Displaying IP addresses

The `ifconfig` command displays details of every active network interface available on the system. However, we can restrict it to a specific interface using:

```
$ ifconfig iface_name
```

For example:

```
$ ifconfig wlan0
wlan0     Link encap:EthernetHWaddr 00:1c:bf:87:25:d2
inet addr:192.168.0.82  Bcast:192.168.3.255
    Mask:255.255.252.0
```

From the outputs of the previously mentioned command, our interests lie in the IP address, broadcast address, hardware address, and subnet mask. They are as follows:

- `HWaddr 00:1c:bf:87:25:d2` is the hardware address (MAC address)
- `inet addr:192.168.0.82` is the IP address
- `Bcast:192.168.3.255` is the broadcast address
- `Mask:255.255.252.0` is the subnet mask

In several scripting contexts, we may need to extract any of these addresses from the script for further manipulations. Extracting the IP address is a frequently needed task. In order to extract the IP address from the `ifconfig` output use:

```
$ ifconfig wlan0 | egrep -o "inet addr:[^ ]*" | grep -o "[0-9.]*"
192.168.0.82
```

Here ,the first command `egrep -o "inet addr:[^ ]*"` will print `inet addr:192.168.0.82`. The pattern starts with `inet addr:` and ends with some non-space character sequence (specified by `[^ ]*`). Now in the next pipe, it prints the character combination of digits and `'.'`.

Spoofing the hardware address (MAC address)

In certain circumstances where authentication or filtering of computers on a network are based on the hardware address, we can use hardware address spoofing. The hardware address appears in `ifconfig` output as `HWaddr 00:1c:bf:87:25:d2`.

We can spoof the hardware address at the software level as follows:

```
# ifconfig eth0 hw ether 00:1c:bf:87:25:d5
```

In the preceding command, `00:1c:bf:87:25:d5` is the new MAC address to be assigned. This can be useful when we need to access the Internet through MAC-authenticated service providers that provide access to the Internet for a single machine. However, note that this only lasts until a machine restarts.

Name server and DNS (Domain Name Service)

The elementary addressing scheme for the Internet is IP addressing (dotted decimal form, for example, `202.11.32.75`). However, the resources on the Internet (for example, websites) are accessed through a combination of ASCII characters called **URLs** or **domain names**. For example, `www.google.com` is a domain name and it corresponds to one (or more) IP address. Typing the IP address in the browser can also access the URL `www.google.com`.

This technique of abstracting IP addresses with symbolic names is called **Domain Name Service** (**DNS**). When we enter www.google.com, our computer uses the DNS servers configured with the network to resolve the domain name into the corresponding IP address. While on a local network, we set up the local DNS for naming local machines on the network symbolically using their hostnames.

Name servers assigned to the current system can be viewed by reading `/etc/resolv.conf`, for example:

```
$ cat /etc/resolv.conf
nameserver 8.8.8.8
```

We can add name servers manually as follows:

```
# echo nameserver IP_ADDRESS >> /etc/resolv.conf
```

How can we obtain the IP address for a corresponding domain name? The easiest method to obtain an IP address is by trying to ping the given domain name and looking at the echo reply. For example:

```
$ ping google.com
PING google.com (64.233.181.106) 56(84) bytes of data.
Here 64.233.181.106 is the corresponding IP address.
```

A domain name can have multiple IP addresses assigned. In that case, `ping` will show one address among the list of IP addresses. To obtain all the addresses assigned to the domain name, we should use a DNS lookup utility.

DNS lookup

There are different DNS lookup utilities available from the command line, which will request a DNS server for an IP address resolution. `host` and `nslookup` are two of such DNS lookup utilities.