Recipes for Networking

It's always fun to try out different hacks under the GNU/Linux freedom platform. The pride of becoming a command-line wizard makes everyone stay close to the CLI. Moreover, the CLI vests you with the ultimate power to control your machine.

In this article, we'll explore networking under GNU/Linux. You'll find it interesting to manage the entire network through certain valid keystrokes known as commands. Imagine that you have to access the contents of several other machines from a mount point in your machine. Then imagine shutting down, rebooting, and installing applications on those remote machines, all at one time? Could you configure the WLAN and LAN interfaces from the CLI? This tutorial gives you some insights to the exciting bytes on controlling your network under GNU/Linux.

First we will learn the 'Hello World' of a networked machine.

Let us *ping*!

ping is a universal command that is available on every operating system to test the reachability of a network. When you shoot your terminal with the *ping* command and an IP address as its argument, the machine will try to send some bits of raw data towards the machine owning that IP address. If some machine exists with that IP address, it will send back certain bits. Thus the machine receives the bits and it confirms that a path is available from the current machine to the other through a network. We can check the existence of certain machines on the network by just pinging.

To see which machines are up in the current network, let's write a bit of shell script. Open *vim* as the root:

```
# vim /usr/bin/netup.sh
```

...and key in the following lines in it:

```
#/bin/bash
for i in 192.168.1.{1..255}; // checks 192.168.1.x class of IPs.
do
ping -c2 $i > /dev/null;
[ $? -eq 0 ] && echo $i is up.

trap "exit" SIGINT // To force exit when Ctrl+C keystroke is
applied.
done
```

Save the file, and make it executable by running the following command:

```
chmod a+x /usr/bin/netup.sh
```

Now, run the script as:

```
[slynux@gnuxbox ~]$ netup.sh
192.168.1.1 is up.
192.168.1.3 is up.
192.168.1.4 is up.
```

Configuring your network

Now, let us look at how to configure your machine on the network. You can configure it using two methods. It can be configured manually by the *ifconfig* command for static IP addressing or it can be done via the DHCP (Direct Host Control Protocol).

Static IP addressing is the one that you explicitly instruct the system to use by giving an IP address for a given Ethernet or wireless interface. In case you're using the DHCP, simply issuing the *dhclient* command will fetch the

system an available IP address from the DHCP server in your network. Note that it may not be the same IP address that your machine fetches each time you issue *dhclient*.

Interface cards

Machines are networked either via network cables or using wireless protocols. LAN cards used for networking are known as Ethernet and wireless LAN (WLAN) cards. We interface the network via this outlet. In *nix platforms, Ethernet cards or WLAN cards are denoted as *eth0, eth1,* etc, or *wlan0, wlan1,* etc, respectively.

ifconfig

We have *ifconfig,* a.k.a the interface config, for setting up a network on the machine. To get information about the availability of interface devices available on the current machine, open a terminal and execute the following as the root:

```
[slynux@gnuxbox ~]# ifconfig -a
eth0      Link encap:Ethernet  HWaddr 00:1C:23:FB:37:22
          inet6 addr: fe80::21c:23ff:fefb:3722/64 Scope:Link
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:9724 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2720 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2400589 (2.2 MiB)  TX bytes:645396 (630.2 KiB)
          Interrupt:17

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:76320 errors:0 dropped:0 overruns:0 frame:0
          TX packets:76320 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:87151068 (83.1 MiB)  TX bytes:87151068 (83.1 MiB)

wlan0     Link encap:Ethernet  HWaddr 00:1C:BF:87:25:D2
          inet addr:192.168.1.143  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::21c:bfff:fe87:25d2/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:45302 errors:0 dropped:0 overruns:0 frame:0
          TX packets:37510 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:31091293 (29.6 MiB)  TX bytes:9734025 (9.2 MiB)
```

Here I have three interfaces -- *eth0*, *lo* and *wlan0,* where:

- *eth0* corresponds to the Ethernet card
- *lo* corresponds to a loopback device that points to the localhost network
- *wlan0* corresponds to the wireless LAN card

Static IP addressing

For static IP addressing, issue the following command as the root:

```
ifconfig <device name> <ip address>
```

For example:

```
ifconfig eth0 192.168.0.2
```

ifconfig -a gives you details of all interface devices and configurations. In order to receive details of only one Ethernet device, execute *ifconfig eth0.* The following is an example output:

```
[root@gnubox slynux]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:1C:23:FB:37:22
          inet addr:192.168.0.2  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::21c:23ff:fefb:3722/64 Scope:Link
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:9724 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2720 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2400589 (2.2 MiB)  TX bytes:645396 (630.2 KiB)
          Interrupt:17
```

Now, let us go about setting the subnet mask. This is done easily with *ifconfig,* as follows:

```
ifconfig eth0 192.168.0.2 -netmask 255.255.255.0
```

Now that the IP address and subnet mask is configured (or reconfigured), how do you get your Ethernet up (i.e., available) and down? The *ifup* and *ifdown* commands help you with that as follows:

```
ifup eth0

ifdown eth0
```

Wireless networking

In order to hack a wireless card, we have another utility called *iwconfig.* It works similar to *ifconfig,* but it has lots of additional features that are bonded to wireless cards. If we are using a wireless network with static IP, we can attach our wireless card interface to a network as follows:

```
iwconfig wlan0 essid slynux
```

...or:

```
iwconfig wlan0 essid slynux key 8c140b2037
```

...where 'slynux' is the ESSID (that is, the name wireless network) and '8c140b2037' is the security key. Of course, you need to replace these variables with the values that hold good in your network. You can also scan and check the availability of wireless network(s) in your vicinity using the *iwlist* command as follows:

```
[root@gnuxbox~]# iwlist wlan0 scan
wlan0     Scan completed :
        Cell 01 - Address: 00:08:5C:52:E9:83
                ESSID:"slynux"
                Mode:Master
                Channel:11
                Frequency:2.462 GHz (Channel 11)
                Quality=92/100  Signal level:-39 dBm  Noise level=-78 dBm
                Encryption key:off
                Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 18 Mb/s
                        24 Mb/s; 36 Mb/s; 54 Mb/s; 6 Mb/s; 9 Mb/s
                        12 Mb/s; 48 Mb/s
                Extra:tsf=00000000fc021187
```

The above command will list out the various wireless networks available with a number of properties.

Then we can set the IP for the interface card using the *ifconfig* command itself:

```
ifconfig wlan0 192.168.0.5
```

If you are using dynamic addressing, you can obtain the IP address as follows:

```
dhclient wlan0
```

The settings that you've configured with the *ifconfig* tool are available until the system reboot. But it's a waste of time if you need to configure it on every system start. And hence we take the aid of network configuration scripts. On an Ubuntu (or any other Debian-based) system, this file is located at */etc/networks/interfaces*, and contains data similar to the following:

```
auto lo
iface lo inet loopback

iface eth0 inet static
address 164.164.32.101
netmask 255.255.255.240
gateway 164.164.32.97
```

It is necessary to learn this scripting in order to play with your network. The syntax for these are as follows:

1. Add the following lines if you want to configure eth0 as the DHCP:

```
auto eth0
iface eth0 inet dhcp
```

2. Add the following files if you want to configure static IP:

```
auto eth0
iface eth0 inet static
address <ip_address>
netmask <netmask>
gateway <gateway_ip>
```

3. If it is a wireless network, add the following lines along with the above lines:

```
wireless-essid <network_name>
wireless-key <key>
```

Now, to restart the network daemon, execute the following as the root:

```
/etc/init.d/network restart
```

Spoofing a MAC ID

It is a real hassle for cable Internet customers that they are restricted to using a single machine for Internet access. If you want to plug your laptop in your friend's cable Internet connection, you have to call the service provider to refresh the MAC address.

The MAC address is permanent to the hardware and cannot be changed. Since we operate the hardware via the software abstraction layer, it is quite possible to do some software-level cheating for the network card's MAC ID. We can simply spoof it to some other MAC addresses.

You can obtain the original MAC ID from the *ifconfig* output. Mine is as follows:

```
eth0      Link encap:Ethernet  HWaddr 00:1C:23:FB:37:22
```

Now, let's change the last part of the MAC ID from 22 to 23:

```
ifconfig eth0 hw ether 00:1C:23:FB:37:23
```

Now, run *ifconfig* again:

```
[root@gnubox slynux]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:1C:23:FB:37:23
          inet addr:192.168.0.2  Bcast:192.168.0.255  Mask:255.255.255.0
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
          Interrupt:17
```

Easy enough? Well, let's consider the following instance: suppose you are in a Wi-Fi campus and the access to the wireless network is restricted using MAC addressing. You can simply look at your friend's laptop MAC ID and spoof it. Yes! You are now free to access the network. Have fun!

DNS (Domain Name Service)

DNS is responsible for name resolution. When you point your browser to *www.google.com*, it points to a server on the Internet. How does that happen? As you are aware, all networked computers are assigned with IP addresses. But how do you access the Web page hosted in Google's remote machine by simply typing a name like *google.com*?

That phenomenon is achieved using domain name resolution. There are some servers on the Web called name servers (or DNS) that resolve certain names to corresponding IP addresses, like google.com to its corresponding IP address, in our case. So, we should have the IP addresses of the DNS servers (generally provided by the Internet service provider) handy so that we don't have to remember everyone else's when we browse the Web. When we point our browser to google.com, it consults one of these name servers to find out the IP address and thus load the Web page. But where do we configure the IP addresses of these name servers?

If your network is configured with DHCP, there is no need to specify the name server explicitly. For static IPs, it is, however, necessary. We enter the DNS servers' IP addresses in the */etc/resolv.conf* file. Mine looks like the following:

```
nameserver 208.67.222.222

nameserver 208.67.220.220
```

Note that you don't really need to use the DNS addresses provided by your ISP. For safety purposes, I use OpenDNS—the IP addresses are listed in the above snippet. You can learn more on why OpenDNS is a much safer bet at *www.opendns.org*.

SSH (Secure Shell)

SSH can be defined as the blood of *nix networks. SSH enables users and administrators to make remote logins to other machines that are connected through any kind of network. If you know the user name, password and IP address of another machine on the network, you can remotely log in to that machine and work on it as if you are actually working in front of *that* machine. The following is an example in which I'm authenticating to a system with the IP address of 192.168.1.3 as the user test:

```
[root@gnubox ~]# ssh test@192.168.1.3
The authenticity of host '192.168.1.3 (192.168.1.3)' can't be established.
RSA key fingerprint is 9f:61:ae:ac:8f:75:bb:3a:02:4a:f4:6c:7d:b9:0d:07.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.3' (RSA) to the list of known hosts.
test@192.168.1.3's password:
-sh-3.2$ echo I am on 192.168.1.3 Machine
I am on 192.168.1.3 Machine
-sh-3.2$
```

You can open the CD tray of the other machine, close the tray, shutdown, reboot the machine -- depending on the privileges the user name you've logged in with, has.

sftp is an extension to the *ssh* protocol that helps us to use the SSH connection to transfer files between machines. The following is an example:

```
[root@localhost ~]# sftp test@192.168.1.3
Connecting to 192.168.1.3...
test@192.168.1.3's password:
sftp> ls
Desktop  Documents  Download  Music  Pictures  Public  Templates
Videos  a.out  test.bin  file.cpp  t.c
sftp> get t.c
Fetching /home/test/t.c to t.c
/home/test/t.c                    100%  239    0.2KB/s   00:00
sftp>
```

To download a file from the remote machine we use the *get* command, and to upload a file, we use *put*. In the above snippet you can see that I'm downloading a file named t.c using the *get* command, after logging in to the remote machine using *sftp*.

sshfs is another extension to SSH, which empowers you to mount directories on a remote machine as a filesystem to a specified mount point:

```
root@localhost ~]# sshfs test@192.168.1.3:/home/test /mnt/test
test@192.168.1.3's password:
```

In the above snippet, I'm mounting the home directory of the user 'test' on 192.168.1.3 to my local machine under the */mnt/test* directory.

Proxy server configuration

Many of us on a college campus or office network access the Internet through a proxy server. How do you set the proxy server details in your shell environment? You can set the proxy for different protocols as follows:

```
export http_proxy="http://192.168.0.1:3128" ; // HTTP proxy
export ftp_proxy="192.168.0.1:3128" ; //FTP proxy
```

If you want these settings to be permanent, each time you log in add these lines to your *~/.bash_profile* file.

That's all, folks! Hope you have enjoyed learning the secrets of networking. Happy hacking till we meet again!

By: Sarath Lakshman

The author is an 18 year old hacker and free software enthusiast from Kerala. He loves working on the GNU/Linux environment and contributes to the PiTiVi video editor project. He is also the developer of SLYNUX, a distro for newbies. He is currently studying at Model Engineering College, Cochin. He blogs at www.sarathlakshman.info