Networking

Netconfig

Computers aren't very interesting on their own. Sure, you can install games on them, but that just turns them into glorified entertainment consoles. Today, computers need to be able to talk to one another; they need to be networked. Whether you're installing a business network with hundreds or thousands of computers or just setting up a single PC for Internet access, Slackware is simple and easy. This chapter should teach you how to setup typical wired networks. Common wireless setup will be thoroughly discussed in the next section, but much of what you read here will be applicable there as well.

There are many different ways to configure your computer to connect to a network or the Internet, but they fall into two main categories: static and dymanic. Static addresses are solid; they are set with the understanding that they will not be changed, at least not anytime soon. Dynamic addresses are fluid; the assumption is that the address will change at some time in the future. Typically any sort of network server requires a static address simply so other machines will know where to contact it when they need services. Dynamic addresses tend to be used for workstations, Internet clients, and any machine that doesn't require a static address for any reason. Dynamic addresses are more flexible, but present complications of their own.

There are many different kinds of network protocols that you might encounter, but most people will only ever need to deal with Internet Protocol (IP). For that reason, we'll focus exclusively on IP in this book.

Manual Configuration

Ok, so you've installed Slackware, you've setup a desktop, but you can't get it to connect to the Internet or your business's LAN (local area network), what do you do? Fortunately, the answer to that question is simple. Slackware includes a number of tools to configure your network connection. The first we will look at is the very powerful ***ifconfig***(8), which is used to setup or modify the configuration of the most common hardware for connecting to networks: a Network Interface Card (NIC or Ethernet Card). ***ifconfig*** is an incredibly powerful tool capable of doing much more than setting IP addresses. For a complete introduction, you should read its man page. For now, we're just going to use it to display and change the network addresses of some ethernet controllers.

```
darkstar:~# ifconfig
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:699 errors:0 dropped:0 overruns:0 frame:0
          TX packets:699 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:39518 (38.5 KiB)  TX bytes:39518 (38.5 KiB)
```

```
wlan0     Link encap:Ethernet  HWaddr 00:1c:b3:ba:ad:4c
          inet addr:192.168.1.198  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::21c:b3ff:feba:ad4c/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1630677 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1183224 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1627370207 (1.5 GiB)  TX bytes:163308463 (155.7 MiB)

wmaster0  Link encap:UNSPEC  HWaddr 00-1C-B3-BA-
AD-4C-00-00-00-00-00-00-00-00-00-00
          UP BROADCAST RUNNING 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)
```

As you can clearly see here, when run without any arguments, ***ifconfig*** will display all the information it has on all the ethernet cards (and wireless ethernet cards) present on your system. The above represents a typical wireless connection from my laptop, so don't be afraid if what you see on your system doesn't match. If you don't see any ethX or wlanX interfaces though, the interface may be down. To show all currently present NICs whether they are *"up"* or *"down"*, simply pass the *-a* argument.

```
darkstar:~# ifconfig -a
eth0      Link encap:Ethernet  HWaddr 00:19:e3:45:90:44
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:122780 errors:0 dropped:0 overruns:0 frame:0
          TX packets:124347 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:60495452 (57.6 MiB)  TX bytes:17185220 (16.3 MiB)
          Interrupt:16

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:699 errors:0 dropped:0 overruns:0 frame:0
          TX packets:699 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:39518 (38.5 KiB)  TX bytes:39518 (38.5 KiB)

wlan0     Link encap:Ethernet  HWaddr 00:1c:b3:ba:ad:4c
          inet addr:192.168.1.198  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::21c:b3ff:feba:ad4c/4 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1630677 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1183224 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1627370207 (1.5 GiB)  TX bytes:163308463 (155.7 MiB)
```

```
wmaster0  Link encap:UNSPEC  HWaddr 00-1C-B3-BA-
AD-4C-00-00-00-00-00-00-00-00-00-00
          UP BROADCAST RUNNING 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)
```

Notice that the eth0 interface is now listed among the returns. ***ifconfig*** can also change the current settings on a NIC. Typically, you would need to change the IP address and subnet mask, but you can change virtually any parameters.

```
darkstar:~# ifconfig eth0 192.168.1.1 netmask 255.255.255.0
darkstar:~# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:19:e3:45:90:44
          inet addr:192.168.1.1  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:122780 errors:0 dropped:0 overruns:0 frame:0
          TX packets:124347 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:60495452 (57.6 MiB)  TX bytes:17185220 (16.3 MiB)
          Interrupt:16
```

If you look carefully, you'll notice that the interface now has the 192.168.1.1 IP address and a 255.255.255.0 subnet mask. We've now setup the basics for connecting to our network, but we still need to setup a default gateway and our DNS servers. In order to do that, we'll need to look at a few more tools.

Next on our stop through networking land is the equally powerful ***route***(8). This tool is responsible for modifying the Linux kernel's routing table which affects all data transmission on a network. Routing tables can become immensely complex or they can be straight-forward and simple. Most users will only ever need to setup a default gateway, so we'll show you how to do that here. If for some reason you need a more complex routing table, you would be well advised to read the entire man page for ***route*** as well as consulting other sources. For now, let's take a look at our routing table immediately after setting up eth0.

```
darkstar:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use
Iface
192.168.1.0     *               255.255.255.0   U     0      0        0 eth0
loopback        *               255.0.0.0       U     0      0        0 lo
```

I won't explain everything here, but the general information should be easy to pick up if you're familiar with networking at all. The Destination and Genmask fields specify a range of IP addresses to match. If a Gateway is defined, information in the form of packets will be sent to that host for forwarding. We also specify an interface in the final field that the information should traverse. Right now, we can only communicate with computers with addresses between 192.168.1.0 and 192.168.1.255 and ourselves through the loopback interface, a type of virtual NIC that is used for routing information from this computer to itself. In order to reach the rest of the world, we'll need to setup a default gateway.

```
darkstar:~# route add default gw 192.168.1.254
darkstar:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use
Iface
192.168.1.0     *               255.255.255.0   U     0      0        0 eth0
loopback        *               255.0.0.0       U     0      0        0 lo
default         192.168.1.254   0.0.0.0         UG    0      0        0 eth0
```

You should immediately notice the addition of a default route. This specifies what router should be used to reach any addresses that aren't specified elsewhere in our routing table. Now, when we try to connect to say, 64.57.102.34, the information will be sent to 192.168.1.254 which is responsible for delivering the data for us. Unfortunately, we're still not quite through. We need some way of converting domain names like slackware.com into IP addresses that the computer can use. For that, we need to make use of a DNS server.

Fortunately, setting up your computer to use an external (or even an internal) DNS server is very easy. You'll need to use your favorite text editor and open the `/etc/resolv.conf` file. Don't ask me what happened to the **e**. On my computer, `resolv.conf` looks like this.

```
# /etc/resolv.conf
search lizella.net
nameserver 192.168.1.254
```

Many users won't need the search line. This is used to map hostnames to domain names. Basically, if I attempt to connect to "*barnowl*", the computer knows to look for *"barnowl.lizella.net"* thanks to this search line. We're mainly interested in the nameserver line. This tells Slackware what domain name servers (DNS) to connect to. Generally speaking, these should always be specified by IP address. If you know what DNS servers you should use, you can just add them one at a time to individual nameserver lines. In fact, I don't know of any practical limit to the number of nameservers that can be specified in `resolv.conf`, so add as many as you like. Once this is done, you should be able to communicate with other hosts via their fully qualified domain name.

But Alan! That's a lot of hard work! I don't want to do this time and again for dozens or even hundreds of machines. You're absolutely right, and that's why smarter people than you and me created DHCP. DHCP stands for Dynamic Host Control Protocol and is a method for automatically configuring computers with unique IP addresses, netmasks, gateways, and DNS servers. Most of the time, you'll want to use DHCP. The majority of wireless routers, DSL or cable modems, even firewalls all have DHCP servers to can make your life much easier. Slackware includes two main tools for connecting to an exising DHCP server and can even act as a DHCP server for other computers. For now though, we're just going to look at DHCP clients.

First on our list is ***dhcpcd***(8), part of the ISC DHCP utilities. Assuming your computer is physically connected to your network, and that you have an operating DHCP server on that network, you can configure your NIC in one shot.

```
darkstar:~/ dhcpcd eth0
```

If everything went according to plan, your NIC should be properly configured, and you should be able to communicate with other computers on your network, and with the Internet at large. If for some reason, ***dhcpcd*** fails, you may want to try ***dhclient***(8). ***dhclient*** is an alternative to ***dhcpcd*** and

works in basically the same way.

```
darkstar:~/ dhclient eth0
Listening on LPF/eth0/00:1c:b3:ba:ad:4c
Sending on   LPF/eth0/00:1c:b3:ba:ad:4c
Sending on   Socket/fallback
DHCPREQUEST on eth0 to 255.255.255.255 port 67
DHCPACK from 192.168.1.254
bound to 192.168.1.198 -- renewal in 8547 seconds.
```

Why does Slackware include two DHCP clients? Sometimes a particular DHCP server may be broken and not respond well to either ***dhcpcd*** or ***dhclient***. In those cases, you can fall back to the other DHCP client in hopes of getting a valid response from the server. Traditionally, Slackware uses ***dhcpcd***, and this works in the vast majority of cases, but it may become necessary at some point for you to use ***dhclient*** instead. Both are excellent DHCP clients, so use whichever you prefer.

Automatic Configuration with rc.inet1.conf

Manually configuring interfaces is an important skill to have, but it can become tedious. No one wants to manually setup their Internet connection every time the system boots. More importantly, you may not always have physical access to the machine when it boots. Slackware makes it easy to automatically configure ethernet (and wireless) cards at system startup with `/etc/rc.d/rc.inet1.conf`. For now, we're going to focus on traditional wired ethernet networking; the next chapter will discuss various wireless options.

`rc.inet1.conf` is an incredibly powerful configuration file, capable of configuring most of your network cards automatically when Slackware is started. The file is filled with useful comments, but there is also a man page that more thoroughly discusses its use. To begin, we're going to look at some of the options used on one of my personal machines.

```
# Config information for eth0:
IPADDR[0]="192.168.1.250"
NETMASK[=]"255.255.255.0"
USE_DHCP[0]=""
DHCP_HOSTNAME[0]=""
# Some lines ommitted.
GATEWAY="192.168.1.254"
```

This represents most of the information necessary to configure a static IP address on a single ethernet controller. ***netconfig*** will usually fill in these values for a single ethernet device for you. If you have multiple network cards in your machine and need all of them activated automatically at boot time, then you'll need to edit or add additional entries into this file in the same manner as above. First, let me go over some of the basics.

As you may have already guessed, IPADDR[n] is the Internet Protocol Address for the **n** network interface card. Typically, **n** corrosponds to *eth0*, *eth1*, and so on, but this isn't always the case. You can specify these values to pertain to a different network controller with the IFNAME[n] variable, but we will reserve that for wireless networking, as it more commonly pertains to wireless network controllers.

Likewise, NETMASK[n] is the subnet mask to use for the network controller. If these lines are left empty, then static IP addresses will not be automatically assigned to this network controller. The **USE_DHCP[n]** variable tells Slackware (naturally) to use DHCP to configure the interface. **DHCP_HOSTNAME[n]** is rarely used, but some DHCP servers may require it. In that case, it must be set to a valid hostname. Finally, we come to the GATEWAY variable. It is actually set lower in the file than it appears in my example, and it controls the default gateway to use. You may be wondering why there is no **GATEWAY[n]** variable. The answer to that lies in how Internet Protocol works. I won't go into an in-depth discussion on that subject, but suffice it to say that there is only ever one default route that a computer can use no matter how many interfaces are attached to it.

If you need to use static IP addressing, you will have to obtain a unique static IP address and the subnet mask for the interface, as well as the default gateway address, and enter those here. There is no place to enter DNS information in `rc.inet1.conf`, so DNS servers will have to be manually placed into `resolv.conf` as discussed in Manual Configuration. Of course, if you use ***netconfig***, this will be handled for you by that program. Now let's take a look at another interface on my computer.

```
# Config information for eth1:
IPADDR[1]=""
NETMASK[1]=""
USE_DHCP[1]="yes"
DHCP_HOSTNAME[1]=""
```

Here I am telling Slackware to configure eth1 using DHCP. I do not need to set the **IPADDR[1]** or **NETMASK[1]** variables when using DHCP (in fact, if they are set, they will be ignored). Slackware will happily contact a DHCP server as soon as the machine begins to boot.

Chapter Navigation

Previous Chapter: Emacs

Next Chapter: Wireless Networking

Sources

- Original source: http://www.slackbook.org/beta
- Originally written by Alan Hicks, Chris Lumens, David Cantrell, Logan Johnson

slackbook, networking, netconfig, dhcpcd