

Setting Up an NFS Server

by Angus Jack

To share files between Linux machines is somewhat trickier than browsing the network neighbourhood under Windows. The Network File System (NFS) is used to allow machines to mount a disk partition on a remote machine as if it was on a local hard drive. This article briefly tries to explain the basics of configuring a Linux machine to allow a remote Linux machine to share part of its file system.

The Configuration Files

```
/etc/exports
```

This file contains a list of entries; each entry indicates a volume that is shared and how it is shared.

An entry in `/etc/exports` will typically look like this:

```
Directory remote1(opt11,opt12)
                                remote2(opt21,opt22)
```

directory

The directory that you want to share. It may be an entire volume, though it need not be. If you share a directory, then all directories under it within the same file system will be shared as well.

remote1 and remote2

Client machines that will have access to the directory. The machines may be listed by their IP address, DNS address or by an alias in the `/etc/hosts` file.

optxx

The option listing for each machine will describe what kind of access that machine will have. Important options are:

ro:

The directory is shared read only; the client machine will not be able to write to it. This is the default.

rw:

The client machine will have read and write access to the directory.

no_root_squash:

By default, any file request made by user `root` on the client machine is treated as if it is made by user `nobody` (a user defined by the system with limited privileges) on the server. If `no_root_squash` is selected, then `root` on the client machine will have the same level of access to the files on the system as `root` on the server. This, of course, can have serious security implications.

Suppose we have two client machines, `remote1` and `remote2`, that have IP addresses `172.16.100.1` and `172.16.100.2`, respectively. We wish to share our software and home directories with these machines. A typical entry in the `/etc/exports` might look like this:

```
/usr/local 172.16.100.1(rw,no_root_squash)
                                172.16.100.2(ro)
/home      172.16.100.1(rw, no_root_squash)
                                172.16.100.2(ro)
```

The machine `172.16.100.1` has read, write and root privileges to the `/usr/local` and `/home` directories of the server but `172.16.100.2` has only read privileges.

Alternatively a range of machines can be given access by using the network mask. If you wanted to allow access to all the machines with IP addresses between `172.16.100.1` and `172.16.100.254` then you could have the entries:

```
/usr/local 172.16.100.0/255.255.255.0(ro)
/home      172.16.100.0/255.255.255.0(rw)
```

The Daemons

There are three main daemons that are needed to run an NFS server.

rpc.portmap: This is the portmapper daemon. This daemon tells requesting clients how to find all the NFS services on the system.

rpc.nfsd: This is the main daemon which does most of the work.

rpc.mountd: This handles the initial mount requests.

Three other daemons are `rpc.statd`, `rpc.rquotad`, and `rpc.lockd`. These daemons do not have to be running to get an NFS server to work but they will help you if a client machine goes down during a file copy.

Typing their name at the command line can start the daemons. To check that NFS is running type `rpcinfo -p` and you should get an output like this:

| Program | vers | proto | port | |
|---------|------|-------|------|------------|
| 100000 | 2 | tcp | 111 | portmapper |
| 100000 | 2 | udp | 111 | portmapper |
| 100011 | 1 | udp | 950 | rquotad |
| 100011 | 2 | udp | 950 | rquotad |
| 100005 | 1 | udp | 957 | mountd |
| 100005 | 1 | tcp | 959 | mountd |
| 100005 | 2 | udp | 962 | mountd |
| 100005 | 2 | tcp | 964 | mountd |
| 100003 | 2 | udp | 2049 | nfs |

This would indicate that I was running NFS version 2, `mountd` versions 1 and 2 etc. Of course, these daemons should be called from a startup script to maintain NFS services after a reboot. Typically `/etc/rc.d/init.d/nfs` would be appropriate. On many distributions, a script will be created automatically if NFS is selected as a kernel module either during the install or later.

Controlling Access

`/etc/hosts.allow` and `/etc/hosts.deny`

These two files specify which computers on the network can use services on your machine. Each line of the file is an entry listing a service and a set of machines. When the server gets a request from a machine, it does the following:

It first checks `hosts.allow` to see if the machine matches a description listed in there. If it does, then the machine is allowed access.

If the machine does not match an entry in `hosts.allow`, the server then checks `hosts.deny` to see if the client matches a listing in there. If it does then the machine is denied access.

If the client matches no listings in either file, then it is allowed access.

Adding the following to the `/etc/hosts.deny` file will first of all block access to all of the NFS daemons:

```
lockd:ALL
mountd:ALL
rquotad:ALL
statd:ALL
```

We can now allow selective access to our client machine `remote1` and `remote 2` by adding them to the `/etc/hosts.allow` file:

```
lockd:      172.16.100.1 , 172.16.100.2
rquotad:    172.16.100.1 , 172.16.100.2
mountd:     172.16.100.1 , 172.16.100.2
statd:      172.16.100.1 , 172.16.100.2
```

Of course you may wish to change the NFS shares at any time. This can be achieved by editing the `/etc/exports` file and then forcing the system to re-read it. The following command is useful:

exportfs -ra: This effectively causes the changes to be pushed into the kernel.

Making a Connection

Finally, having gone to all this effort, you need to access the NFS server from your remote machine. First you need to create a directory on the remote machine where your server volume will be mounted. This would typically be under the `/mnt` directory, let's say it's called `/mnt/nfs_server`.

Then, sitting at the remote machine and assuming the server has the IP address `172.16.100.100`, you would issue the following:

```
mount -t nfs 172.16.100.100://home /mnt/
                                           nfs_server
```

This would cause the directory `/home` on the server to appear on the remote machine and it would appear to have all the properties of a local directory, subject to the privileges set on the server.

Angus Jack is BUG's software engineer and group leader for the South of England, and is rapidly becoming the group expert on Linux. You can contact Angus on angus@richplum.co.uk.