

Lab 1: Basic Network Setup

In this lab, you will learn about your machine's networking hardware and setup basic Linux TCP/IP networking to add your machine to the lab network. Perform the indicated steps, and write the answers to questions in the appropriate spaces on this hand-out. Turn-in one completed handout per computer system. Be sure to include your name and your partner's name, if you have one.

Configure the System to Boot Single-user

Boot the system into single-user mode. Set the system to boot into single-user mode by default.

- Step 1. Boot your system into 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.
- Step 2. Setup the system to boot into single user mode each time the system is booted. This will make configuring network services easier. Edit the file `/etc/inittab` and change the line that contains **initdefault**. There is a number there, either a 3 or a 5. Change it to a 1, which tells Linux to boot into single user mode.

Configure basic networking

Learn about the basic system networking hardware and how to configure and activate the network interface.

- Step 3. To enable networking, you need a device driver for the network interface card (NIC). Your machine has a 3Com NIC, either model **3c509** or **3c59x**. The server machines (the two machines in the middle of each row of tables) have two NICs, one of each type. Examine the NICs in your system to see which one(s) you have. The NIC with the round plug-like BNC connector is the older 3c509 model. The NIC without this connector is the newer 3c59x model.

Q1. How many NICs does your system have? _____

Q2. Which NIC(s) does your system have? _____

- Step 4. Manually load the 3c509 device driver module using **insmod** (recall the module loading utilities **insmod**, **lsmod**, and **rmmod** from CIS68C1 UNIX Administration). Ensure the module is loaded with **lsmod**. If your system has two NICs, try loading the **3c59x** device driver module. If you try to load the wrong device driver module, **insmod** will fail and the device will not function. If it loads successfully, you may see some initialization messages printed out by the driver. Unload the module using **rmmod**.

Q3. Which device driver module(s) works for your NIC? _____

- Step 5. In Linux, networking devices are known as **eth0**, **eth1**, **eth2**, etc. where the first loaded device is always **eth0**, the second **eth1**, etc. But the Linux kernel needs to know the name of the driver module, not the generic **ethN** name used by many startup scripts and networking utilities. To enable the use of generic names, and to allow the module(s) to auto-load upon first use, you need to add an **alias** in the file `/etc/modules.conf` so that the kernel module auto-loader knows which specific kernel device driver module is associated with the generic networking interface name **eth0**, etc. Edit the file `/etc/modules.conf`, add the line below, save the file and exit.

```
alias eth0 module-name
```

- Step 6. Set the hostname for the machine. Your system's hostname will be **hostN**, where *N* is your machine's number (e.g. **host4** or **host18**). Edit the file `/etc/sysconfig/network` and add the lines below. The **HOSTNAME** variable is used by the Red Hat Linux startup scripts to assign as the hostname during startup. The **NETWORKING** variable indicates that networking is enabled. Make the changes, save the file and exit

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

Q4. What system command is responsible for setting the hostname on a UNIX/Linux system? _____

- Step 7. 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 system's hostname, to the file.

Step 8. The network configuration files are stored in `/etc/sysconfig/network-scripts`, and the files read by the network startup scripts are named `ifcfg-ethX`, where `X` is the interface number. Change directories into `/etc/sysconfig/network-scripts` and create the file named `ifcfg-eth0`. This file is used to store network settings for the `eth0` interface. The `DEVICE` variable is used to specify the interface name, `NETMASK`, `NETWORK`, and `BROADCAST` are all standard IP values, and `ONBOOT` indicates that networking should be enabled at boot time. Comment out any existing lines with the `#` character, and add the lines below. Where you see the `N`, replace it with the value of your machine number.

```
DEVICE=eth0
NETMASK=255.255.255.0
IPADDR=10.0.0.N
NETWORK=10.0.0.0
BROADCAST=10.0.0.255
ONBOOT=yes
```

Step 9. Use the Red Hat Linux shell script in the current directory named `ifup` to establish that your system's Network Layer functions correctly. This program will indirectly cause the loading of 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. If there are any errors or problems bring up the interface, they are most likely being caused by incorrect variable names or values in the `ifcfg-eth0` file are problematic, or you specified the wrong device driver in `/etc/modules.conf`: Resolve any problems before moving on.

```
# ./ifup eth0
```

Step 10. The `ifconfig` command is used to set or view network interface parameters such as IP, netmask, etc. 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` at the beginning of the next line; if `UP` is missing, the interface is not up and running.

```
# ifconfig -a
```

Step 11. Use the command `ping 10.0.0.200` to test that packets are leaving the NIC and going out to the network. You can examine the lights for the port 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, which means they are leaving your `eth0` interface and arriving at the hub. Kill the ping with Control-C. Again, if there are problems, fix them before moving on.

Step 12. Try `ping`'ing your system's loopback interface. You can use either the IP address 127.0.0.1 or the name `localhost`.

Q5. Does the ping succeed? _____

Step 13. Now try to ping your `eth0` interface's IP address (10.0.0.N).

Step 14. You probably found that the `ping` failed. This is because your loopback interface is not up. Run `ifconfig -a` again, and notice the absence of the word `UP` in the `lo` interface section. This means the interface is down (not running). Bring up the `loopback` interface with:

```
# ./ifup lo
```

Step 15. Now try to `ping` both your `loopback` interface and your own IP address again. This time both should succeed.

Step 16. It is very important to understand that pinging your own IP address does not prove that you have network connectivity to the network to which your card is connected. This is easily proven by pinging your own IP address (not the loopback address) and unplugging the network cable. You will notice that the ping continues to report responses! This means that packets are reaching their destination without going out over the wire. This occurs because IP intercepts the packets, and sends them instead through the loopback interface. And this is exactly why your ping did not work earlier when the loopback interface `lo` was down.

Step 17. Bring down the `eth0` network interface with the command:

```
# ./ifdown eth0
```

Step 18. You can now bring the machine up to multi-user level 3 (do not start a graphical session – level 5) and networking should be enabled. Use the command:

```
# init 3
```

Step 19. Find out which of your classmates has their networks up and running. Use the **ping** command to test connectivity from your machine to theirs. 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.

```
$ ping destination-ip-address
```

Configure and use telnet

Step 20. Try using the **telnet** command to connect to your own machine using the **student** account that should be setup already.

```
$ telnet your-ip-address
```

Step 21. The telnet above should have failed to connect. This is because **telnet** requires the telnet daemon to be running on the remote machine. Your system may not have the telnet daemon software installed (if it was installed using the **workstation** configuration). To test for this software, run the command:

```
$ rpm -qa | grep telnet-server
```

Step 22. If the above command produced a single line (the package name of the telnet server software) skip to step Step 24. Otherwise, the telnet software is not installed, so you will need to install the telnet server package from the Red Hat Linux CD-ROM (or any place you can find RPMs). Insert a Red Hat Linux Installer Disk 1 into the CD-ROM drive and mount the disk using:

```
# mount /mnt/cdrom
```

Step 23. The Red Hat software is available in RPM package formats in the directory **/mnt/cdrom/RedHat/RPMS**. Change to that directory and install the telnet server package with the command:

```
# rpm -ivh telnet-server*
```

Step 24. Now that you have the telnet server software installed, you can configure the system to start up the server. The telnet daemon is configured to be controlled with **xinetd**, the networking super-daemon. To allow **xinetd** to run the telnet daemon so that others can use **telnet** to connect to your machine, you need to edit the **xinetd** configuration file for the telnet daemon. The file is named **/etc/xinetd.d/telnet**. Change the line in the file **disabled = yes** to **disabled = no**.

Step 25. Now try using **telnet** again as you did above.

Step 26. Changing a configuration file is not enough to cause **xinetd** to re-read its configuration files. The currently running **xinetd** is unaware that you have changed the file **/etc/xinetd.d/telnet**. In fact, and this is very important to understand, *no* daemons will notice changes to their configuration files until you force them to re-read their configuration files. Many daemons will re-read their configuration files when they receive a particular signal (which signal depends on the daemon). To force **xinetd** to re-read its configuration file, review the bottom of the **xinetd** man page to learn how to tell **xinetd** to re-read its configuration files. Hint: Search for the word **signal**.

Step 27. Deliver the appropriate signal to **xinetd** daemon using the **kill** command – make sure you send the correct signal, or you may kill **xinetd**! Check that it is still running using the **ps -ef** command.

Q6. What is the exact command that causes **xinetd** to re-read its configuration file: _____

Step 28. Once again, try to **telnet** into your own machine. Be sure that you are able to connect and login before proceeding.

- Step 29. Now try to **telnet** into one of your classmate's systems. Once you have logged into their system, **telnet** from their system back to your own system. Exit both **telnet** sessions when you have succeeded. You will not be able to telnet as **root** (a security measure) – use your system's **student** account.
- Step 30. Typing IP addresses is tedious and host names are much easier. To associate a host name with an IP address, edit the file **/etc/hosts** and add a new line that looks like the line below, 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 192.168.10.120 host20. Never remove or change the **localhost** entry!

IP-address hostname

- Step 31. Now try to **ping** and **telnet** again using the name of the host instead of its IP address. When you want to use a host's name instead of an IP address, you must have an entry in the **/etc/hosts** file (unless of course you are running DNS or NIS which you will learn about in due course). Add names whenever you want.

Configure and use rsh and rlogin

- Step 32. Try logging into your own machine using the **rlogin** (remote login) command:

```
$ rlogin -l student your-hostname
```

- Step 33. The **rlogin** should have failed to connect. Again, you must install and enable the appropriate daemons, which in this case are the **rsh** and **rlogin** services. Install the **rsh-server*** package, as you did previously for the **telnet** package.
- Step 34. The **rsh** and **rlogin** servers are also controlled via **xinetd**. Configure the **rsh** and **rlogin** configuration files for **xinetd** as you did earlier for **telnet**.
- Step 35. Try to **rlogin** again. Exit the **rlogin** session just like you exit any shell. Once this succeeds, try to **rlogin** to another host.
- Step 36. Try doing the same, this time with **rsh** instead of **rlogin**.

Q7. What is the difference between these two commands? _____