Configuring An Interface

In this lesson, we'll configure the network interface in your machine for the first time, using the **ip** utility. We'll also go over network routes and learn how they affect the path that your packets take to reach their destinations.

First Steps

Before you can configure your machine's IP address, you need to know the address to assign to it. To figure out what your IP address should be for this course, use the **host** command. First, launch a terminal in CodeRunner by clicking the **New Terminal** () button. This starts a new shell for you on our "cold" login server. If you're not automatically logged in, go ahead and log in. Once you have a command prompt, you can use the **host** command to look up your machine's IP address. The hostname you should use for the lookup is "*username*-m0.unix.useractive.com," replacing *username* with your username.

INTERACTIVE SESSION:

```
cold1:~$ host username-m0.unix.useractive.com
username-m0.unix.useractive.com has address 172.16.n.n
```

The IP returned will have numbers in place of each *n* and is the IP address of your LLE. Make a note of your IP address; this will be used as the IP of the ethernet device you will be setting up on your LLE. If you receive any kind of error message, notify your mentor. Once you have your IP address, type **exit** to log out of the terminal session.

Link Status

If you haven't already started your machine and logged into it, do so now. Before we do any configuration, let's take a look at the state of your ethernet interface, using the **ip** command. The format that the ip command uses is:

OBSERVE:

```
ip object command
```

We'll introduce objects and commands as we use them. The first **object** we're concerned with is **link**, which will allow us to control and view the link status of our interfaces. The **show** command will print out the current status for us:

INTERACTIVE SESSION:

```
[username@username-m0 ~]$ ip link show
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
2: sit0: <NOARP> mtu 1480 qdisc noop state DOWN
    link/sit 0.0.0.0 brd 0.0.0.0
```

```
3: eth0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN
qlen 1000
    link/ether fe:fd:00:00:04:14 brd ff:ff:ff:ff:ff:ff
```

That's quite a confusing mess of information! In fact, it's a lot more information than we actually need. Of the three interfaces listed, we are only concerned with the status of one of them: eth0. The name of this interface is derived from **eth**ernet interface **0**. If you had two ethernet interfaces in your computer, they would be called "eth0" and "eth1." All ethernet interfaces under Linux are named this way. Let's rerun the **ip** command, but this time we'll tell it we only want to know about eth0. The important part of the output is in **red**:

INTERACTIVE SESSION:

```
[username@username-m0 ~]$ ip link show eth0
3: eth0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN
qlen 1000
    link/ether fe:fd:00:00:04:14 brd ff:ff:ff:ff:ff:ff
```

Much better! We can see that as of this instant, **eth0's link is down**. We can't connect to anything while our interface is down, so let's bring it up. To do this, we need root privileges, so we'll use **sudo** with the **ip** command. Again, we highlighted the important parts in **red**:

INTERACTIVE SESSION:

```
[username@username-m0 dbassett]# sudo ip link set eth0 up
We trust you have received the usual lecture from the local
System
Administrator. It usually boils down to these three things:
#1) Respect the privacy of others.
#2) Think before you type.
#3) With great power comes great responsibility.
[sudo] password for username:
[username@username-m0 dbassett]# ip link show eth0
3: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
pfifo_fast state UNKNOWN qlen 1000
    link/ether fe:fd:00:00:04:14 brd ff:ff:ff:ff:ff:ff
```

Note There is one weird issue with the output of **ip link show eth0**. The state is **UNKNOWN**. Usually when the interface is brought up, we would see "state UP." But due to limitations with our virtualization environment, the **ip**command can't report the status of the interface correctly. Fortunately, the status flags displayed between the < and > brackets are correct.

Based on the status flags between the < and > brackets, our interface is **UP**. Also take note of the flag that says **LOWER_UP**. There is an important distinction to be made between "UP" and "LOWER_UP." "UP" just means that the interface is activated for the OS to use; "LOWER_UP" means that the interface has an active physical link. If this were a real machine and its ethernet cable was unplugged, you would not see "LOWER_UP," and you would not be able to transmit data.

Setting An Address

We have both "UP" and "LOWER_UP," but we still can't do anything useful yet. We still have to assign an IP address and netmask to the interface. This can also be done with the **ip** command, using the **addr** (address) object. We'll give the **ip** command an address to assign to the interface using dotted quad notation, and we'll specify the netmask using prefix length notation. You may recall that your machine will be on the 172.16.0.0/12 network. This means that you use /12 (or 255.240.0.0 in dotted quad notation) as your subnet mask. You can use this information, along with the IP address you found at the beginning of this lesson, to set the IP address of your machine (substitute your IP address for "172.16.*n.n*"):

INTERACTIVE SESSION:

```
[username@username-m0 ~]$ sudo ip addr add 172.16.n.n/12 dev eth0
[username@username-m0 ~]$ ip addr show eth0
3: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN qlen 1000
    link/ether fe:fd:00:00:04:14 brd ff:ff:ff:ff:ff:ff
    inet 172.16.n.n/12 scope global eth0
    inet6 fe80::fcfd:ff:fe00:414/64 scope link
       valid_lft forever preferred_lft forever
```

Let's break down that ip command a little to see what we've done. Running ip with the **addr** object tells it that we want to work with the address settings for some device (in our case, **eth0**, as specified at the end of the command). We want to add an address to this interface, so we use the **add** command. The **add** command requires that we specify an address (with netmask), and a device where we'll add that address. You specify the device as **dev *devicename***. After we've added the address, we want to check to make sure it's actually there; we can use the **show** command with the **addr** object to check the current status of the interface. As with the **link** object, specifying the device you want to show information for is optional, but useful.

Now that your interface is up and has an address, you can test to see if it's working. A good tool for this task is **ping**. Ping sends a special kind of data packet, called an ICMP packet, to a specified host. When that host receives the ICMP packet, it generally sends back a response to let you know that it can be reached. In some circumstances, however, the host on the other end will be configured not to respond to pings. If you know of a host that is configured to return pings, you can ping that address as a quick

test to make sure your network interface is sending and receiving traffic. Let's ping an address on our network to see if everything is working as it should (use **Ctrl+c** to stop ping, shown below as **^C**):

INTERACTIVE SESSION:

```
[username@username-m0 ~]$ ping 172.16.0.1
PING 172.16.0.1 (172.16.0.1) 56(84) bytes of data.
64 bytes from 172.16.0.1: icmp_seq=1 ttl=64 time=15.2 ms
64 bytes from 172.16.0.1: icmp_seq=2 ttl=64 time=0.673 ms
^C
--- 172.16.0.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1859ms
rtt min/avg/max/mdev = 0.673/7.982/15.292/7.310 ms
```

It looks like everything is working properly. Now let's try pinging a publicly accessible address, the O'Reilly School of Technology's website:

INTERACTIVE SESSION:

```
[username@username-m0 ~]$ ping 199.27.144.89
connect: Network is unreachable
```

Network is unreachable? But we were just able to ping something else! This is what happens when your machine doesn't know where to send a packet. At this point, you are able to ping hosts on your network, but your machine doesn't know how to communicate with machines outside of your network. To tell your machine what to do with packets destined for other networks, you use a *network route*. The idea behind a network route is that a packet that is not destined for your network can be sent to another device on the network (like a router or gateway) that knows where the packet should go and can handle forwarding it for you. In normal usage, defining something called the *default route* is good enough. The default route is where all packets not destined for your network (or in more advanced usage, another specifically defined route) are sent. Ideally, the device on the other end of the default route will know where to send your packets.

Network Routes

Once again, the **ip** command can help us achieve our goal. We want to set a default route. Fortunately, **ip** has a **route** object for us to use. Before we add anything, let's take a look at the status of our *routing table*. It holds information regarding routes to various networks:

INTERACTIVE SESSION:

```
[username@username-m0 ~]$ ip route show
172.16.0.0/12 dev eth0  proto kernel  scope link  src 172.16.n.n
```

Note It seems that there's a pattern developing here. Whenever you want to see the status of an aspect of the networking on your machine using **ip**, you run **ip *object* show**.

We have exactly one route in our routing table right now, and that route points to our own network. We could anticipate this to be the case because our machine knows how to handle packets bound for our network. Now let's add a default route so we can send packets outside:

INTERACTIVE SESSION:

```
[username@username-m0 ~]$ sudo ip route add default via
172.16.0.1
[username@username-m0 ~]$ ip route show
172.16.0.0/12 dev eth0  proto kernel  scope link  src 172.16.n.n
default via 172.16.0.1 dev eth0
```

Now we have two routes. Any additional routes displayed via **ip** will look like the entry for the default route, with the network address displayed instead of "default." For example, if we added a route to the network 192.168.0.0/24, and the gateway to that network was listening on 172.16.0.5, the route entry would look like "192.168.0.0/24 via 172.16.0.5 dev eth0." We've set our route; let's see if we can ping outside of our network now:

INTERACTIVE SESSION:

```
[username@username-m0 ~]$ ping 199.27.144.89
PING 199.27.144.89 (199.27.144.89) 56(84) bytes of data.
64 bytes from 199.27.144.89: icmp_seq=1 ttl=63 time=34.9 ms
64 bytes from 199.27.144.89: icmp_seq=2 ttl=63 time=0.859 ms
^C
--- 199.27.144.89 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1710ms
rtt min/avg/max/mdev = 0.859/17.921/34.984/17.063 ms
```

Brilliant! You can now reach the outside world. You may want to take a look at **ip**'s **man** page to see what else it's capable of doing before you move on.

Source:

http://courses.oreillyschool.com/sysadmin2/configuring_an_interface.html