

## Introduction to \*nix Network Programming

### Aims:

The aim of this worksheet is to introduce tools that are useful in determining the nature of a network, e.g. **ifconfig**, **ping** and **arp**. It also takes a first step into using 'well known' ports by experimenting with standard client/ server facilities, e.g. **telnet** and **echo** and examines the files that must be maintained for network access.

### Tools.

#### **ifconfig.** Interface Configuration.

Using **ifconfig** (with no command line parameters) gives a list of all of the network interfaces installed and configured on the host machine. In an appropriate window type:

```
ifconfig
```

(If this results in an error message, `ifconfig: command not found`, this implies that the correct search path has not been set up. Try the full path name, `/sbin/ifconfig`). This should display details of the loopback interface. This interface is useful if we have no connection to a network or to test programs before using the network itself. Details of the Ethernet interface (`eth0`) should also be displayed. The loopback details should be something like:

```
lo          Link encap:Local Loopback
            inet addr:127.0.0.1  Mask:255.0.0.0
            UP LOOPBACK RUNNING  MTU:3924  Metric:1
            RX packets:230 errors:0 dropped:0 overruns:0 frame:0
            TX packets:230 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:0
```

It should be easy enough to guess what the fields mean. From the details for `eth0`, write down the IP address and broadcast address of the host machine (these could be useful later!). To find out more information on **ifconfig** try reading the **manual** page, type:

```
man ifconfig
```

Use the *spacebar* to move forward a page, *b* to move back a page, *enter* to move a line at a time, */* to search for a keyword and *q* to exit. Use the **man** page to discover what MTU stands for in the **ifconfig** display.

#### **netstat.** Network Statistics.

Another way to obtain information about the network interfaces is to use **netstat**. Not only can details of interfaces be displayed but information about any servers that are running.

Try:

```
netstat -i
```

This gives data about the interfaces in a slightly different form to that of **ifconfig**.

```
Kernel Interface table
Iface  MTU Met  RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
eth0   1500 0      0      0      0      0      806   0      0      0      0 BRU
lo     3924 0     344    0      0      0      344   0      0      0      0 LRU
```

This tool can be used to get details of running servers and open sockets. Try:

```
netstat -a
```

to include **all** interfaces. This gives rather a lot of information so it might be best to pipe it into **more** so that the output is paginated (as with **man** pages!), i.e. type:

```
netstat -a | more
```

The resulting output should be something like the following.

```
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp      0      0 *:www                   *:                       LISTEN
tcp      0      0 *:smtp                  *:                       LISTEN
tcp      0      0 *:finger                *:                       LISTEN
tcp      0      0 *:login                 *:                       LISTEN
tcp      0      0 *:shell                 *:                       LISTEN
tcp      0      0 *:telnet                *:                       LISTEN
tcp      0      0 *:ftp                   *:                       LISTEN
tcp      0      0 localhost:domain       *:                       LISTEN
udp      0      0 localhost:domain       *:                       LISTEN
udp      0      0 *:sunrpc                *:                       LISTEN
raw      0      0 *:icmp                  *:                       7
raw      0      0 *:tcp                   *:                       7

Active UNIX domain sockets (servers and established)
Proto RefCnt Flags               Type           State         I-Node Path
unix  1      [ ]                  STREAM        CONNECTED     1030 @00000047
unix  0      [ ACC ]              STREAM        LISTENING     852  /tmp/.font-unix/fs-1
unix  1      [ ]                  STREAM        CONNECTED     450  @0000001a
unix  1      [ ]                  STREAM        CONNECTED     1029 @00000046
unix  1      [ ]                  STREAM        CONNECTED     1023 @00000045
unix  1      [ ]                  STREAM        CONNECTED     1021 @00000044
unix  1      [ ]                  STREAM        CONNECTED     758  @00000031
unix  0      [ ACC ]              STREAM        LISTENING     920  /tmp/.X11-unix/X0
unix  1      [ ]                  STREAM        CONNECTED     540  @00000021
unix  0      [ ]                  STREAM        CONNECTED     119  @00000010
unix  1      [ ]                  STREAM        CONNECTED     1016 @00000042
unix  0      [ ACC ]              STREAM        LISTENING     790  /dev/gpmctl
unix  0      [ ACC ]              STREAM        LISTENING     434  /dev/log
unix  1      [ ]                  STREAM        CONNECTED     1032 /tmp/.X11-unix/X0
```

To get the process id (pid) and name of the function using the connection we can use the **p** switch. Try:

```
netstat -ap | more
```

and notice the difference. Note whether the daytime and echo servers are running, we shall need these for testing some of our socket programs. If they are not running we need to get them started (more later).

### **ping.** Packet Internet Groper.

This tool may be used to discover whether a particular host is connected to the network. It repeatedly sends packets of data to the requested host and expects the data to be echoed back. To stop it, use Ctrl-C. Try **pinging** the loopback address by:

```
ping 127.0.0.1 or ping localhost
```

Try **pinging** the IP address of the neighbouring machine.

Try **pinging** the broadcast address discovered by using **ifconfig**. What is displayed?

Note that if we `ping localhost` the display actually shows the IP address of *localhost*.

Determine the IP address of *kenny* (the ‘controller’ of our little network!). Something to think about - How does the operating system resolve the host names to IP addresses? (see later!)

**hostname**. Get Host Name.

To get the name of the machine we are working on type:

```
hostname
```

**arp**. Address Resolution Protocol.

Used to display or set IP address MAC address pairs, i.e. associates IP addresses to their hardware addresses. Try:

```
arp -a
```

Use the **man** pages for more information on these tools.

### **‘Well known’ Ports.**

Standard services are connected to ‘well known’ ports. These ports have servers actively listening on them at all times (if they have been activated!). Clients can connect to these services to obtain the ‘data’ they supply. A useful *client* for us to use to test our own network servers is **telnet**. Useful *servers* for us to test our own clients are the daytime service and the echo service. When a connection is established either the client or the server may close the connection (perform an *active close!*). Try getting the date and time of day from the localhost by:

```
telnet localhost daytime
```

Try replacing `localhost` with the IP address of your host, your neighbour’s and perhaps *kenny*. Which end performed the *active close*?

Now try connecting to the echo server. You will need to type in some text to be echoed and close the connection using Ctrl-]. Finally end the **telnet** session with *quit*.

### **Files used in Network Access.**

**/etc/services**.

This lists all the possible network services, their port number and the protocol they use, even if the server is not actually running. Try typing:

```
more /etc/services
```

What are the ‘well known’ port numbers for daytime and echo? Try to **telnet** to these services by their port number rather than their name.

If we were to create a new service then we would need to add details of that service to this file. Unfortunately we need to be *root* (the System Administrator to do this!).

**/etc/protocols.**

Just as services are referred to by a number so are protocols. The file `/etc/protocols` lists the names of the protocols together with their associated number and the equivalent name for that number. If we were to invent a new protocol we would add to this file! Inspect the file using `more`. What are the numbers associated with `tcp` and `udp`?

**/etc/hosts.**

A local list of host names and IP addresses. This is used to resolve host names to IP addresses for use by tools such as `ping`. As `root` we can alter this list to hold details of hosts we contact often. It remains static the details entered by `root` do not change dynamically as we access the network.

**/etc/resolv.conf.**

Instead of using the static file `/etc/hosts` to resolve hostnames to IP addresses a host can make use of another machine that acts as a name server (e.g. kenny). The Domain Name Service (DNS) needs to know the IP address of the name server. This information is held in the file `/etc/resolv.conf`.

**/etc/host.conf.**

Obviously it is quicker to use a local list of hostname/IP address pairs than it is to use DNS. The system can be instructed to inspect `/etc/hosts` before consulting the nameserver. This is achieved by specifying the lookup order in the file `/etc/host.conf`.

**/etc/inetd.conf.**

Since there are many standard servers that need to be started at boot-up they are listed in `/etc/inetd.conf`. When Linux boots a super internet daemon (`inetd`) gets started at some point and this starts the other daemons (servers) listed in `/etc/inetd.conf`.

If our `telnet` connection to `daytime` or `echo` is refused then it is probably the case that these servers have not been started. Inspect `/etc/inetd.conf` by using `more`. The declaration of `daytime` and `echo` probably start with a `#` sign, this indicates a comment. We need to login as `root` and use `emacs` (or some text editor) to remove the comment. These services need to be made active. As `root` we need to hangup the inet daemon (`inetd`) and restart it so that it can act on the new services. As `root` we need to get the process id (pid) of `inetd`, hang up the process and then restart it. So type:

```
ps -ax | grep inetd    the process status of all processes then inetd in particular.
kill -HUP pid         use the pid gained above.
```

Once this has been done the original user can re-enter the `telnet` command and the connection should be made.