4. Setting up an NFS Client

4.1. Mounting remote directories

Before beginning, you should double-check to make sure your mount program is new enough (version 2.10m if you want to use Version 3 NFS), and that the client machine supports NFS mounting, though most standard distributions do. If you are using a 2.2 or later kernel with the `/proc` filesystem you can check the latter by reading the file `/proc/filesystems` and making sure there is a line containing nfs. If not, typing **`insmod nfs`** may make it magically appear if NFS has been compiled as a module; otherwise, you will need to build (or download) a kernel that has NFS support built in. In general, kernels that do not have NFS compiled in will give a very specific error when the **mount** command below is run.

To begin using machine as an NFS client, you will need the portmapper running on that machine, and to use NFS file locking, you will also need **rpc.statd** and **rpc.lockd** running on both the client and the server. Most recent distributions start those services by default at boot time; if yours doesn't, see Section 3.2 for information on how to start them up.

With **portmap**, **lockd**, and **statd** running, you should now be able to mount the remote directory from your server just the way you mount a local hard drive, with the mount command. Continuing our example from the previous section, suppose our server above is called *master.foo.com*,and we want to mount the `/home` directory on *slave1.foo.com*. Then, all we have to do, from the root prompt on *slave1.foo.com*, is type:

```
# mount master.foo.com:/home /mnt/home
```

and the directory `/home` on master will appear as the directory `/mnt/home` on *slave1*. (Note that this assumes we have created the directory `/mnt/home` as an empty mount point beforehand.)

If this does not work, see the Troubleshooting section (Section 7).

You can get rid of the file system by typing

```
# umount /mnt/home
```

just like you would for a local file system.

4.2. Getting NFS File Systems to Be Mounted at Boot Time

NFS file systems can be added to your `/etc/fstab` file the same way local file systems can, so that they mount when your system starts up. The only difference is that the file system type will be set to **`nfs`** and the dump and fsck order (the last two entries) will have to be set to zero. So for our example above, the entry in `/etc/fstab` would look like:

```
# device       mountpoint     fs-type     options      dump fsckorder
...
master.foo.com:/home  /mnt    nfs          rw            0    0
...
```

See the man pages for `fstab` if you are unfamiliar with the syntax of this file. If you are using an automounter such as amd or autofs, the options in the corresponding fields of your mount listings should look very similar if not identical.

At this point you should have NFS working, though a few tweaks may still be necessary to get it to work well. You should also read Section 6 to be sure your setup is reasonably secure.

4.3. Mount options

4.3.1. Soft vs. Hard Mounting

There are some options you should consider adding at once. They govern the way the NFS client handles a server crash or network outage. One of the cool things about NFS is that it can handle this gracefully. If you set up the clients right. There are two distinct failure modes:

soft

If a file request fails, the NFS client will report an error to the process on the client machine requesting the file access. Some programs can handle this with composure, most won't. We do not recommend using this setting; it is a recipe for corrupted files and lost data. You should especially not use this for mail disks --- if you value your mail, that is.

hard

The program accessing a file on a NFS mounted file system will hang when the server crashes. The process cannot be interrupted or killed (except by a "sure kill") unless you also specify **`intr`**. When the NFS server is back online the program will continue undisturbed from where it was. We recommend using **`hard,intr`** on all NFS mounted file systems.

Picking up the from previous example, the fstab entry would now look like:

```
# device             mountpoint  fs-type    options    dump fsckord
...
master.foo.com:/home  /mnt/home   nfs      rw,hard,intr  0     0
...
```

4.3.2. Setting Block Size to Optimize Transfer Speeds

The **`rsize`** and **`wsize`** mount options specify the size of the chunks of data that the client and server pass back and forth to each other.

The defaults may be too big or to small; there is no size that works well on all or most setups. On the one hand, some combinations of Linux kernels and network cards (largely on older machines) cannot handle blocks that large. On the other hand, if they can handle larger blocks, a bigger size might be faster.

Getting the block size right is an important factor in performance and is a must if you are planning to use the NFS server in a production environment. See Section 5 for details.

Prev Home Next
Setting Up an NFS Server Optimizing NFS Performance