

Introduction to Networking with Unix

Scott Dickson
Penn State University
Center for Academic Computing
+1 814 865 0829
dickson@iris.psu.edu

24 Oct 1992

1 Introduction

In this document, we will cover a number of introductory issues regarding networking on Unix systems. Since most networking on modern Unix systems is done using TCP/IP, we will concentrate on TCP/IP based communication networks and not concern ourselves with other types of networks. The basic issues that will be covered here are:

- Overview of TCP/IP
- Introduction to the Structure of the PSU Backbone
- Introduction to the Structure of the Internet
- Identifying Networked Computers
- The Domain Naming System
- Associating Host Names with Internet Addresses
- Network Login Access
- Network File Transfer
- Remote Execution of Commands
- Network Queries
- Electronic Mail
- Network News
- Getting More Information

2 Overview of TCP/IP

TCP/IP is not a type of network, but rather it is a part of a suite of communication protocols. TCP is only one of several different types of protocols that are used in this suite. The actual details of how TCP/IP works are far beyond the scope of this document. For a good introduction to TCP/IP see [2].

For the purposes of this document, it is sufficient to say that TCP is a packet-based communications protocol that provides applications with a reliable stream of bytes. This means that messages between machines are broken into packets and that the series of packets is transmitted to the intended recipient. Each packet contains information specifying its origin and its destination as well as whatever data a program chooses to place in it. Packet sizes vary as does the amount of data that is included in each packet. This might range from a single keystroke on the keyboard to a fairly substantial portion of a file being transferred. That TCP provides a reliable stream of bytes means that it is known whether or not data sent from one machine to another is actually received. Along with a mechanism for acknowledging the receipt of data there is a mechanism for attempting to resend data when the sender is notified that a packet has been lost in transit. In addition, there is no guarantee of the order in which packets will arrive, so TCP takes care of reordering those packets so that they will actually be received in the order that they were sent.

3 Overview of the Penn State - University Park Data Backbone

A large-scale fiber-optic network is being installed on the Penn State University Park campus. Plans are in place to connect all major buildings on the campus via this network. Work is progressing with many buildings already being connected to the network. The Office of Telecommunications supervises this network, referred to as the data backbone. This network is not based on Ethernet, but rather on other protocols known as FDDI and ProNET. In each building that is connected to the backbone, there is a gateway connecting the campus-wide network to the building's network. The network within a building could be Ethernet, IBM Token Ring, or some other type of network. Ethernet and Token Ring are the most commonly used types of network at Penn State for Unix systems, and both can support TCP/IP. The gateway is a special purpose computer with connections to both networks and is responsible for making a physical connection. It is responsible for translating between the protocol used on the backbone and the network protocol used in the local network, and is responsible for providing routing service. This means that it takes packets from the local network that are destined for locations not on the local network and sends them on their way, via the backbone.

The Penn State data backbone, then, provides a high-speed connection between different local networks at Penn State as well as a connection to the rest of the Internet.

For a complete description of the Penn State Data Backbone, its general use, and policies pertaining to its use and connection, see [10]. This document is available via anonymous ftp (which will be discussed later in this handout) from `ftp.otc.psu.edu`.

4 Introduction to the Internet

Machines can be connected together to form local networks. At Penn State, these local networks are all connected to each other via the campus backbone. On a larger scale, the networking world is made up of many different wide-area networks, each very much interconnected with the others. For this reason, the collection of all interconnected networks is known as the Internet.

Penn State is connected to the rest of the world through its connections to two larger scale networks: PREPNet and JvNCNet. PREPNet, the Pennsylvania Research & Economic Partnership Network, is a network connecting a number of colleges and universities in Pennsylvania with various businesses and research organizations in the State. This network connects to the larger NSFNet at the Pittsburgh Supercomputing Center. JvNCNet is a network connecting the members of the Consortium for Supercomputing, a group of universities joined by the National Science Foundation (NSF) to govern the

John von Neumann Center for Supercomputing. JvNCNet connects to NSFNet at JvNC in Princeton, NJ. NSFNet is, in turn, a larger network that spans the continental United States with multiple, high-speed connections. NSFNet, then, is only one of several large networks in the US. Other countries have their own national networks as well, and sites on most of these networks can connect with sites on any of the other networks, in their own country, or in some other. The Internet is a large collection of networks of different sizes, all cooperating for maximum connectivity. For a good description of the Internet, see [2].

5 Identifying Networked Computers on the Internet

There are at least three ways to uniquely identify any computer connected to a network. The most hardware specific is to use the serial number (or address) of the piece of hardware used to connect to the physical network. (For the rest of this document, network connections will be assumed to use Ethernet) In the case of Ethernet, each interface has a unique six byte code associated with it. Since this code is long, cryptic, and doesn't contain a lot of information to users, it is not generally used by people to identify machines. However, it is used by the physical network to keep track of systems connected to a local area network. As an example, the Ethernet address for one of the Sun Microsystems machines in the Center for Academic Computing is `08:00:20:00:b1:f1`. Each machine has at least one Ethernet address. The addresses of machines within a group have nothing to do with each other; these addresses are assigned by the manufacturer when the network interface cards are manufactured. A second method of identifying machines on the Internet is through the use of Internet (or IP) Addresses. These are four byte numbers, broken into the individual bytes (as the Ethernet addresses are) with the four bytes separated by dots (.). However, addresses of related machines are easily associated. The Internet Address for the same machine used above is `128.118.58.23`. Internet addresses contain more information than Ethernet addresses and are much easier to remember. The Internet address is made up of a network portion and a local portion and gives information about the organization to which a machine is attached.

Internet network addresses are assigned by the Network Information Center (the NIC) and are divided into three classes. Class A networks have addresses where the first byte is less than 128. The organization to which a Class A network is assigned is free to assign all of the numbers for the remaining three bytes. This allows them to have a large number of systems connected within their organizational network. Examples of this kind of network are the networks of large corporations (for example, Xerox) and large universities (for example, MIT). Class B networks have a first byte between 128 and 191. In addition, the second byte of the Internet address is also assigned by the NIC in these cases, leaving the organizations the last two bytes to assign internally. A Class C network has a first byte between 192 and 239, with the second and third bytes assigned by the NIC. These are used for smaller organizations.

Penn State has at least two Class B networks, with the first two bytes of the network addresses being `128.118` and `146.186`. In addition, the Computer Science Department at Penn State has its own Class B network with the address `130.203` and the Hershey Medical Center has a Class B network with the address `150.231`.

At Penn State, a technique known as *subnetting* is used to assign networks and network numbers efficiently. When a network is subnetted, part of the local portion the the address is used as if it were a part of the network address. This means that a single large network may be broken down into a number of separate, smaller networks. This provides a great advantage in network administration and assignment. Generally, subnets are assigned to departments, buildings, labs, or other sensibly defined units. Table 1 lists some of the subnets in use at Penn State.

The software used on the Internet uses internet addresses to uniquely identify machines on the network. Rarely will users need to use the internet addresses; generally, another system for identifying machines is

Location	Network Number(s)
Computer Building	128.118.2, 56-59
Center for Academic Computing Microcomputer Labs	128.118.23, 37, 51
Willard Building	128.118.3, 39
Telecommunications Building	128.118.4, 25, 46, 100
Engineering Computer Lab	128.118.5
Davey Lab, Pond Lab	128.118.30, 49, 84
College of Earth & Mineral Sciences	128.118.41
Hammond Building SGI Lab	146.186.125
Computer Science Department	130.203.1-254

Table 1: Some Penn State Subnets

used.

The third and most commonly used way of identifying machines on the network is to name each machine. Names are given to the machines by their owners, and these names are registered with the local networking authorities. Each machine, when it is registered, is assigned its network (or internet) address. There are as many ways for selecting names for computers as there are owners of computers. While the names may vary, and may or may not convey the actual use of the machine, they are much more useful and easy to remember than addresses. Generally, the Domain Naming System is used to provide the necessary translation between names and IP addresses. It is described in the next section.

6 The Domain Naming System

Just as networks are arranged in a hierarchical system, names for computers are also arranged hierarchically. Each level of this hierarchy is known as a domain. Conceptually, there can be as many levels to the hierarchy as needed.

The designers of the domain-naming system initiated several general categories of names as top-level domain names so that each could accommodate a variety of organizations. The current top-level domains registered with the DDN Network Information Center are **COM**, **EDU**, **GOV**, **MIL**, **NET**, and **ORG**, plus a number of top-level ISO country domains.

- **COM** is meant to incorporate subdomains of companies and businesses, eg. **Xerox.com** and **IBM.com**.
- **EDU** was initiated to accommodate subdomains set up by universities and other educational institutions, eg. **psu.edu** and **mit.edu**.
- **GOV** exists to act as parent domain for subdomains set up by government agencies, eg. **nsf.gov** and **nist.gov**.
- **MIL** was initiated to act as parent to subdomains that are developed by military organizations, eg. **eglin.af.mil**.
- **NET** was introduced as a parent domain for various network-type organizations. Organizations that belong within this top-level domain are generic or network-specific, such as network service centers and consortia. **NET** also encompasses network management-related organizations, such as information centers and operations centers, eg. **uunet.uu.net**, and **sh.cs.net**.

- **ORG** exists as a parent to subdomains that do not clearly fall within the other top-level domains. This may include technical-support groups, professional societies, or similar organizations, eg. `jvnca.csc.org`.

It is easy to see that Penn State will fall into the **EDU** top-level domain. At the next level, the organizations to which network numbers have been assigned select a domain name. At Penn State, the local domain name is **PSU**. Within this domain, there exist other local sub-domains as needed by different organizations, eg. `cs.psu.edu` and `math.psu.edu`.

Each machine is given a local name that is selected by its owner. This name must be unique within the local domain. The domain name is then appended to the hostname to get the fully-qualified hostname. Since names are unique locally, appending the domain name, which is also unique, guarantees that there will be no other machine on the network with the same fully-qualified host name.

Examples will help to show how domain names work. For example, the machine `psusun01.psu.edu` has a local hostname of `psusun01`, and is a part of the `psu.edu` domain. Domain names can be complex as well; `osteocyber.ortho.hmc.psu.edu` is a machine at the Hershey Medical Center. Domain names are generally given to administrative units such as `hmc` for Hershey Medical Center, `cs` for the Computer Science Department, etc. Hosts may also be part of no local subdomain, and are then just members of the local domain.

7 Associating Machine Names and Internet Addresses

The software used on the Internet identifies computers via their Internet Address (or IP number), while people tend to use host names. Therefore, a means for mapping between these two forms of naming is necessary. Usually, the mapping needs to go from a name to a number but sometimes a conversion from a number to a name is also needed. There are two generally used ways to do this.

The first is to use a static table that contains a list of all of the other machines that a particular computer knows about. In Unix, this file is called `/etc/hosts` and is maintained by the system administrator. Using a static table like this generally limits the number of hosts that may be easily accessible, since the table is finite, incomplete, and never up-to-date. Hosts are added to the network too fast to have accurate host tables at all times, and the number of hosts is prohibitively high to keep a list of them all. For a small, personal machine in an office or lab, static tables may be sufficient. However, for machines that are to be used by a number of people, this usually isn't satisfactory since the sites they may be trying to connect to change rapidly and might be very numerous.

Figure 1 give an example of the format of the `/etc/hosts` file. On each line, there is an IP address and a list of names by which that host can be addressed. The first name in the list is the real name of the host. Sometimes this should be the fully-qualified name of the host, but on other systems, this should be the short name of the local machine. Following the real name, there is a list of other names that the machine might be called. These are generally used by programs and not by people, for special purposes.

A much better alternative to static tables has been developed for dynamic resolution of names into addresses. BIND, the Berkeley Internet Name Domain server, actually queries sites on the network to resolve a name into an address. With BIND, a system of nameservers is used. A nameserver is a process that knows how to convert names into addresses for a particular domain as well as what other nameserver to query in the next level of the domain hierarchy. When a program tries to resolve a hostname into a number, it consults its local nameserver for that address. The local nameserver will either reply with an address or it will ask another nameserver that it thinks might know the answer. In this manner, requests

```

#
# Loopback Interface
#
127.0.0.1      localhost
128.118.56.2   psuvm.psu.edu psuvm vm
128.118.58.6   psusun01.psu.edu psusun01 sun1 wilbur
192.48.96.2    uunet.uu.net

```

Figure 1: An Example of the `/etc/hosts` file

for addresses are distributed, and each nameserver only has to know its own local hosts. When a new host is added to the network, all other machines can immediately find its address, since eventually the local nameserver for it will be consulted and will reply with the address.

The local machine must be told what nameserver to consult as well as alternate nameservers to consult if the primary nameserver is unable to find an answer. This is controlled in the file `/etc/resolv.conf` that is created and maintained by the system administrator of the local machine. The format of this file can be found in the online manual pages.

A facility exists for users to interface with BIND directly. This facility, available with the command `nslookup`, allows the user to interactively specify not only a host name to resolve but also what nameserver to use. With `nslookup`, it is also possible to find other information, such as a list of all machines within a domain, what kind of machine a particular host is, etc. Figure 2 shows a sample session using `nslookup`.

8 Network Login Access

Now that some introduction has been presented explaining how the Internet works, the mechanisms for identifying machines, and how a user can specify a machine, let us move on to a discussion of the services available on the network that one might want to access. There are a wide variety of different services available through the Internet, but a few major types of services dominate most people's use of the network. These services are remote login to machines, file transfer, remote execution of commands, network queries, and mail. Mail is not covered in this document. We will turn our attention first to remote login facilities.

Two different programs are used for remote login on the Internet: `rlogin` is used to log into another machine that is running Unix and whose name is known; `telnet` is used to log into a machine running Unix or some other operating system. Additionally, with `telnet`, it is possible to connect to a machine using just the IP address, if it is known. We will look at each of these programs in turn, but we will first look at the issue of trust between computer systems.

Some systems trust certain users for remote login. This means that the trusted users, on trusted hosts, are able to login, execute commands, and copy files without having to give a password for verification each time. This facility is used by `rlogin` and is controlled by two different tables. The first, `/etc/hosts.equiv`, is a systemwide table of trusted machines. Any user on a machine in this list may log into any other machine in the list without specifying a password. This is useful in a departmental or laboratory situation, when there are a number of systems that share resources. The `/etc/hosts.equiv` table is maintained by the system administrator and should contain only machines that are truly trusted and under the same local control. The second method for establishing trust is the `.rhosts` file. This file, located in a user's home directory, lists hostnames and account names from which the user grants unchallenged access to this

```

psusun01.1: nslookup
Default Server: psusun01.psu.edu
Address: 128.118.58.6

> expo.lcs.mit.edu
Server: psusun01.psu.edu
Address: 128.118.58.6

Name: expo.lcs.mit.edu
Address: 18.30.0.212

> help
Commands: (identifiers are shown in uppercase, [] means optional)

NAME- print info about the host/domain NAME using default server
NAME1 NAME2- as above, but use NAME2 as server
help - print help information
exit - exit the program
set OPTION- set an option
server NAME- set default server to NAME, using current default server
ls NAME- list the domain NAME
> ls chem.psu.edu
[psusun01.psu.edu]
Host or domain name      Internet address
chem                     server = isengard.cs.psu.edu
isengard                 128.118.6.30
isengard                 130.203.1.2
chem                     server = psusun01.psu.edu
psusun01                 128.118.58.6
flo                      128.118.30.201
ra                       128.118.30.102
was                      128.118.30.101
isaac                   128.118.30.110
felix                   128.118.30.100
bigapl                  128.118.30.52
vixen                   128.118.30.31
dali                     128.118.30.112
kepler                  128.118.30.104
retina                  128.118.30.113
nowhere                 128.118.30.103
stella                  128.118.30.111
galileo                 128.118.30.106
> exit
psusun01.2:

```

Figure 2: Sample Session Using nslookup

```
psusun01.1> rlogin euler.psu.edu
Last login: Mon Nov 20 08:54:30 from psusun01.psu.edu
SunOS Release 4.0.3 (Euler) #1: Thu Oct 12 12:28:39 EDT 1989
euler.1: logout
Connection closed.
```

Figure 3: `rlogin`: Logging in from a trusted account

```
psusun01.3> rlogin psuvax1.cs.psu.edu
Password:
Last login: Fri Dec 1 14:11:07 from psusun01.psu.edu
SunOS Release 4.0.3 (Psvax1) #7: Mon Oct 30 16:06:44 EST 1989
Fri Dec 1 14:12:24 EST 1989
psuvax1.2: logout
Connection closed.
```

Figure 4: `rlogin`: Logging in from a non-trusted account

account. This table is maintained by the user and is a convenience feature. Again, it should be used with care.

8.1 `rlogin`: Remote Login to Other Unix Systems

`rlogin` allows you to log into another machine running Unix from a machine running Unix. The basic syntax of the command for `rlogin` is

```
rlogin machinename [-l username ]
```

If a connection is established, you will be prompted for a password (if an entry is not found in `/etc/hosts.equiv` or in the `.rhosts` file on the target system), and you will be logged on. For more detailed information on `rlogin`, see the online manual pages, which describe the details of the syntax of `rlogin` as well as special sequences for terminating and disconnecting a remote session. Figures 3-5 show several examples of the use of `rlogin`.

```
psusun01.3> rlogin psuvax1.cs.psu.edu -l dickson
Password:
Last login: Fri Dec 1 14:11:07 from psusun01.psu.edu
SunOS Release 4.0.3 (Psvax1) #7: Mon Oct 30 16:06:44 EST 1989
Fri Dec 1 14:12:24 EST 1989
psuvax1.2: logout
Connection closed.
```

Figure 5: `rlogin`: Logging in using a different username

```
psusun01.1> telnet psusun03 Trying 128.118.58.9 ... Connected to
psusun03.psu.edu. Escape character is '^']'.
```

```
SunOS UNIX (psusun03)
```

```
login: dickson
```

```
Password:
```

```
Last login: Wed Nov 29 15:09:22 on console
```

```
SunOS Release 4.0 (PSUSUN03) #7: Sun Jul 16 02:47:40 EDT 1989
```

```
Fri Dec 1 14:34:06 EST 1989
```

```
psusun03.2: logout
```

```
Connection closed by foreign host.
```

Figure 6: `telnet`: Connecting to a known host

```
psusun01.3> telnet 128.174.5.101
Trying 128.174.5.101 ...
Connected to 128.174.5.101.
Escape character is '^']'.
```

```
This is NCSAVMSB (VAX-11/785)
```

```
Username:
```

```
Connection closed by foreign host.
```

Figure 7: `telnet`: Connecting to an unknown host

8.2 `telnet`: Remote Login to Other Systems

`telnet` is the second major way of establishing a remote login session. It is particularly well suited for making the connection between systems that are all connected to the Internet but that use dissimilar operating systems. It is also useful for making the connection to hosts for whom a name and an IP address are known by the user but for which the system is unable to resolve an address. In some cases, a special version of `telnet` is necessary to connect to some operating systems, such as IBM's VM family. Examples follow of establishing connections using `telnet` and `tn3270`. For more information as to the details and options available with `telnet`, consult the online manual pages.

`telnet` is also useful in establishing other sorts of general network connections to access other services at other sites. More will be said about using `telnet` in this fashion later on.

9 Network File Transfer

While it is very convenient to be able to log into a remote system, it is also necessary to be able to transfer files between systems. We will look at two different methods of doing this. Similar to `rlogin` and `telnet`, `ftp` and `rcp` are both used for file transfer: `ftp` between machines of many different architectures and

```
psusun01.4> telnet JvNcc.csc.org
Trying 128.121.50.3 ...
Connected to JvNcc.csc.org.
Escape character is '^]'.
```

John von Neumann National Supercomputer Center VAX/VMS 8600

```
Username:
Connection closed by foreign host.
```

Figure 8: `telnet`: Connecting to a non-Unix host

```
psusun01.5> tn3270 psuvm
Trying...
Connected to psuvm.psu.edu.
VM/XA SP ONLINE

CONNECT= 00:00:02 VIRTCPU= 000:00.00 TOTCPU= 000:00.00
LOGOFF AT 15:07:37 EST FRIDAY 12/01/89
Connection closed by foreign host.
```

Figure 9: `tn3270`: Connecting to a VM system

operating systems, `rcp` between Unix machines and trusted users.

9.1 `ftp`: File Transfer Protocol

`ftp`, the File Transfer Protocol, is a standard method for transferring files between computer systems via the Internet. It can be used between machines on the same local network or across continents. It is used on a variety of different types of machines, from microcomputers to workstations to mainframes to supercomputers. `ftp` is supported on a host of operating systems from MS-DOS to VM/XA to Unix to VMS. Security within `ftp` is handled by requiring the user to give a userid and password that are valid on the remote machine.

Operation of `ftp` is fairly simple and straightforward. The command

`ftp hostname`

is used to establish a connection to a remote system. The remote machine will then prompt for a userid and a password. Once these are validated, you are then able to issue `ftp` commands to transfer files to and from the remote system. Differences in filename styles, directory styles, etc. are for the most part taken care of by `ftp` by using a standard, intermediate, language for its communication.

The following shows an example session using `ftp` to transfer files to and from a remote host. Comments and additional explanations are interspersed in the example. Comments are shown in this font; actual text of the example will be in **this font**. A wide variety of options and other commands are available. For more information on these, refer to the online manual pages.

Sample Session Using ftp

Establish a connection to the remote host

```
jellyroll.1: ftp shire.cs.psu.edu
Connected to shire.cs.psu.edu.
220 shire FTP server (SunOS 4.1) ready.
```

Enter your userid and password to identify yourself.

```
Name (shire.cs.psu.edu:dickson): dickson
331 Password required for dickson.
Password:
230 User dickson logged in.
```

See what files are available on the remote system.

```
ftp> dir
200 PORT command successful.
150 ASCII data connection for /bin/ls (128.118.58.171,2577) (0
bytes).
total 92
drwxr-sr-x  2 dickson  cac           512 Feb 16  1989 bin
-rw-----  1 dickson  cac           830 Dec 13  1988 mbox
drwxr-xr-x  2 dickson  cac           512 Aug 15  15:06 news
drwxr-xr-x  3 dickson  cac           512 Nov 13  09:33 pic448
226 ASCII Transfer complete.
1838 bytes received in 0.82 seconds (2.19 Kbytes/s)
```

Transfer a file from the remote host to the local host, with a new name.

```
ftp> get mbox shire.mbox
200 PORT command successful.
150 ASCII data connection for mbox (128.118.58.171,2578) (830 bytes).
226 ASCII Transfer complete.
local: shire.mbox remote: mbox
853 bytes received in 0.02 seconds (51.26 Kbytes/s)
```

Check to see what files are available locally. Under Unix, the escape character ! allows you to execute commands on the local machine without leaving ftp.

```
ftp> !ls
Apps                Programming          shire.mbox
JGADemo             Unix                typescript
Library             WNDictionary.wndict
Mailboxes           ftp
```

Transfer a file to the remote system, with a new name

```

ftp> put .login jellyroll.login
200 PORT command successful.
150 ASCII data connection for jellyroll.login (128.118.58.171,2579).
226 ASCII Transfer complete.
local: .login remote: jellyroll.login
449 bytes sent in 0.03 seconds (12.55 Kbytes/s)

```

See what other commands are available in this version of ftp.

```

ftp> help
Commands may be abbreviated.  Commands are:
!          cr          macdef      proxy      send
$          delete       mdelete    sendport   status
account   debug          mdir       put        struct
append    dir            mget       pwd        sunique
ascii     disconnect    mkdir      quit       tenex
bell      form          mls        quote      trace
binary    get           mode       recv       type
bye       glob         mput       remotehelp user
case      hash         nmap       rename     verbose
cd        help         ntrans     reset      ?
cdup      lcd          open       rmdir
close     ls           prompt     runique

```

Exit from ftp.

```

ftp> quit
221 Goodbye.
jellyroll.2:

```

9.2 Anonymous ftp: A Special Case

Sometimes a system will have publicly accessible files and information that its administrators want to distribute freely. **ftp** contains provisions for this, known as *anonymous ftp*. In an anonymous ftp session, you connect to the remote host, just as in any other **ftp** session, however you use a special userid, **anonymous**, to log in. Instead of a password, you enter your e-mail address (userid@host). You are then able to read a limited set of the files on the remote system. You generally are not able to write any files on that system, but occasionally you may be able to. You are unable to access any files, other than the ones that the system administrator has made publicly accessible. The anonymous **ftp** facility is widely used to distribute different types of documents as well as freely-distributable software. Some computer vendors, such as Sun Microsystems, make patches and updates to their products available via anonymous **ftp**. In the following section, a sample anonymous **ftp** session is shown. In this session, we **ftp** to the Office of Telecommunications. On one of the OTC machines, there is a collection of documents describing the network. As an exercise, you might want to try to **ftp** there and collect the documents in the directory `/pub/otcdoc` on the host `ftp.otc.psu.edu`.

One point that often brings about some degree of confusion relating to anonymous **ftp**: many sites have given the system that they choose to make available for anonymous **ftp** the name **ftp**. Hence, the system that you want to connect to may be named `ftp.foo.bar`. Remember that this is still the only the name of the machine and that you still have to give to **ftp** command. This is shown in the example below.

Many sites that maintain anonymous ftp archives set up the filesystems in easy to navigate fashions. Often each directory will have a file called **README** or something similar that will describe the contents of that directory. Also, files containing indexes of what is available often exist as well. Many times, a file named **ls-lR.Z** also exist. This file contains the compressed output from a long-style, recursive directory listing of the system. These can be downloaded and used rather than spending lots of time getting lists of the directories. Often these lists are generated automatically and are up-to-date as of the start of the day.

Sample Session using Anonymous ftp

```
jellyroll.2: ftp ftp.otc.psu.edu.
220 otc1 FTP server (SunOS 4.0) ready.

Log in as anonymous, send your userid and hostname as the password.

Name (otc1.psu.edu:dickson): anonymous
331 Guest login ok, send ident as password.
Password:
230 Guest login ok, access restrictions apply.
ftp> dir
200 PORT command successful.
150 ASCII data connection for /bin/ls (128.118.58.171,2582).
total 5
d--x--s--x  2 0      0      512 Aug 29 16:56 bin
d--x--s--x  2 0      0      512 Aug 30 18:19 dev
d--x--s--x  2 0      0      512 Aug 29 16:59 etc
dr-xrwsrwx  7 109    0      512 Nov 27 14:50 pub
d--x--s--x  3 0      0      512 Aug 29 17:00 usr
226 ASCII Transfer complete.
304 bytes received in 0.09 seconds (3.3 Kbytes/s)
```

The /pub directory is generally where publicly accessible files are kept.

```
ftp> cd pub
250 CWD command successful.
ftp> dir
200 PORT command successful.
150 ASCII data connection for /bin/ls (128.118.58.171,2583).
total 5
drwxr-xr-x  2 103    0      512 Sep 24 21:09 ftpsoft
drwxr-xr-x  2 102    0     1024 Nov  9 22:02 novell
drwxr-xr-x  2 103    0      512 Oct  3 20:57 ntp
drwxr-xr-x  2 106    0      512 Oct 18 18:18 otdoc
drwxr-xr-x  2 106    0      512 Nov 27 15:02 utools
226 ASCII Transfer complete.
317 bytes received in 0.09 seconds (3.46 Kbytes/s)
ftp> cd otdoc
250 CWD command successful.
ftp> dir
200 PORT command successful.
150 ASCII data connection for /bin/ls (128.118.58.171,2584).
total 509
```

```
psusun01.4: rcp telnet.script shire:jellyroll.script
```

Figure 10: rcp: Copying a Single File

```
-rw-r--r--  1 106      10          453 Oct 18 18:48 OTCDOC.TXT
-rw-r--r--  1 106      0         80875 Oct 18 17:55 backbone.memo
-rw-r--r--  1 106      0         61740 Oct 18 17:56 hitchhkr.memo
-rw-r--r--  1 106     10         89868 Oct 18 17:01 internet.memo
-rw-r--r--  1 106      0        159200 Oct 18 17:56 resguide.memo
-rw-r--r--  1 106      0        103884 Oct 18 17:56 rfclist.memo
226 ASCII Transfer complete.
421 bytes received in 0.1 seconds (3.95 Kbytes/s)
ftp> get OTCDOC.TXT
200 PORT command successful.
150 ASCII data connection for OTCDOC.TXT (128.118.58.171,2585) (453
bytes).
226 ASCII Transfer complete.
local: OTCDOC.TXT remote: OTCDOC.TXT
464 bytes received in 0.01 seconds (30.46 Kbytes/s)
ftp> quit
221 Goodbye.
```

9.3 rcp: Remote Copy

A second facility for transferring files between machine is known as **rcp**, or remote copy. **rcp** can only be used between Unix systems. **rcp**, like **rlogin**, uses the `.rhosts` file and `/etc/hosts.equiv` to determine whether you have unchallenged access to the remote system. However, it does not give you the opportunity to supply a password if you do not have unchallenged access.

rcp does have some advantages over **ftp**, when it can be used. **rcp** is able to recursively traverse a directory tree, copying all of the files in a tree, where **ftp** forces you to manually copy all of the files at each level of the tree. **rcp** preserves the execution privileges of the files it copies, where **ftp** does not. This means that if you copy an executable file with **ftp**, you must use **chmod** to make it executable again, whereas with **rcp** this would not be necessary. Figures 10 and 11 give examples of **rcp** are shown below. For more complete descriptions, see the online manual pages.

10 Remote Execution of Commands

In addition to file transfer and remote login, remote execution of commands is also useful. **rsh**, the remote shell command, allows you to execute a single command on a remote system without having to formally log on. This can save you time and be easier when you know that you only want to do a single command. **rsh**, like **rlogin** and **rcp**, uses the `.rhosts` file and `/etc/hosts.equiv` to determine whether you have unchallenged access to the remote system. However, it does not give you the opportunity to supply a password if you do not have unchallenged access. For more information, see the online manual pages for **rsh**. The basic syntax of the **rsh** command is shown below:

```
psusun01.7: rcp -r shire:pic448 pic448
```

Figure 11: `rcp`: Copying a Directory Structure

```
psusun01.1> ping igor  
igor.psu.edu is alive
```

Figure 12: `ping`: Single Echo Request

```
rsh hostname [-l username] [-n] command
```

11 Network Queries

The Internet, as well as any local network, is useful for executing queries at remote machines. You might be interested in the status of the machine, whether it is up or down, how long it has been up, or even what the load is on the machine. Additionally, you might be interested in a particular user at another host. Several commands exist for executing remote queries; here we will talk about some of the most commonly used.

11.1 `ping`: Checking the Reachability of a Remote Host

The `ping` command is used to determine whether or not a remote host is reachable via the network. Its syntax is simple, and the information that it returns is also simple. The basic syntax of the `ping` command is shown below.

```
ping hostname
```

`ping` works by sending a request to the remote host to echo back the data sent to it. This is known as an echo request. On some systems, `ping` will send only a single query to the remote host, and will return whether or not it is able to return its echo request. Other systems continue sending echo requests each second, displaying the time that the request took to be answered. See your local manual page for exact information. The `ping` command should be used with care, since it can impose a heavy load on the network.

11.2 `rup`: Displaying System Loads on a Local Network

Sometimes it is useful to know not only that a particular host is reachable through the network but also how long it has been running and what the system load is. `rup` will broadcast a request to all of the machines on the local network and request this information from them. In its general form, it will only report on machines on the same local network. However, if hostnames are supplied, to query particular machines, the request is sent directly to that machine and can cross onto other networks. The basic syntax of `rup` is shown below. If no hosts are listed, information is returned for all machines answering on the local network.

```

jellyroll.1: ping psusun01
PING psusun01.psu.edu: 56 data bytes
64 bytes from 128.118.58.6: icmp_seq=0. time=5. ms
64 bytes from 128.118.58.6: icmp_seq=1. time=4. ms
64 bytes from 128.118.58.6: icmp_seq=2. time=4. ms
64 bytes from 128.118.58.6: icmp_seq=3. time=4. ms
^C
----psusun01.psu.edu PING Statistics----
4 packets transmitted, 4 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 4/4/5

```

Figure 13: ping: Multiple Echo Requests

```

psusun01.1: rup
bigboote.psu    up  4 days, 19:16,    load average: 0.00, 1.17, 0.07
bubba.psu.ed   up 12 days, 10:24,    load average: 0.00, 0.00, 0.00
jellyroll.ps   up           3:31,    load average: 0.00, 0.22, 0.39
buckaroo.psu   up 18 days,  2:35,    load average: 0.04, 0.00, 0.00
psusun03.psu   up  4 days,  2:06,    load average: 0.00, 0.00, 0.00
psusun01.psu   up  2 days,  6:59,    load average: 1.25, 1.13, 1.01
igor.psu.edu   up 12 days,  1:27,    load average: 0.82, 0.79, 0.50

```

Figure 14: rup:Querying Local Network

```
rup [hostname...]
```

11.3 rusers: Listing Users Logged in at Remote Sites

It is also useful to be able to find out who is logged onto another machine; **rusers** will give you this information. **rusers** is much like **rup** in that it will, by default, broadcast to all of the machines on its local network, but if given a hostname for the query, will request the information directly from that machine. See the manual page for more a more complete description of the command syntax. The basic syntax is shown below.

```
rusers [hostname]
```

```

jellyroll.9: rusers
jellyroll    dickson dickson
psusun03.psu jchen jchen jchen jchen jchen jchen
psusun01.psu sahani dickson

```

Figure 15: rusers: Querying Local Network

```
jellyroll.14: rusers -l expo.lcs.mit.edu
keith    EXPO.LCS.MIT:ttyp0    Dec  2 11:18    1:27 (xenon:0.0)
cjl      EXPO.LCS.MIT:ttyp1    Dec  3 09:06    8:42 (otis.ai.mit.edu)
jim      EXPO.LCS.MIT:ttyp2    Dec  3 11:51    4:45 (kanga:0.0)
converse EXPO.LCS.MIT:ttyp3    Dec  3 13:11     6 (exalt:0.0)
rws      EXPO.LCS.MIT:ttyp4    Dec  3 13:23    51 (xyther:0.0)
swick    EXPO.LCS.MIT:ttyp5    Dec  3 15:28    2:33 (liar.mit.edu)
rws      EXPO.LCS.MIT:ttyp6    Dec  3 15:54    47 (xyther:0.0)
```

Figure 16: `rusers`: Querying a Specific Host

```
psusun01.1: finger
Login      Name          TTY Idle   When      Where
sahni     Anirudh Sahni   co      Sun 15:27
dickson   Scott Dickson  p5 2:34 Sun 15:38  jellyroll.psu.ed
```

Figure 17: `finger`: Querying the Local Host

11.4 `finger`: Getting More Detailed Information About Users

Sometimes, you might want more information about a user than whether or not she is logged on to a particular system. `finger` lists the login name, full name, terminal name and write status, idle time, login time, and office location and phone number (if they are known) for each current UNIX user. A longer format also exists and is used by `finger` whenever a list of people's names is given. (Account names as well as first and last names of users are accepted.) This format is multi-line and includes all the information described above as well as the user's home directory, login shell, plan, and project. The plan and project information is located in the user-customizable files `.plan` and `.project` in the user's home directory. `.project` is generally a one-line file that tells what project you are currently working on. The `.plan` file contains more general information such as an address and phone number as well as other personal data.

`finger` may be used to lookup users on a remote machine by specifying a user and a host to query.. The format of a name to be queried is `user@host`. If only `@hostname` is provided, the standard format listing is provided on the remote machine. The basic syntax of `finger` is shown below and in the examples below. For more complete information, see the manual pages on your local machine.

```
finger [names ...]
```

12 Mail and Unix Networking

Mail is a large and important topic that is certainly related to Unix networking, but is not necessarily integral to it. The fact that sending and receiving mail may use networking protocols is not really so important as the fact that you *can* send and receive mail in a timely fashion. In fact, many sites do not use Internet protocols for electronic mail, but use slower methods.

```

jellyroll.17: finger jim
Login name: jim                In real life: Jim Duncan
Directory: /Users/jellyroll/jim  Shell: /bin/csh
Last login Wed Oct 25 23:55 on ttyp0 from tni.psu.edu
Project: To find an interesting Unix job in State College,
Pennsylvania.
Plan:

```

If you ask me for help, I expect you to have at least read the man page(s) related to your question, or at least to have made an attempt to do so.

Figure 18: `finger`: Querying a Local User by Userid

```

jellyroll.16: finger @shire.cs.psu.edu
[shire.cs.psu.edu]
Login      Name                TTY Idle   When      Where
tim        Tim Thomas          p0  42 Sun 16:28  belegost.endor.c
owens      Robert Michael Owens *p1 2:00 Sun 12:22  annex3.cs.psu.ed
pangrle    Barry Pangrle       p3  7:05 Sun 11:20  annex3.cs.psu.ed
flee       Felix Lee           p7  35 Sun 14:13  annex2.cs.psu.ed

```

Figure 19: `finger`: Querying a Remote System

12.1 Parts of the Mail System

However, a short discussion of how to send and receive mail is in order. First, you need to understand that there are several parts to the process of sending and receiving mail. The portion of the process you are probably most familiar with is using the *user-agent* program. The user-agent is the program that the user uses to interface with the mail system. This program is responsible for allowing you to read and send mail, as well as possibly manage folders of saved mail. Secondly, there is a *delivery-agent* whose job is to actually manage the transmission and receipt of the mail. This is done under the covers, and generally outside of the control of the user.

On Unix systems, the `sendmail` program is the standard delivery-agent. It must be set up by the system administrator to handle mail to all of the sorts of places that users of the system will want to send mail to.

12.2 Mail User-Agent Programs

There are a huge number of user-agent programs available for Unix. One common thing to do is to write a new user-agent to do more exactly what you want. However, a standard agent on virtually all systems is called simply `mail`, `Mail`, or `/usr/ucb/mail`. This is a simple, line-at-a-time user-agent that allows you to read messages, send messages, and do simple folder manipulation. We will discuss `Mail` more fully shortly.

Some of the other user-agent programs of note would be `elm`, `pine`, `mh`, `xmh`, and `mailtool`. `elm` and `pine` are terminal-based, full-screen mail systems, somewhat reminiscent of `RDRLIST` on VM. They present a list of all of your messages, you use the cursor keys to select messages to view, and have more or less full control

```

jellyroll.22: finger scott@shire.cs.psu.edu
[shire.cs.psu.edu]
Login name: dickson                In real life: Scott Dickson
Directory: /home/shire/dickson     Shell: /bin/csh
Last login Thu Nov 16 09:54 on ttyb from 128.118.88.254
No unread mail
No Plan.

Login name: schwartz              In real life: Scott Schwartz
Directory: /home/staff/schwartz    Shell: /usr/bin/tcsh
Last login Sat Dec  2 21:45 on tty6 from psusun01.psu.edu
New mail received Sun Dec  3 18:13:43 1989;
  unread since Sun Dec  3 04:47:06 1989
Project: Turing Police. 865-1212(home) 865-4700(lab)
Plan:
  "Part of the charm [of Alan Turing] is that there are so few dead
  computer scientists." -- Paul Callahan
  "Tell me about... computers!" -- The Evil Being, _Time Bandits_

Login name: scott                 In real life: L Ridgway Scott
Directory: /home/shire/scott       Shell: /bin/csh
Last login Tue Oct 17 16:49 on tty6 from lagrange.psu.edu
No unread mail
No Plan.

```

Figure 20: finger: Querying a Remote User by Name

of the screen. **mh** is a collection of programs that provide a set of filters for dealing with your mail. **xmh** is an X Window System interface to the **mh** mail programs. **mh** and **xmh** provide an amazingly strong and flexible system for managing mail, however they are somewhat large and hard to learn. For information on them, see [11]. **mailtool** is a windows-based user-agent that is supplied by Sun with their Open Windows product. Like **xmh**, it uses X Windows, and allows fairly easy manipulation of electronic mail.

There are many, many other user agent programs available for handling mail. The one you select is dependent on what is available on your system, and how much flexibility and power you really need in managing mail. **Mail** is almost universally available, so we will discuss it more thoroughly.

12.3 Mail Addresses

The addressing of electronic mail is seen by some as magic. Generally, it is quite simple and straight-forward. An address can be broken down into two parts: a local part and a remote part. These two parts are joined by an at sign (`@`). The remote part specifies where this mail is going to be delivered to, and the local part says to what entity at the remote site the mail should go to.

Most of the time, the local part of an address is a person's name or login name and the remote part of the address is the name of the machine on the network where this person receives mail. In some large organizations, it may be the case that the local and remote parts of the address appear to be like this, but some other style of local delivery may actually take place. That should not concern you, since the local and remote parts of the address are still sensible.

However, sometimes, special provisions must be made to address mail so that it actually will get to a particular site. If no automatic passing of mail from one network to another happens, you may have to specify an intermediate point that is connected to both networks as the remote part, and then make the local part of the address specify how to get the rest of the way to the desired destination.

For this discussion, we will assume that standard, simple addressing is sufficient and that in order to send mail to a particular person at a particular system, you may address the mail to `user@host.domain`. For example, to send to the user `dickson` at the host `iris.psu.edu`, send mail to `dickson@iris.psu.edu`.

12.4 /usr/ucb/mail

The standard mail program on Unix is the **Mail** program developed at U.C. Berkeley. This is a simple, command-oriented, line-at-a-time mail system. It has two different modes of operation. First, if the command **Mail** is entered on the command line, you will be placed in an interactive session where you can give single letter commands to review what mail is waiting, to read mail, to move messages between folders, and to send mail.

Secondly, if an electronic mail address is given as an argument, **Mail** assumes that you want to send mail to that address. You will then be prompted for the subject of the message and then its body. Once you have entered all of the message, simply end the note with either the Control-D key or a period on a line by itself, and the mail will be sent.

The following is a sample session with **Mail**. It will show most of the major features of the system. For more information, consult the manual pages for **Mail**.

Sample Session Using /usr/ucb/mail

iris[1]: Mail

Mail version SMI 4.1-0WV3 Mon Sep 23 07:17:24 PDT 1991 Type ? for help.

"/var/spool/mail/dickson": 107 messages

```
> 1 ramani Sat Oct 17 19:51 25/873 Difficulty connecting to
2 BLL@PSUVM.PSU.EDU Wed Oct 21 16:37 19/809 Re: reserving 141
3 WHV@PSUVM.PSU.EDU Fri Oct 9 15:13 68/2817 Campaign '92 newsgroups
4 WHV@PSUVM.PSU.EDU Wed Oct 21 15:28 41/1739 Re: nptn.campaign92.* new
5 REH113@PSUVM.PSU.EDU Fri Sep 11 01:20 50/2636 new weather program
6 REH113@PSUVM.PSU.EDU Sun Sep 13 01:31 19/738 wx program
7 REH113@PSUVM.PSU.EDU Tue Sep 29 17:13 24/881
8 bufalini@gis.psu.edu Sun Aug 9 22:22 31/938 Retaining my account
9 bufalini@gis.psu.edu Tue Oct 6 14:59 25/862 My account
10 chakra Tue Oct 13 05:16 32/734 cron job..
11 cocklin@astro.psu.edu Mon Aug 17 16:40 38/1175 Re: pumping data over the
12 ESH101@PSUVM.PSU.EDU Sat Oct 3 20:15 40/1538 Re: How?
13 GEORGE@ecl.psu.edu Fri Oct 2 15:37 29/871 seminar handouts
14 meij@pop.psu.edu Thu Sep 10 14:24 32/1187 Re: stf files
15 ALMONEY@PSUVM.PSU.EDU Thu Oct 1 11:27 21/769 Micro Bats
16 ALMONEY@PSUVM.PSU.EDU Thu Oct 8 19:04 41/1515 UNIX Account
17 wagner@orville.psu.edu Wed Oct 14 23:48 23/756 TIME
18 constabl Wed Sep 9 08:34 36/1070 Help
19 P-MDICKSON@bss3.umd.edu Thu Oct 22 09:11 47/2124 nothing much
20 PGP@PSUVM.PSU.EDU Wed Aug 26 15:14 18/653 unix adm
21 REH113@PSUVM.PSU.EDU Wed Sep 9 15:00 37/1732 fortran call to unix
22 laforge@wilbur.psu.edu Mon Aug 31 11:51 34/958 small mail problem on wil
23 DDR104@PSUVM.PSU.EDU Thu Oct 8 15:31 25/1140 sun account
24 GHB1@PSUVM.PSU.EDU Mon Sep 21 11:22 26/966 SUN account
```

&

Start Mail and get a list of messages pending. & is the default prompt in Mail.

& h 25

```
> 25 meyer@vivaldi.psu.edu Tue Oct 6 12:41 23/897 Disk usage
26 bill@essc.psu.edu Mon Oct 19 12:16 180/6745 FYI: new release of NCARG
27 hewitson@gis.psu.edu Tue Oct 20 15:00 20/636 Re: cloth
28 cert-tools-request@cert.org Tue Sep 1 22:22 88/4368 log_tcp (tcp wrapper) upd
29 cert-tools-request@cert.org Thu Oct 15 12:53 684/35179 An Architectural Overview
30 cms@wilbur.psu.edu Wed Jul 29 15:06 24/708 I-DEAS
31 mcginnis@essc.psu.edu Tue Oct 6 11:41 21/1730 printer
32 elp@wilbur.psu.edu Mon Jul 27 19:21 27/1047 e-mail difficulty
33 Elaine_Wolfe@transarc.com Mon Oct 19 10:53 55/2356 Next AFSUG
34 coppeler@cs.psu.edu Thu Aug 27 11:35 20/662 Re: SGI boxes
35 GWI@icf.hrb.com Thu Oct 22 09:08 45/1823 News feed?
36 HAMMAN@nbc.upenn.edu Thu Oct 15 17:23 36/1587 Saying Hi
37 admin@hpcwire.ans.net Mon Oct 12 18:26 85/4344 HPCwire
38 vincent Sat Oct 17 20:29 19/630 Disk quota
39 vincent@orville.psu.edu Tue Jul 28 10:54 25/992 Terminal emulation
40 jsb@fubar.bk.psu.edu Sun Aug 23 19:00 29/1083 NeXT's name service
41 jeremy@essc.psu.edu Wed Sep 9 05:10 21/770 Re: closed account..
42 jim@math.psu.edu Tue Aug 18 11:25 85/2998 pod bug and patch
43 jim@math.psu.edu Wed Aug 26 13:54 378/4305 meeting times
44 jim@math.psu.edu Fri Oct 2 11:17 51/2351 meeting POSTPONED
```

```
45 jim@math.psu.edu   Fri Oct 16 13:36   27/1032   ud and isode
46 jim@math.psu.edu   Mon Oct 19 10:41   42/1455   Re: unix-admin today
47 jgd@uwm.edu        Thu Aug 20 19:17   42/1498   Re: Announcing the releas
48 jajohns@moe.coe.uga.edu Mon Oct 19 10:05   40/1636   Re: go bravos
```

&

Get a list of messages starting with a particular number. Every message is numbered.

& ?

```
cd [directory] chdir to directory or home if none given
d [message list] delete messages
e [message list] edit messages
f [message list] show from lines of messages
h print out active message headers
m [user list] mail to specific users
n goto and type next message
p [message list] print messages
pre [message list] make messages go back to system mailbox
q quit, saving unresolved messages in mbox
r [message list] reply to sender (only) of messages
R [message list] reply to sender and all recipients of messages
s [message list] file append messages to file
t [message list] type messages (same as print)
top [message list] show top lines of messages
u [message list] undelete messages
v [message list] edit messages with display editor
w [message list] file append messages to file, without from line
x quit, do not change system mailbox
z [-] display next [previous] page of headers
! shell escape
```

A [message list] consists of integers, ranges of same, or user names separated by spaces. If omitted, Mail uses the current message.

&

Get a list of commands that can be used inside of Mail.

& 47

Message 47:

```
From psu-nntp-managers-request@cs.psu.edu Thu Aug 20 19:17:07 1992
From: John Dobnick <jgd@uwm.edu>
Resent-From: psu-nntp-managers-request@cs.psu.edu
Resent-To: nntp-mgrs-list@cs.psu.edu
Subject: Re: Announcing the release of InterNetNews
To: rsalz@uunet.uu.net (Rich Salz)
Date: Thu, 20 Aug 1992 18:55:51 -0400
Cc: nntp-managers@ucbvax.Berkeley.EDU
In-Reply-To: <9208201440.AA00382@news.UU.NET>; from "Rich Salz" at Aug 20, 92 10:40 am
Resent-Sender: prepnet-nntp-managers-request@cs.psu.edu
Sender: psu-nntp-managers-request@cs.psu.edu
Resent-Message-Id: <92Aug20.191657edt.294367@psuvax1.cs.psu.edu>
```

Resent-Date: Thu, 20 Aug 1992 19:16:44 -0400
Content-Length: 524
X-Status:

There is a small typo in the INN announcement. In the list of FTP sites, please change

```
> ftp.uu.net /pub/news/nntp/inn
to      ftp.uu.net          /news/nntp/inn
```

[Oh boy! More stuff to read! :-)]

--

```
John G Dobnick (News Janitor)      ATTnet: (414) 229-5727
Computing Services Division        INTERNET: jgd@uwm.edu
University of Wisconsin - Milwaukee UUCP: uunet!uwm!jgd
```

"Knowing how things work is the basis for appreciation,
and is thus a source of civilized delight." -- William Safire

```
& s inn
"inn" [New file] 42/1498
& q
Held 106 messages in /var/spool/mail/dickson
```

View a particular message, and save it in a folder. Then quit the mail system.

The following example shows how you would go about sending a piece of mail using **Mail**. In it, we send mail to a particular user (**kalbach@hornet.psu.edu**), and we see what special commands can be accessed while entering the text of a message. These commands are accessed via the key. All of the commands are a single character preceded by the tilde. These commands allow you to enter an editor, modify who gets the mail, read in other files, and so forth.

Sending Mail with Mail

```
iris[1]: Mail kalbach@hornet.psu.edu
Subject: Hi John
This is just a test message.
```

~?

```
----- ~ ESCAPES -----
~~ Quote a single tilde
~a,~A Autograph (insert 'sign' variable)
~b users Add users to Bcc list
~c users Add users to Cc list
~d Read in dead.letter file
~e Edit the message buffer
~m messages Read in messages, right-shifted by a tab
~f messages Read in messages, do not right-shift
```

```

~h Prompt for To list, Subject and Cc list
~p Print the message buffer
~q,~Q Quit, save letter in $HOME/dead.letter
~x Quit, do not save letter
~r file Read a file into the message buffer
~s subject Set subject
~t users Add users to To list
~v Invoke display editor on message
~w file Write message onto file
~. End of input
~? Print this message
~!command Run a shell command
~|command Pipe the message through the command
~:command Execute regular Mail command
-----
.
EOT

```

13 Usenet and Network News

Like electronic mail, Usenet and network news are not completely related to Unix networking, but no discussion of networking under Unix would be complete without a mention of Usenet. Usenet is a world-wide computer bulletin board system that is accessible from many different types of computers. It is closely associated with Unix because it was first created on a Unix system. It is closely related to networking because the Internet is one of the primary ways that data is placed on the bulletin board and that data may be accessed. However, other, non-Internet methods for using Usenet exist and are widely used.

Usenet is organized in a hierarchical structure of newsgroups and hierarchies that looks much like a Unix filesystem (or internet names, for that matter). At the top level are a small number of hierarchies or general categories of news. These include the following list of mainstream classifications:

- **comp** for computer related discussions.
- **misc** for miscellaneous discussions.
- **soc** for social and societal discussions.
- **rec** for recreational discussions.
- **news** for news-system related discussions.
- **talk** for open, high-volume, spirited discussions.

In addition to the above list of hierarchies, other hierarchies, such as the **alt** hierarchy for alternative discussions, and **psu** for Penn State specific discussions, also exist. These may be restricted to a geographic or organizational level or may not have widespread acceptance.

Within each hierarchy are many discussion groups, each based around a particular topic, for example `soc.roots` for genealogical discussions or `comp.binaries.ibm.pc` for the distribution of IBM PC and DOS software.

Groups are not just created at a whim. A discussion of the need for a particular newsgroup will go on in a closely related newsgroup. Once consensus (or near consensus) has been reached as to its need, a call for votes will take place, in which the users of the network will vote electronically whether or not the group should be created. If the vote passes, then one of the administrators of the system will send a message out to all news systems to add the group.

Currently, over 2500 different discussion topics exist on over 40,000 different news systems. It is estimated that over 2 million people read news on Usenet regularly.

Like mail, news has two major parts as well. The system-agent portion of news is responsible for distributing news to any other machines that have requested it, as well as maintaining a current news system. The system is responsible for accumulating incoming news, keeping track of when it arrived, what category the news belongs in, when it should be thrown away, and of any changes in the set of newsgroups. Probably the most commonly used system agent is a package called *C-News*.

Again like mail, a huge variety of user-agent programs exist that allow you to read and post news. These range from line-at-a-time, **Mail**-like programs to fancy window interfaces to news. One common feature of many of these systems is the use of the **NNTP** network protocol to access the news. News may be kept on a central server somewhere on the network. Individual user-agents then can request articles or indexes from newsgroups from the server using this network protocol. News can also be posted and transferred between servers using **NNTP**.

Rather than demonstrate individual user-agents here, we will refer you to the manual for some of the major programs in use. The following is a list of some of the more popular news user-agent programs under Unix:

- **rn** is a simple, line-at-a-time news program, sort of like **Mail**.
- **xrn** is an X Window System interface for the **rn** program.
- **tin** is a full-screen news reader that is capable of following discussions by topic.
- **gnus** is a full-screen news reader for use with the Emacs editor.

14 Getting More Information

The information presented in the document barely scratches the surface of what can be done with networking on a Unix system. The best way to find out more is to read the manual and try what you can find. All of the commands described here can be found in the Unix online documentation with the command

`man commandname`

If you are unsure what command you are looking for, the **man** command has a keyword search facility. Using keyword searches, you give keywords that might appear in a description of a command you need. In order to use keyword searches, use the **-k** option for **man**. For example

```

vivaldi:1 man -k login
last (1)           - indicate last logins of users and teletypes
login (1)          - sign on
rlogin (1C)        - remote login
rlogind (8C)       - remote login server
tn3270 (1)         - full-screen remote login to IBM VM/CMS

```

In addition to the online manual pages, several computer vendors have special manuals for networking. Sun Microsystems has a relatively useful document called *Using the Network: Beginner's Guide* [9] that comes in the standard Sun documentation kit.

Many other in-depth references are also available. Perhaps the best book currently available for an overview of how the network works is *Internetworking with TCP/IP* [2]. This book describes how TCP and other network services work at a low level. It also discusses the RFC system.

A very good new book describing various networks and conferencing systems is *The Matrix: Computer Networks and Conferencing Systems Worldwide* [12] by John S. Quarterman.

The Whole Internet User's Guide & Catalog [6] by Ed Krol is a new, comprehensive guide to the Internet. It includes information relating to how the Internet is put together and how it functions as well as how to access different services such as **telnet**, **ftp**, **rlogin**, etc. Of special interest is a catalog of interesting data sources and services accessible on the network.

One of the primary ways that information about implementations of different network services is shared within the Internet is through RFC's. An RFC, or Request For Comments, is a document that describes some aspect of system operation. These are available for anonymous **ftp** at several sites around the Internet, or through the NIC, mentioned earlier. RFC's describe the low level functionality of different network services and are not generally designed for the novice user. However they provide invaluable assistance to network programmers and maintainers. Several helpful RFC's are listed in the Bibliography [8, 7, 13, 5].

Locally, the Office of Telecommunications has several useful documents. These are available via anonymous **ftp** from **ftp.otc.psu.edu** or through the Office of Telecommunications. Available documents include Guide to the Penn State Data Backbone[10] by OTC, Introduction to Internet Protocols[3] and Introduction to the Administration of an Internet-based Local Network[4] by C.L. Hedrick, and the Internet Resource Guide[1].

15 Acknowledgements

Ideas and information for this document were drawn from many sources and many people. Some of these include the various Internet RFC's mentioned above, *Internetworking with TCP/IP*[2] by Douglas Comer, and *A Local Internet User's Guide* by Jim Duncan of the Department of Mathematics. I would like to thank the people who read this document and provided valuable comments and suggestions: Jim Duncan, Dan Ehrlich, Lou Pepe, and Ray Masters.

References

- [1] NSF Network Service Center. Internet Resource Guide. 1991.

- [2] Douglas Comer. *Internetworking with TCP/IP: Principles, Protocols, and Architecture*. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1988.
- [3] Charles L. Hedrick. *Introduction to Internet Protocols*. 1987.
- [4] Charles L. Hedrick. *An Introduction to Administration of an Internet-Based Local Network*. 1988.
- [5] Krol. *Hitchhiker's Guide to the Internet*.
- [6] Ed Krol. *The Whole Internet User's Guide & Catalog*. O'Reilly & Associates, Inc., Sebastopol, CA, 1992.
- [7] Malkin. RFC 1206: FYI on Questions and Answers - Answers to Commonly Asked New Internet User Questions.
- [8] Malkin. RFC 1207: FYI on Questions and Answers - Answers to Commonly Asked Experienced Internet User Questions.
- [9] Sun Microsystems. *Using the Network: Beginner's Guide*. 1988.
- [10] Office of Telecommunications. *A Guide to the Penn State Data Backbone*. 1991.
- [11] Jerry Peek. *MH & xmh - E-mail for Users & Programmers*. O'Reilly & Associates, Inc., Sebastopol, CA, 1990.
- [12] John S. Quarterman. *The Matrix: Computer Networks and Conferencing Systems Worldwide*.
- [13] Socolofsky. RFC 1180: A TCP/IP Tutorial.
- [14] Richard Stevens. *Unix Network Programming*.