

## Basic Linux Network Setup

In this lab, you will learn how to setup basic Linux TCP/IP networking to add your machine to the lab network. Note: the lab network is a private, isolated network - they cannot reach the outside world.

1. Boot your system in single user mode. To accomplish this, reboot the system, and interrupt the Red Hat splash screen at the beginning of the boot process with Control-X. Then type "linux single" at the boot: prompt.
2. Last week, you determined the name of your NIC device driver. Recalling that name, edit the file `/etc/modules.conf`, adding the line:

```
alias eth0 your-module-name
```

Some of the systems may already have such a line in the file - if so, skip this step. Otherwise, add the line above, save the file and exit.

3. Ensure the module loads correctly. Use the `insmod` command to load your module. If it loads successfully, you will see some initialization messages printed out by the driver. Unload the module using `rmmmod`. If the module did not load correctly, go back and figure out the name of the module, and restart at step 2.
4. Set the hostname for the machine. Use the hostname `host` plus your machine number (e.g. `host4` or `host23`). Edit the file `/etc/sysconfig/network` and add the lines:

```
HOSTNAME=your-hostname  
NETWORKING=yes
```

The `HOSTNAME` variable is used by Red Hat Linux to assign as the hostname during startup. The `NETWORKING` variable indicates that networking is enabled. Save the file and exit.

5. You also should update the `/etc/hostname` file, a file used now for backwards compatibility (it is also commonly used on other UNIX systems). Add just one line, your hostname, to the file.
6. Change directories into `/etc/sysconfig/network-scripts`. Create the file named `ifcfg-eth0`. This file is used to store network settings for the `eth0` device. The `DEVICE` variable is used to specify the device, `NETMASK`, `NETWORK`, and `BROADCAST` are all standard IP values, and `ONBOOT` indicates that networking should be enabled at boot time. You will be adding the lines below. Where you see the `NNN`, replace it with the value of 100 + your machine number (e.g. 104 and 123 for machines number 4 and 23, respectively).

```
DEVICE=eth0  
NETMASK=255.255.255.0  
IPADDR=192.168.10.NNN  
NETWORK=192.168.10.0  
BROADCAST=192.168.10.255  
ONBOOT=yes
```

7. Use the script in the current directory named `ifup` to establish that at least your networks *Network Layer* works. This program will load the NIC's device driver, and use the settings from the file `ifcfg-eth0` to configure IP. When the *Network Layer* is established, the IP protocol is ready for use.

```
$ ./ifup eth0
```

You may see the same initialization messages you saw in step 3, but most likely the command will be silent. If there are any errors or problems bring up the interface, go back and resolve them before moving on. Most likely your settings in the `ifcfg-eth0` file are problematic.

8. Use the **ifconfig** command to check that your network settings look correct. Look for the **inet addr**, **Bcast**, and **Mask** values. Also look for the word **UP** on the next line.

```
$ ifconfig -a
```

9. Use the command **ping 192.168.10.1** to send a networking packet to an address, to test that packets are leaving the NIC and going out to the network. Since there is no machine at that IP address, don't expect a *reply*. Also, ping'ing your own machine does not accomplish anything. Instead, you can look at the lights on the *hub* to which your machine is attached, and see if your light is flashing. If it is flashing, the hub is receiving packets. Kill the ping with Control-C. Again, if there are problems, fix them before moving on.
10. Disable the **eth0** network *interface* with the command:

```
$ ./ifdown eth0
```

11. You can now bring the machine up to multi-user level 3 (do not start a graphical session) and networking should be enabled. Use the command:

```
$ init 3
```

12. Discover which of your classmates has their networks up and running. Use the **ping** command to test connectivity from your machine to theirs:

```
$ ping their-ip-address
```

You are looking for the response from **ping** that indicates the host is *alive*. If not, try other host numbers. This is a good first step connectivity test.

13. Now try using **telnet** to connect to some machine using the **joeuser** account that was setup in lab 1. The password should also be **joeuser**:

```
$ telnet their-ip-address
```

14. Typing IP addresses is a pain - host names are nicer. To associate a host with an IP address, edit the file **/etc/hosts** and add a new line that looks like:

```
IP-address hostname
```

where IP-address is the IP address of the machine you want to call *hostname*. If you are connecting to host **host20**, then your entry would like like:

```
192.168.10.120 host20
```

Do not remove the **localhost** entry!

15. Now try steps 11 and 12 again using the host's hostname instead of an IP address.
16. In this step, you will enable **rsh**-type logins. You must start the **portmap** daemon. Run the command:

```
$ /etc/init.d/portmap start
```

17. Now try logging into your own machine using the **rsh** (remote shell) command:

```
$ rsh your-hostname -l joeuser
```

Exit the **rsh** session just like you exit any shell.