Networking with TCP/IP.

Linux supports a full implementation of the TCP/IP (Transport Control Protocol/Internet Protocol) networking protocols. TCP/IP has become the most successful mechanism for networking computers worldwide. With Linux and an Ethernet card, you can network your machine to a local area network, or (with the proper network connections) to the Internet--the worldwide TCP/IP network.

Hooking up a small LAN of UNIX machines is easy. It simply requires an Ethernet controller in each machine and the appropriate Ethernet cables and other hardware. Or, if your business or university provides access to the Internet, you can easily add your Linux machine to this network.

The current implementation of TCP/IP and related protocols for Linux is called ``NET-3,'' and before that, ``NET-2.'' This has no relationship to the so-called NET-2 release of BSD UNIX; instead, ``NET-3'' in this context means the second implementation of TCP/IP for Linux.

Linux NET-3 also supports SLIP--Serial Line Internet Protocol and PPP--Point-to-Point Protocol. SLIP and PPP allow you to have dialup Internet access using a modem. If your business or university provides SLIP or PPP access, you can dial in to the SLIP or PPP server and put your machine on the Internet over the phone line. Alternately, if your Linux machine also has Ethernet access to the Internet, you can set up your Linux box as a SLIP or PPP server.

For complete information on setting up TCP/IP under Linux, we encourage you to read the Linux NET-3 HOWTO, available via anonymous FTP from sunsite.unc.edu. The NET-3 HOWTO is a complete guide to configuring TCP/IP, including Ethernet and SLIP or PPP connections, under Linux. The Linux Ethernet HOWTO is a related document that describes configuration of various Ethernet card drivers for Linux. The Linux Network Administrator's Guide, from the Linux Documentation Project, is also available. See Appendix A for more information on these documents.

Also of interest is the book TCP/IP Network Administration, by Craig Hunt. It contains complete information on using and configuring TCP/IP on UNIX systems.

TCP/IP Hardware requirements.

You can use Linux TCP/IP without any networking hardware at all--configuring ``loopback'' mode allows you to talk to yourself. This is necessary for some applications and games which use the ``loopback'' network

device.

However, if you want to use Linux with an Ethernet TCP/IP network, you need an Ethernet card. Common cards such as the 3com 3c503, HP PCLAN (27245 and 27xxx series), Western Digital WD80x3, and Novell NE2000/NE1000 are supported, as well as many more. See the Linux Ethernet and Hardware HOWTOs for details.

There are a few common situations that you should watch out concerning supported cards: 1) Several cards are support but offer shoddy performance or have other restrictions. Examples are the 3Com 3C501 which works but gives absolutely horrible performance and the Racal-Interlan NI6510 using the am7990 lance chip which doesn't work with more than 16 megabytes of RAM. In the same vein, many cards are NE1000/NE2000 compatible clones and can have various problems. See the Linux Ethernet HOWTO for a more complete discussion of Linux Ethernet hardware compatibility.

Linux also supports SLIP and PPP, which allows you to use a modem to access the Internet over the phone line. In this case, you'll need a modem compatible with your SLIP or PPP server--most servers require a 14.4bps V.32bis modem at a minimum . Performance is greatly improved with a 33.6bps or higher modem.

6.1.1 Configuring TCP/IP on your system.

In this section we're going to discuss how to configure an Ethernet TCP/IP connection on your system. Note that this method should work for many systems, but certainly not all. This discussion should be enough to get you on the right path to configuring the network parameters of your machine, but there are numerous caveats and fine details not mentioned here. We direct you to the Linux Network Administrators' Guide and the NET-3-HOWTO for more information.[gif]

First, we assume that you have a Linux system that has the TCP/IP software installed. This includes basic clients such as telnet and ftp, system administration commands such as ifconfig and route (usually found in /etc), and networking configuration files (such as /etc/hosts). The other Linux-related networking documents described above explain how to go about installing the Linux networking software if you do not have it already.

We also assume that your kernel has been configured and compiled with TCP/IP support enabled. See Section 4.9 for information on compiling your kernel. To enable networking, you must answer ``yes'' to the appropriate questions during the make config step, and rebuild the kernel.

Once this has been done, you must modify a number of configuration files

used by NET-3. For the most part this is a simple procedure. Unfortunately, however, there is wide disagreement between Linux distributions as to where the various TCP/IP configuration files and support programs should go. Much of the time, they can be found in /etc, but in other cases may be found in /usr/etc, /usr/etc/inet, or other bizarre locations. In the worst case you'll have to use the find command to locate the files on your system. Also note that not all distributions keep the NET-3 configuration files and software in the same location--they may be spread across several directories.

The following information applies primarily to Ethernet connections. If you're planning to use SLIP or PPP, read this section to understand the concepts, and follow the more specific instructions in the following sections.

Your network configuration.

Before you can configure TCP/IP, you need to determine the following information about your network setup. In most cases, your local network administrator can provide you with this information.

* IP address. This is the unique machine address in dotted-decimal format. An example is 128.253.153.54. Your network admins will provide you with this number.

  If you're only configuring loopback mode (i.e. no SLIP, no Ethernet card, just TCP/IP connections to your own machine) then your IP address is 127.0.0.1.

* Your network mask (``netmask''). This is a dotted quad, similar to the IP address, which determines which portion of the IP address specifies the subnetwork number, and which portion specifies the host on that subnet. (If you're shaky on these TCP/IP networking terms, we suggest reading some introductory material on network administration.) The network mask is a pattern of bits, which when overlayed onto an address on your network, will tell you which subnet that address lives on. This is very important for routing, and if you find, for example, that you can happily talk to people outside your network, but not to some people within your network, there is a good chance that you have an incorrect mask specified.

  Your network administrators will have chosen the netmask when the network was designed, and therefore they should be able to supply you with the correct mask to use. Most networks are class C subnetworks

which use 255.255.255.0 as their netmask. Class B networks use 255.255.0.0. The NET-3 code will automatically select a mask that assumes no subnetting as a default if you do not specify one.

This applies as well to the loopback port. Since the loopback port's address is always 127.0.0.1, the netmask for this port is always 255.0.0.0. You can either specify this explicitly or rely on the default mask.

* Your network address. This is your IP address masked bitwise-ANDed the netmask. For example, if your netmask is 255.255.255.0, and your IP address is 128.253.154.32, your network address is 128.253.154.0. With a netmask of 255.255.0.0, this would be 128.253.0.0.

  If you're only using loopback, you don't have a network address.

* Your broadcast address. The broadcast address is used to broadcast packets to every machine on your subnet. Therefore, if the host number of machines on your subnet is given by the last byte of the IP address (netmask 255.255.255.0), your broadcast address will be your network address ORed with 0.0.0.255.

  For example, if your IP address is 128.253.154.32, and your netmask is 255.255.255.0, your broadcast address is 128.253.154.255.

  Note that for historical reasons, some networks are setup to use the network address as the broadcast address, if you have any doubt, check with your network administrators. (In many cases, it will suffice to duplicate the network configuration of other machines on your subnet, substituting your own IP address, of course.)

  If you're only using loopback, you don't have a broadcast address.

* Your gateway address. This is the address of the machine which is your ``gateway'' to the outside world (i.e. machines not on your subnet). In many cases the gateway machine has an IP address identical to yours but with a ``.1'' as its host address; e.g., if your IP address is 128.253.154.32, your gateway might be 128.253.154.1. Your network admins will provide you with the IP address of your gateway.

  In fact, you may have multiple gateways. A gateway is simply a machine that lives on two different networks (has IP addresses on different subnets), and routes packets between them. Many networks have a single

gateway to ``the outside world'' (the network directly adjacent to your own), but in some cases you will have multiple gateways--one for each adjacent network.

If you're only using loopback, you don't have a gateway address. The same is true if your network is isolated from all others.

* Your name server address. Most machines on the net have a name server which translates host names into IP addresses for them. Your network admins will tell you the address of your name server. You can also run a server on your own machine by running named, in which case the name server address is 127.0.0.1. Unless you absolutely must run your own name server, we suggest using the one provided to you on the network (if any). Configuration of named is another issue altogether; our priority at this point is to get you talking to the network. You can deal with name resolution issues later.

  If you're only using loopback, you don't have a name server address.

SLIP/PPP users: You may or may not require any of the above information, except for a name server address. When using SLIP, your IP address is usually determined in one of two ways: Either (a) you have a ``static'' IP address, which is the same every time you connect to the network, or (b) you have a ``dynamic'' IP address, which is allocated from a pool available addresses when you connect to the server. In the following section on SLIP configuration this is covered in more detail.

NET-3 supports full routing, multiple routes, subnetworking (at this stage on byte boundaries only), the whole nine yards. The above describes most basic TCP/IP configurations. Yours may be quite different: when in doubt, consult your local network gurus and check out the man pages for route and ifconfig. Configuring TCP/IP networks is very much beyond the scope of this book; the above should be enough to get most people started.

The networking rc files.

rc files are systemwide configuration scripts executed at boot time by init, which start up all of the basic system daemons (such as sendmail, cron, etc.) and configure things such as the network parameters, system host name, and so on. rc files are usually found in the directory /etc/rc.d but on other systems may be in /etc. In general Slackware distributions use the files rc.inet1, etc. in /etc/rc.d whereas the RedHat distributions use a series of directories

Here, we're going to describe the rc files used to configure

TCP/IP. There are two of them: rc.inet1 and rc.inet2. rc.inet1 is used to configure the basic network parameters (such as IP addresses and routing information) and rc.inet2 fires up the TCP/IP daemons (telnetd, ftpd, and so forth).

Many systems combine these two files into one, usually called rc.inet or rc.net. The names given to your rc files doesn't matter, as long as they perform the correct functions and are executed at boot time by init. To ensure this, you may need to edit /etc/inittab and uncomment lines to execute the appropriate rc file(s). In the worst case you will have to create the rc.inet1 and rc.inet2 files from scratch and add entries for them to /etc/inittab.

As we said, rc.inet1 configures the basic network interface. This includes your IP and network address, and the routing table information for your network. The routing tables are used to route outgoing (and incoming) network datagrams to other machines. On most simple configurations, you have three routes: One for sending packets to your own machine, another for sending packets to other machines on your network, and another for sending packets to machines outside of your network (through the gateway machine). Two programs are used to configure these parameters: ifconfig and route. Both of these are usually found in /etc.

ifconfig is used for configuring the network device interface with the parameters that it requires to function, such as the IP address, network mask, broadcast address and the like. Route is used to create and modify entries in the routing table.

For most configurations, an rc.inet1 file that looks like the following should work. You will, of course, have to edit this for your own system. Do not use the sample IP and network addresses listed here for your own system; they correspond to an actual machine on the Internet.

[tscreen6272]

Again, you may have to tweak this file somewhat to get it to work. The above should be sufficient for the majority of simple network configurations, but certainly not all.

rc.inet2 starts up various servers used by the TCP/IP suite. The most important of these is inetd. Inetd sits in the background and listens to various network ports. When a machine tries to make a connection to a certain port (for example, the incoming telnet port), inetd forks off a copy of the appropriate daemon for that port (in the case of the telnet port, inetd starts in.telnetd). This is simpler than running many separate, standalone daemons (e.g., individual copies of telnetd, ftpd, and so

forth)--inetd starts up the daemons only when they are needed.

Syslogd is the system logging daemon--it accumulates log messages from various applications and stores them into log files based on the configuration information in /etc/syslogd.conf. routed is a server used to maintain dynamic routing information. When your system attempts to send packets to another network, it may require additional routing table entries in order to do so. routed takes care of manipulating the routing table without the need for user intervention.

Our example rc.inet2, below, only starts up the bare minimum of servers. There are many other servers as well--many of which have to do with NFS configuration. When attempting to setup TCP/IP on your system, it's usually best to start with a minimal configuration and add more complex pieces (such as NFS) when you have things working.

Note that in the below file, we assume that all of the network daemons are held in /etc. As usual, edit this for your own configuration.

[tscreen6302]

Among the various additional servers that you may want to start in rc.inet2 is named. Named is a name server--it is responsible for translating (local) IP addresses to names, and vice versa. If you don't have a name server elsewhere on the network, or want to provide local machine names to other machines in your domain, it may be necessary to run named. (For most configurations it is not necessary, however.) Named configuration is somewhat complex and requires planning; we refer interested readers to a good book on TCP/IP network administration.

The /etc/hosts file.

/etc/hosts contains a list of IP addresses and the host names that they correspond to. In general, /etc/hosts only contains entries for your local machine, and perhaps other ``important'' machines (such as your name server or gateway). Your local name server will provide address-to-name mappings for other machines on the network, transparently.

For example, if your machine is loomer.vpizza.com with the IP address 128.253.154.32, your /etc/hosts would look like:
[tscreen6318]
If you're only using loopback, the only line in /etc/hosts should be for 127.0.0.1, with both localhost and your host name after it.

The /etc/networks file.

The /etc/networks file lists the names and addresses of your own, and other, networks. It is used by the route command, and allows you to specify a network by name, should you so desire.

Every network you wish to add a route to using the route command (generally called from rc.inet1--see above) must have an entry in /etc/networks.

As an example,
[tscreen6333]

The /etc/host.conf file.

This file is used to specify how your system will resolve host names. It should contain the two lines:
[tscreen6338]
These lines tell the resolve libraries to first check the /etc/hosts file for any names to lookup, and then to ask the name server (if one is present). The multi entry allows you to have multiple IP addresses for a given machine name in /etc/hosts.

The /etc/resolv.conf file.

This file configures the name resolver, specifying the address of your name server (if any) and your domain name. Your domain name is your fully-qualified host name (if you're a registered machine on the Internet, for example), with the host name chopped off. That is, if your full host name is loomer.vpizza.com, your domain name is just vpizza.com.

For example, if your machine is goober.norelco.com, and has a name server at the address 128.253.154.5, your /etc/resolv.conf would look like:
[tscreen6351]
You can specify more than one name server--each must have a nameserver line of its own in resolv.conf.

Setting your host name.

You should set your system host name with the hostname command. This is usually called from /etc/rc or /etc/rc.local; simply search your system rc files to determine where it is invoked. For example, if your (full) host name is loomer.vpizza.com, edit the appropriate rc file to execute the command:
[tscreen6364]
Note that the hostname executable may not be found in /bin on your system.

Trying it out.

Once you have all of these files set up, you should be able to reboot your new kernel and attempt to use the network. There are many places where things can go wrong, so it's a good idea to test individual aspects of the network configuration (e.g., it's probably not a good idea to test your network configuration by firing up Mosaic over a network-based X connection).

You can use the netstat command to display your routing tables; this is usually the source of the most trouble. The netstat man page describes the exact syntax of this command in detail. In order to test network connectivity, we suggest using a client such as telnet to connect to machines both on your local subnetwork and external networks. This will help to narrow down the source of the problem. (For example, if you're unable to connect to local machines, but can connect to machines on other networks, more than likely there is a problem with your netmask and routing table configuration). You can also invoke the route command directly (as root) to play with the entries in your routing table.

You should also test network connectivity by specifying IP addresses directly, instead of host names. For example, if you have problems with the command
[tscreen6377]
the cause may be incorrect name server configuration. Try using the actual IP address of the machine in question; if that works, then you know your basic network setup is (more than likely) correct, and the problem lies in your specification of the name server address.

Debugging network configurations can be a difficult task, and we can't begin to cover it here. If you are unable to get help from a local guru we strongly suggest reading the Linux Network Administrators' Guide from the LDP.

6.1.2 SLIP configuration.

SLIP (Serial Line Internet Protocol) allows you to use TCP/IP over a serial line, be that a phone line, with a dialup modem, or a leased asynchronous line of some sort. Of course, to use SLIP you'll need access to a dial-in SLIP server in your area. Many universities and businesses provide SLIP access for a modest fee.

There are two major SLIP-related programs available--dip and slattach. Both of these programs are used to initiate a SLIP connection over a serial device. It is necessary to use one of these programs in order to enable SLIP--it will not suffice to dial up the SLIP server (with a communications

program such as kermit) and issue ifconfig and route commands. This is because dip and slattach issue a special ioctl() system call to seize control of the serial device to be used as a SLIP interface.

dip can be used to dial up a SLIP server, do some handshaking to login to the server (exchanging your username and password, for example) and then initiate the SLIP connection over the open serial line. slattach, on the other hand, does very little other than grab the serial device for use by SLIP. It is useful if you have a permanent line to your SLIP server and no modem dialup or handshaking is necessary to initiate the connection. Most dialup SLIP users should use dip, on the other hand.

dip can also be used to configure your Linux system as a SLIP server, where other machines can dial into your own and connect to the network through a secondary Ethernet connection on your machine. See the documentation and man pages for dip for more information on this procedure.

SLIP is quite unlike Ethernet, in that there are only two machines on the ``network''--the SLIP host (that's you) and the SLIP server. For this reason, SLIP is often referred to as a ``point-to-point'' connection. A generalization of this idea, known as PPP (Point to Point Protocol) has also been implemented for Linux.

When you initiate a connection to a SLIP server, the SLIP server will give you an IP address based on (usually) one of two methods. Some SLIP servers allocate ``static'' IP addresses--in which case your IP address will be the same every time you connect to the server. However, many SLIP servers allocate IP addresses dynamically--in which case you receive a different IP address each time you connect. In general, the SLIP server will print the values of your IP and gateway addresses when you connect. dip is capable of reading these values from the output of the SLIP server login session and using them to configure the SLIP device.

Essentially, configuring a SLIP connection is just like configuring for loopback or ethernet. The main differences are discussed below. Read the previous section on configuring the basic TCP/IP files, and apply the changes described below.

Static IP address SLIP connections using dip.

If you are using a static-allocation SLIP server, you may want to include entries for your IP address and host name in /etc/hosts. Also, configure these files listed in the above section: rc.inet2, host.conf, and resolv.conf.

Also, configure rc.inet1, as described above. However, you only want to

execute ifconfig and route commands for the loopback device. If you use dip to connect to the SLIP server, it will execute the appropriate ifconfig and route commands for the SLIP device for you. (If you're using slattach, on the other hand, you will need to include ifconfig/route commands in rc.inet1 for the SLIP device--see below.)

dip should configure your routing tables appropriately for the SLIP connection when you connect. In some cases, however, dip's behavior may not be correct for your configuration, and you'll have to run ifconfig or route commands by hand after connecting to the server with dip (this is most easily done from within a shell script that runs dip and immediately executes the appropriate configuration commands). Your gateway is, in most cases, the address of the SLIP server. You may know this address before hand, or the gateway address will be printed by the SLIP server when you connect. Your dip chat script (described below) can obtain this information from the SLIP server.

ifconfig may require use of the pointopoint argument, if dip doesn't configure the interface correctly. For example, if your SLIP server address is 128.253.154.2, and your IP address is 128.253.154.32, you may need to run the command
[tscreen6437]
as root, after connecting with dip. The man pages for ifconfig will come in handy.

Note that SLIP device names used with the ifconfig and route commands are sl0, sl1 and so on (as opposed to eth0, eth1, etc. for Ethernet devices).

In Section 6.1.2, below, we explain how to configure dip to connect to the SLIP server.

Static IP address SLIP connections using slattach.

If you have a leased line or cable running directly to your SLIP server, then there is no need to use dip to initiate a connection. slattach can be used to configure the SLIP device instead.

In this case, your /etc/rc.inet1 file should look something like the following:
[tscreen6459]

slattach allocates the first unallocated SLIP device (sl0, sl1, etc.) to the serial line specified.

Note that the first parameter to slattach is the SLIP protocol to use. At

present the only valid values are slip and cslip. Slip is regular SLIP, as you would expect, and cslip is SLIP with datagram header compression. In most cases you should use cslip; however, if you seem to be having problems with this, try slip.

If you have more than one SLIP interface then you will have routing considerations to make. You will have to decide what routes to add, and those decisions can only be made on the basis of the actual layout of your network connections. A book on TCP/IP network configuration, as well as the man pages to route, will be of use.

Dynamic IP address SLIP connections using dip.

If your SLIP server allocates an IP address dynamically, then you certainly don't know your address in advance--therefore, you can't include an entry for it in /etc/hosts. (You should, however, include an entry for your host with the loopback address, 127.0.0.1.)

Many SLIP servers print your IP address (as well as the server's address) when you connect. For example, one type of SLIP server prints a string such as,
[tscreen6477]
dip can capture these numbers from the output of the server and use them to configure the SLIP device.

See page [gif] , above, for information on configuring your various TCP/IP

files for use with SLIP. Below, we explain how to configure dip to connect to the SLIP server.

Using dip.

dip can simplify the process of connecting to a SLIP server, logging in, and configuring the SLIP device. Unless you have a leased line running to your SLIP server, dip is the way to go.

To use dip, you'll need to write a ``chat script'' which contains a list of commands used to communicate with the SLIP server at login time. These commands can automatically send your user name/password to the server, as well as get information on your IP address from the server.

Here is an example dip chat script, for use with a dynamic IP address server. For static servers, you will need to set the variables $local and $remote to the values of your local IP address and server IP address,

respectively, at the top of the script. See the dip man page for details.

[tscreen6494]

dip automatically executes ifconfig and route commands based on the values of the variables $local and $remote. Here, those variables are assigned using the get...remote command, which obtains text from the SLIP server and assigns it to the named variable.

If the ifconfig and route commands that dip runs for you don't work, you can either run the correct commands in a shell script after executing dip, or modify the source for dip itself. Running dip with the -v option will print debugging information while the connection is being set up, which should help you to determine where things might be going awry.

Now, in order to run dip and open the SLIP connection, you can use a command such as:
[tscreen6510]
Where the various dip files, and the chat script (mychat.dip), are stored in /etc/dip.

The above discussion should be enough to get you well on your way to talking to the network, either via Ethernet or SLIP. Again, we strongly suggest looking into a book on TCP/IP network configuration, especially if your network has any special routing considerations, other than those mentioned here.

Dial-up networking and PPP.

Linux supports a full implementation of the PPP (Point-to-Point) networking protocols. PPP is a mechanism for creating and running IP (the Internet Protocol) and other network protocols over a serial connection (using a null-modem cable), over a telnet established link or a link made using modems and telephone lines (and of course using digital lines such as ISDN). This section will only cover configuring PPP as a client connecting via an analog modem to a remote machine that provides PPP dialup service.

For complete information on setting up PPP under Linux, we encourage you to read the Linux PPP HOWTO, available via anonymous FTP from sunsite.unc.edu. The PPP HOWTO is a complete guide to configuring PPP, including modem, ISDN and null-modem cables, under Linux. Much of the information in this section was gleaned from this document. The Linux Network Administrator's Guide, from the Linux Documentation Project, is also available. See Appendix A for more information on these documents.

6.2.1 What you need to get started.

We assume that your kernel has been configured and compiled with TCP/IP support enabled. See Section 4.9 for information on compiling your kernel. To enable networking, you must answer ``yes'' to the appropriate questions during the make config step, and rebuild the kernel. We also assume that ppp has been compiled and installed on your system as well. We assume that you are using a Linux 1.2.x kernel with the PPP 2.1.2 software or Linux 1.3.X/2.0.x and PPP 2.2.0. At the time of writing, the latest official version of PPP available for Linux is ppp-2.2f. Please see the kerneld mini-HOWTO if you plan to use modules to load ppp into your kernel. It is highly recommended that you use a version of the Linux kernel and the appropriate PPP version that are known to be stable together.

You should also read

* the documentation that comes with the PPP package;
* the pppd and chat man pages; (use man chat and man pppd to explore these)
* the Linux Network Administration Guide (NAG);
* the Net-2/3 HOWTO;
* Linux kernel documentation installed in /usr/src/linux/Documentation when you install the Linux source code;
* The modem setup information page--see Modem Setup Information (http://www.in.net/info/modems/index.html)
* The excellent Unix/Linux books published by O'Reilly and Associates. See (O'Reilly and Associates On-Line catalog (http://www.ora.com/). If you are new to Unix/Linux, run (don't walk) to your nearest computer book shop and invest in a number of these immediately!
* The PPP-FAQ maintained by Al Longyear, available from (ftp://sunsite.unc.edu/pub/Linux/docs/faqs; see Appendix B). This contains a great deal of useful information in question/answer format that is very useful when working out why PPP is not working (properly).

6.2.2 An overview of the steps involved.

There are several steps to setting up your system to use PPP. We recommend that you read through all of these steps thoroughly before attempting to actually bring up a PPP connection. Each of these steps will be discussed in detail later.

1. Make sure that TCP/IP support is compiled into your kernel.
2. Make sure that PPP support is compiled into your kernel either statically or as a loadable module.
3. Make sure that PPP software is compiled and installed on your systems.

4. Make sure that you have a modem configured and installed/attached to your computer and that you know which serial port