Network Commands

The network commands chapter explains various tools which can be useful when networking with other computers both within the network and accross the internet, obtaining more information about other computers. This chapter also includes information on tools for network configuration, file transfer and working with remote machines.

netstat

Displays contents of /proc/net files. It works with the Linux Network Subsystem, it will tell you what the status of ports are ie. open, closed, waiting, masquerade connections. It will also display various other things. It has many different options.

tcpdump

This is a sniffer, a program that captures packets off a network interface and interprets them for you. It understands all basic internet protocols, and can be used to save entire packets for later inspection.

ping

The ping command (named after the sound of an active sonar system) sends echo requests to the host you specify on the command line, and lists the responses received their round trip time.

You simply use ping as:

```
ping ip_or_host_name
```

Note to stop ping (otherwise it goes forever) use **CTRL−C** (break).

hostname

Tells the user the host name of the computer they are logged into. Note: may be called *host.*

traceroute

traceroute will show the route of a packet. It attempts to list the series of hosts through which your packets travel on their way to a given destination. Also have a look at *xtraceroute* (one of several graphical equivalents of this program).

Command syntax: `traceroute machine_name_or_ip`

tracepath

tracepath performs a very simlar function to *traceroute* the main difference is that *tracepath* doesn't take complicated options.

Command syntax:

```
tracepath machine_name_or_ip
```

ifconfig

This command is used to configure network interfaces, or to display their current configuration. In addition to activating and deactivating interfaces with the "up" and "down" settings, this command is necessary for setting an interface's address information if you don't have the *ifcfg* script.

Use *ifconfig* as either:

```
ifconfig
```

This will simply list all information on all network devices currently up.

```
ifconfig eth0 down
```

This will take eth0 (assuming the device exists) down, it won't be able to receive or send anything until you put the device back "up" again.

Clearly there are a lot more options for this tool, you will need to read the manual/info page to learn more about them.

host

Performs a simple lookup of an internet address (using the Domain Name System, DNS). Simply type:

```
host ip_address
```

or

```
host domain_name
```

Scheduling Commands to run in the background

There are two main tools used to perform scheduled tasks, *at* and *cron*. You may also like to try anacron if your computer does not run continuously, as cron will only work if your computer is left on (anacron can catch up with the scheduled tasks the next time the computer is on...).
at
'at' executes a command once on a particular day, at a particular time. *at* will add a particular command to be executed.
Examples:

```
at 21:30
```

You then type the commands you want executed then press the end−of−file key (normally **CTRL−D**). Also try:

```
at now + time
```

This will run at the current time + the hours/mins/seconds you specify (use *at now + 1 hour* to have command(s) run in 1 hour from now...)
You can also use the *−f* option to have at execute a particular file (a shell script).

```
at −f shell_script now + 1 hour
```

This would run the shell script 1 hour from now.

cron
cron can be used to schedule a particular function to occur every minute, hour, day, week, or month. It's normal to use the crontab to perform the editing functions as this automates the process for the cron daemon and makes it easier for normal users to use cron.
Anacron
anacron is another tool designed for systems which are not always on, such as home computers
While *cron* will not run if the computer is off, *anacron* will simply run the command when the computer is next on (it catches up with things).

crontab is used to edit, read and remove the files which the cron daemon reads.
Options for crontab (use *crontab −option(s)*):
◊ *−e* −−− to edit the file.
◊ *−l* −−− to list the contents of the file.
−u username −−− use the *−u* with a username argument to work with another users crontab file.
◊
When using *crontab −e* you have a number of fields (6) what they mean is listed below:
Field Allowed Values
minute 0−59
hour 0−23
day of month 1−31
month 1−12 (or names, see below)
day of week 0−7 (0 or 7 is Sun, or use three letter names)
There are also a number of shortcut methods for common tasks, including:[9]
◊ *@reboot* −−− run command at reboot
◊ *@yearly* −−− same as 0 0 1 1 *
◊ *@annually* −−− same as @yearly
◊ *@monthly* −−− same as 0 0 1 * *
◊ *@weekly* −−− same as 0 0 * * 0
◊ *@daily* −−− same as 0 0 * * *
◊ *@midnight* −−− same as @daily

◊ *@hourly* −−− same as 0 * * * *
[10]
Note that * (asterisk) is used to mean anything (similar to the wildcard). For example if you leave the day part (the 5th place) with an asterisk it would mean everyday.
Lists are allowed. A list is a set of numbers (or ranges) separated by commas. Examples: ``1,2,5,9'', ``0−4,8−12".

Printing files

8.1.1. General

RedHat ships with LPRng, an improved version of the well−known UNIX print system. If the printer has been
set up, all you need to do is learn how to use **lpr** to send files to the printer, which basically comes down to
lpr file(s)
Lpr uses a spooling daemon, **lpd** to print the named files when facilities become available. If no names appear, standard input is assumed (such as when the output of a command is piped into **lpr**). The **lpr** command has a lot of options, which can be displayed using the `−−help` option. Full information is in the Info pages. You will also find the **lp** command on your Linux system, for compatibility reasons with other (UNIX) programs. You will find that **lp** is in fact a symbolic link to **lpr**:

```
davy:~> ls −l /usr/bin/lp*
lrwxrwxrwx 1 root root 3 Oct 28 14:21 /usr/bin/lp −> lpr
−rwxr−xr−x 1 lp lp 395192 Aug 11 2001 /usr/bin/lpq
−rwxr−xr−x 1 lp lp 408536 Aug 11 2001 /usr/bin/lpr
−rwxr−xr−x 1 lp lp 392984 Aug 11 2001 /usr/bin/lprm
−rwxr−xr−x 1 root root 4651 Oct 19 22:17 /usr/bin/lprsetup.sh
−rwxr−xr−x 1 lp lp 398488 Aug 11 2001 /usr/bin/lpstat
davy:~> ps −ef | grep lpd
lp 1003 1 0 Feb22 ? 00:00:00 lpd Waiting
```

Once the file is accepted in the print queue, an identification number for the print job is assigned:

```
davy:~> lp /etc/profile
request id is davy@blob+253
```

To view (query) the print queue, use the **lpq** command. When entered without arguments, it displays the
contents of the default print queue.

```
davy:~> lpq
Printer: lp@blob
Queue: no printable jobs in queue
Status: job 'cfA284blob.somewhere.org' removed at 11:02:47.098
```

If you don't like what you see, use **lprm** to delete jobs. Use **lprm −** to delete all jobs which you submitted. If
you only want to cancel one job, use the number of that job as an argument to **lprm**.