ~]# tcpdump -i eth0 host 8.8.8.8 and port 53
```

To capture only HTTPS traffic,

```
[root@server ~]# tcpdump -i eth0 -c 10 host www.google.com and port 443
```

RESULT:

Thus the basic commands in window/Linux operating Systems were studied