NFS HOWTO

Nicolai Langfeldt janl@math.uio.no v0.5, 9 March 1997

HOWTO set up NFS clients and servers.

Contents

1 Preamble **2**

1.1 Legal stuff . . . 2

1.2 Other stuff . . . 2

1.3 Dedication . . . 2

2 README.first **2**

3 Setting up a NFS server **3**

3.1 Prerequisites . . . 3

3.2 Starting at the path . . . 3

3.3 The portmapper . . . 3

3.4 Mountd and nfsd . . . 3

4 Setting up a NFS client **5**

4.1 Optimizing NFS . . . 6

4.2 Other options . . . 7

5 Security and NFS **7**

5.1 Client Security . . . 8

5.2 Server security: nfsd . . . 8

5.3 Server security: the portmapper . . . 9

5.4 NFS and firewalls . . . 10

5.5 Summary . . . 10

6 FAQs **11**

7 PC-NFS **12**

1 Preamble

1.1 Legal stuff

(C)opyright 1997 Nicolai Langfeldt and Alan Cox. Do not modify without amending copyright, distribute freely but retain copyright message.

1.2 Other stuff

This will never be a finished document, please send me mail about your problems and successes, it can make this a better HOWTO. Please send money, comments and/or questions to janl@math.uio.no. If you send E-mail please *make sure* that the return address is correct and working, I get *a lot* of E-mail and figuring out your e-mail address can be a lot of work. Please.

If you want to translate this HOWTO please notify me so I can keep track of what languages I have been published in :-).

Curses and Thanks to Olaf Kirch who got me to write this and then gave good suggestions for it :-)

1.3 Dedication

This HOWTO is dedicated to Anne Line Norheim. Though she will probably never read it since she's not that kind of girl.

2 README.first

NFS, the Network File System has two important characteristics:

- It makes sharing of files over a network possible.
- It opens a can of security risks that are well understood by crackers, and easily exploited to get access (read, write and delete) to all your files.

I'll say something on both issues in this HOWTO. Please make sure you read the *whole* of this HOWTO, and you will be vulnerable to fewer silly security risks. Don't say I didn't warn you. The passages about security will at times be pretty technical and require some knowledge about IP networking and the terms used. If you don't recognize the terms you can either go back and check the networking HOWTO, wing it, or get a book about TCP/IP network administration to familiarize yourself with TCP/IP. That's a good idea anyway if you're administrating UNIX/Linux machines. A very good book on the subject is *TCP/IP Network Administration* by Craig Hunt, published by O'Reilly & Associates, Inc. And after you've read it and understood it you'll have higher value on the job market, you can't loose ;-)

3 Setting up a NFS server

3.1 Prerequisites

Before you continue reading this HOWTO you will need to be able to telnet back and forth between the machine you're using as server and the client. If that does not work you need to check the networking/NET-2 HOWTO and set up networking properly.

3.2 Starting at the path

Before we can do anything else we need a NFS server set up. If you're part of a department or university network there are likely numerous NFS servers already set up. If they will let you get access to them, or indeed, if you're reading this HOWTO to get access to one of them you obviously don't need to read this section and can just skip ahead to the section on 4

If you need to set up a non-Linux box as server you will have to read the system manual(s) to discover how to enable NFS serving and export of file systems through NFS. Keywords are: nfsd, system administration tool, rc scripts, boot scripts, boot sequence, /etc/exports, exportfs. After you have figured all that out you can continue reading the next section of this HOWTO. Or read more of this section since some of the things I will say are relevant no matter what kind of machine you use as server.

Those of you still reading will need to set up a number of programs.

3.3 The portmapper

The portmapper on Linux is called either `portmap` or `rpc.portmap`. The man page on my system says it is a "DARPA port to RPC program number mapper". It is the first security holes you'll open reading this HOWTO. Description of how to close one of the holes is in the 5. Which I, again, urge you to read.

Start the portmapper. It's either called `portmap` or `rpc.portmap` and it should live in the `/usr/sbin` directory (on some machines it's called rpcbind). You can start it by hand now, but it will need to be started every time you boot your machine so you need to make/edit the rc scripts. Your rc scripts are explained more closely in the init man page, they usually reside in `/etc/rc.d`, `/etc/init.d` or `/etc/rc.d/init.d`. If there is a script called something like `inet` it's probably the right script to edit. But, what to write or do is outside the scope of this HOWTO. Start portmap, and check that it lives by running `ps aux`. It does? Good.

3.4 Mountd and nfsd

The next programs we need running are mountd and nfsd. But first we'll edit another file. `/etc/exports` this time. Say I want the file system `/mn/eris/local` which lives on the machine `eris` to be available to the machine called `apollon`. Then I'd put this in `/etc/exports` on eris:

```
/mn/eris/local  apollon(rw)
```

The above line gives apollon read/write access to `/mn/eris/local`. Instead of `rw` it could say `ro` which means read only (if you put nothing it defaults to read only). There are other options you can give it, and I will discuss some security related ones later. They are all enumerated in the `exports` man page which you should have read at least once in your life. There are also better ways than listing all the hosts in the exports file. You can for example use net groups if you are running NIS (or NYS) (NIS was known as YP), and always specify domain wild cards and IP-subnets as hosts that are allowed to mount something. But you should consider who can get access to the server in unauthorized ways if you use such blanket authorizations.

Note: This exports file is not the same syntax that other Unixes use. Most other Unixes use a format with options separated by commas (,) and host lists colon (:) separated. There is a 4.1. Except for Solaris 2 which is completely different.

Now we're set to start mountd (or maybe it's called `rpc.mountd` and then nfsd (which could be called `rpc.nfsd`). They will both read the exports file.

If you edit `/etc/exports` you will have to make sure nfsd and mountd knows that the files have changed. On most Unix systems the command `exportfs -av` achives this. Except for Solaris 2 which is completely different. Many Linux distributions lack a exportfs program. If you're exportfs-less you can install this script on your machine:

```
#!/bin/sh
killall -HUP /usr/sbin/rpc.mountd
killall -HUP /usr/sbin/rpc.nfsd
echo re-exported file systems
```

Save it in, say, `/usr/sbin/exportfs`, and don't forget to `chmod a+rx` it. Now, whenever you change your exports file, you run exportfs after, as root.

Now you should check that mountd and nfsd are running properly. First with `rpcinfo -p`. It should show something like this:

```
   program vers proto   port
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
    100005    1   udp    745  mountd
    100005    1   tcp    747  mountd
    100003    2   udp   2049  nfs
    100003    2   tcp   2049  nfs
```

As you see the portmapper has announced it's services, and so has mountd and nfsd.

If you get `rpcinfo: can't contact portmapper: RPC: Remote system error - Connection refused` or something similar instead then the portmapper isn't running. Fix it. If you get `No remote programs registered.` then either the portmapper doesn't want to talk to you, or something is broken. Kill nfsd, mountd, and the portmapper and try the ignition sequence again.

After checking that the portmapper reports the services you can check with ps too. The portmapper will continue to report the services even after the programs that extend them have crashed. So a ps check can be smart if something seems broken.

Of course, you will need to modify your system rc files to start mountd and nfsd as well as the portmapper when you boot. It is very likely that the scripts already exist on your machine, you just have to uncomment the critical section or activate it for the correct init run levels.

Man pages you should be familiar with now: portmap, mountd, nfsd, and exports.

Well, if you did everything exactly like I said you should you're all set to start on the NFS client.

4 Setting up a NFS client

First you will need a kernel with the NFS file system either compiled in or available as a module. This is configured before you compile the kernel. If you have never compiled a kernel before you might need to check the kernel HOWTO and figure it out. If you're using a very cool distribution (like Red Hat) and you've never fiddled with the kernel or modules on it (and thus ruined it ;-), nfs is likely automagicaly available to you.

You can now, at a root prompt, enter a appropriate mount command and the file system will appear. Continuing the example in the previous section we want to mount `/mn/eris/local` from eris. This is done with this command:

```
mount -o rsize=1024,wsize=1024 eris:/mn/eris/local /mnt
```

(We'll get back to the rsize and wsize options.) The file system is now available under `/mnt` and you can `cd` there, and `ls` in it, and look at the individual files. You will notice that it's not as fast as a local file system, but a lot more convenient than ftp. If, instead of mounting the file system, mount produces a error message like `mount: eris:/mn/eris/local failed, reason given by server: Permission denied` then the exports file is wrong, or you forgot to run exportfs after editing the exports file. If it says `mount clntudp_create: RPC: Program not registered` it means that nfsd or mountd is not running on the server.

To get rid of the file system you can say

```
umount /mnt
```

To make the system mount a nfs file system upon boot you edit `/etc/fstab` in the normal manner. For our example a line such as this is required:

```
# device      mountpoint     fs-type     options              dump fsckorder
...
eris:/mn/eris/local  /mnt    nfs        rsize=1024,wsize=1024 0    0
...
```

That's all there is too it, almost. Read on please.

4.1 Optimizing NFS

Normally, if no rsize and wsize options are specified NFS will read and write in chunks of 4096 or 8192 bytes. But some combinations of Linux kernels and network cards cannot handle this, and it might not be optimal, anyway. So we'll want to experiment and find a rsize and wsize that works and is as fast as possible. You can test the speed of your options with some simple commands. Given the mount command above and that you have write access to the disk you can do this to test the sequential write performance:

```
time dd if=/dev/zero of=/mnt/testfile bs=16k count=4096
```

This creates a 64Mb file of zeroed bytes. Do it a couple (5-10?) of times and average the times. It is the 'elapsed' or 'wall clock' time that's most interesting in this connection. Then you can test the read performance by reading back the file:

```
time dd if=/mnt/testfile of=/dev/null bs=16k
```

do that a couple of times and average. Then umount, and mount again with a larger rsize and wsize. They should be multiples of 1024, and probably not larger as large as 16384 bytes, since large blocks slows down random accesses. Directly after mounting with a larger size cd into the mounted file system and do things like ls, explore the fs a bit to make sure everything is as it should. If the rsize/wsize is too large the symptoms are *very* odd and not 100% obvious. A typical symptom is incomplete file lists when doing 'ls' with no error messages. Or reading files failing mysteriously with no error messages. After establishing that the given rsize/wsize works you can do the speed tests again. Different server platforms are likely to have different optimal sizes. SunOS and Solaris is reputedly a lot faster with 4096 byte blocks than with anything else.

Newer Linux kernels (since 1.3 sometime) perform read ahead for rsizes larger or equal to the machine page size. On Intel cpus the page size is 4096 bytes. Read ahead will *significantly* increase the NFS read performance. So on a Intel machine you will want 4096 byte rsize if at all possible.

Remember to edit `/etc/fstab` to reflect the rsize/wsize you found.

A trick to increase NFS write performance is to disable synchronous writes on the server. The NFS specification states that NFS write requests shall not be considered finished before the data written is on a non-volatile medium (normally the disk). This restricts the write performance somewhat, and asynchronous writes will speed NFS writes up. The Linux nfsd has never done synchronous writes since the Linux file system implementation does not lend itself to this, but on non-Linux servers you can increase the performance this way with this in your exports file:

```
/dir    -async,access=linuxbox
```

or something similar. Please refer to the exports man page on the machine in question.

4.2 Other options

There are some other options you should consider adding at once. The govern the way the NFS client handles a server crash. There are two distinct failure modes:

soft

The NFS client will report and error to the process accessing a file on a NFS mounted file system. Some programs can handle this with composure, most won't. I cannot recommend using this setting.

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 unless you also specify `intr`. When the NFS server is back online the program will continue undisturbed from where it were. This is probably what you want. I recommend using `hard,intr` on all NFS mounted file systems.

Picking up the previous example, assuming that you found 4096 bytes to be optimal read/write sizes, this is now your fstab entry:

```
# device      mountpoint     fs-type    options                  dump fsckorder
...
eris:/mn/eris/local  /mnt    nfs        rsize=4096,wsize=4096,hard,intr 0 0
...
```

5 Security and NFS

I am by no means a computer security expert. But I've been around the block so I have a *little* advice for the security conscious. But be warned: This is by no means a complete list of NFS related problems and if you think you're safe once you're read and implemented all this I have a bridge I want to sell you.

This section is probably of no concern if you are on a *closed* network where you trust all the users, and no-one you don't trust can get access to machines on the network. I.e., there should be no way to dial into the network, and it should in no way be connected to other networks where you don't trust everyone using it as well as the security. Do you think I sound paranoid? I'm not at all paranoid. This is just *basic* security advice. And remember, the things I say here is just the start of it. A *secure* site needs a diligent and knowledgeable admin that knows where to find information about current and potential security problems.

NFS has a basic problem in that the client, if not told otherwise, will trust the NFS server and vice versa. This can be bad. That means that if the server's root account is broken into it can be quite easy to break into the client's root account as well. And vice versa. There are a couple of coping strategies for this, which we'll get back to.

Something you should read is the CERT advisories on NFS, most of the text below deals with issues CERT has written advisories about. See ftp.cert.org/01-README for a up to date list of CERT advisories. Here are some NFS related advisories:

```
CA-91:21.SunOS.NFS.Jumbo.and.fsirand                                12/06/91
     Vulnerabilities concerning Sun Microsystems, Inc. (Sun) Network
     File System (NFS) and the fsirand program.  These vulnerabilities
     affect SunOS versions 4.1.1, 4.1, and 4.0.3 on all architectures.
     Patches are available for SunOS 4.1.1.  An initial patch for SunOS
     4.1 NFS is also available. Sun will be providing complete patches
     for SunOS 4.1 and SunOS 4.0.3 at a later date.

CA-94:15.NFS.Vulnerabilities                                        12/19/94
     This advisory describes security measures to guard against several
     vulnerabilities in the Network File System (NFS). The advisory was
     prompted by an increase in root compromises by intruders using tools
     to exploit the vulnerabilities.

CA-96.08.pcnfsd                                                     04/18/96
     This advisory describes a vulnerability in the pcnfsd program (also
     known as rpc.pcnfsd). A patch is included.
```

5.1 Client Security

On the client we can decide that we don't want to trust the server too much a couple of ways with options to mount. For example we can forbid suid programs to work off the NFS file system with the `nosuid` option. This is a good idea and you should consider using this with all NFS mounted disks. It means that the server's root user cannot make a suid-root program on the file system, log in to the client as a normal user and then use the suid-root program to become root on the client too. We could also forbid execution of files on the mounted file system altogether with the `noexec` option. But this is more likely to be impractical than `nosuid` since a file system is likely to at least contain *some* scripts or programs that needs to be executed. You enter these options in the options column, with the `rsize` and `wsize`, separated by commas.

5.2 Server security: nfsd

On the server we can decide that we don't want to trust the client's root account. We can do that by using the root_squash option in exports:

```
/mn/eris/local apollon(rw,root_squash)
```

Now, if a user with UID 0 on the client attempts to access (read, write, delete) the file system the server substitutes the UID of the servers 'nobody' account. Which means that the root user on the client can't access or change files that only root on the server can access or change. That's good, and you should probably use `root_squash` on all the file systems you export. "But the root user on the client can still use 'su' to become any other user and access and change that users files!" say you. To which the answer is: Yes, and that's the way it is, and has to be with Unix. This has one important implication: All important binaries and files should be owned by `root`, and not `bin` or other non-root account, since the only account

the clients root user cannot access is the servers root account. In the nfsd man page there are several other squash options listed so that you can decide to mistrust whomever you (don't) like on the clients. You also have options to squash any UID and GID range you want to. This is described in the Linux nfsd man page.

Another important thing is to ensure that nfsd checks that all it's requests comes from a privileged port. If it accepts requests from any old port on the client a user with no special privileges can run a program that's is easy to obtain over the Internet. It talks nfs protocol and will claim that the user is anyone the user wants to be. Spooky. The Linux nfsd does this check by default, on other OSes you have to enable this check yourself. This should be described in the nfsd man page for the OS.

Another thing. Never export a file system to 'localhost' or 127.0.0.1. Trust me.

5.3 Server security: the portmapper

The basic portmapper, in combination with nfsd has a design problem that makes it possible to get to files on NFS servers without any privileges. Fortunately the portmapper Linux uses is relatively secure against this attack, and can be made more secure by configuring up access lists in two files.

First we edit `/etc/hosts.deny`. It should contain the line

```
portmap: ALL
```

which will deny access to *everyone*. That's a bit drastic perhaps, so we open it again by editing `/etc/hosts.allow`. But first we need to figure out what to put in it. It should basically list all machines that should have access to your portmapper. On a run of the mill Linux system there are very few machines that need any access for any reason. The portmapper administrates nfsd, mountd, ypbind/ypserv, pcnfsd, and 'r' services like ruptime and rusers. Of these only nfsd, mountd, ypbind/ypserv and perhaps pcnfsd are of any consequence. All machines that needs to access services on your machine should be allowed to do that. Let's say that your machines address is 129.240.223.254 and that it lives on the subnet 129.240.223.0 should have access to it (those are terms introduced by the networking HOWTO, go back and refresh your memory if you need to). Then we write

```
portmap: 129.240.223.0/255.255.255.0
```

in `hosts.allow`. This is the same as the network address you give to route and the subnet mask you give to ifconfig. For the device `eth0` on this machine `ifconfig` should show

```
...
eth0      Link encap:10Mbps Ethernet  HWaddr 00:60:8C:96:D5:56
          inet addr:129.240.223.254  Bcast:129.240.223.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:360315 errors:0 dropped:0 overruns:0
          TX packets:179274 errors:0 dropped:0 overruns:0
          Interrupt:10 Base address:0x320
...
```

and `netstat -rn` should show

```
Kernel routing table
Destination     Gateway         Genmask         Flags Metric Ref Use    Iface
...
129.240.223.0   0.0.0.0         255.255.255.0   U     0      0   174412 eth0
...
```

(Network address in first column).

The `hosts.deny` and `hosts.allow` files are described in the manual pages of the same names.

IMPORTANT: Do *not* put *anything* but *IP NUMBERS* in the portmap lines of these files. Host name lookups can indirectly cause portmap activity which will trigger host name lookups which can indirectly cause portmap activity which will trigger...

The above things should make your server tighter. The only remaining problem (Yeah, right!) is someone breaking root (or boot MS-DOS) on a trusted machine and using that privilege to send requests from a secure port as any user they want to be.

5.4 NFS and firewalls

It's a very good idea to firewall the nfs and portmap ports in your router or firewall. The nfsd operates at port 2049, both udp and tcp protocols. The portmapper at port 111, tcp and udp, and mountd at port 745 and and 747, tcp and udp. Normally. You should check the ports with the `rpcinfo -p` command.

If on the other hand you want NFS to go through a firewall there are options for newer nfsds and mountds to make them use a specific (nonstandard) port which can be open in the firewall.

5.5 Summary

If you use the hosts.allow/deny, root_squash, nosuid and privileged port features in the portmapper/nfs software you avoid many of the presently known bugs in nfs and can almost feel secure about *that* at least. But still, after all that: When an intruder has access to your network, s/he can make strange commands appear in your .forward or mailbox file when `/home` or `/var/spool/mail` are mounted over NFS. For the same reason, you should never access your PGP private key over nfs. Or at least you should know the risk involved. And now you know a bit of it.

NFS and the portmapper makes up a complex subsystem and therefore it's not totally unlikely that new bugs will be discovered, either in the basic design or the implementation we use. There might even be holes known now, which someone is abusing. But that's life. To keep abreast of things like this you should at least read the newsgroups comp.os.linux.announce and comp.security.announce at a absolute minimum.

6 FAQs

This is the FAQ section. Most of it was written by Alan Cox.

1. When I try to mount a file system I get

   ```
   can't register with portmap: system error on send
   ```

   You are probably using a Caldera system. There is a bug in the rc scripts. Please contact Caldera to obtain a fix.

2. Why can't I execute a file after copying it to the NFS server?

   The reason is that nfsd caches open file handles for performance reasons (remember, it runs in user space). While nfsd has a file open (as is the case after writing to it), the kernel won't allow you to execute it. Nfsds newer than spring 95 release open files after a few seconds, older ones would cling to them for days.

3. My NFS files are all read only

   The Linux NFS server defaults to read only. RTFM the "exports" and nfsd manual pages. You will need to alter `/etc/exports`.

4. I mount from a linux nfs server and while ls works I can't read or write files.

   On older versions of Linux you must mount a NFS servers with `rsize=1024,wsize=1024`.

5. I mount from a Linux NFS server with a block size of between 3500-4000 and it crashes the Linux box regularly

   Basically don't do it then.

6. Can Linux do NFS over TCP

   No, not at present.

7. I get loads of strange errors trying to mount a machine from a Linux box.

   Make sure your users are in 8 groups or less. Older servers require this.

8. When I reboot my machine it sometimes hangs when trying to unmount a hung NFS server.

   Do **not** unmount NFS servers when rebooting or halting, just ignore them, it will not hurt anything if you don't unmount them. The command is `umount -avt nonfs`.

9. Linux NFS clients are very slow when writing to Sun & BSD systems

   NFS writes are normally synchronous (you can disable this if you don't mind risking losing data). Worse still BSD derived kernels tend to be unable to work in small blocks. Thus when you write 4K of data from a Linux box in the 1K packets it uses BSD does this

```
read 4K page
alter 1K
write 4K back to physical disk
read 4K page
alter 1K
write 4K page back to physical disk
etc..
```

Better systems don't have this problem. The Linux client is however quite slow anyway.

10. I've heard NFS is not secure is this true?

    Yes, totally. Running NFS in an uncontrolled environment is rather like leaving your front door open, painting 'On holiday' on your house and posting maps to every known criminal... In a fairly secure environment or when you can recover data from stupid misuse its pretty much OK. The worst someone can easily do is alter all the files on an NFS mounted disk and/or crash the machine. So long as you don't mount your system files writable you should be vaguely safe.

11. Why do we use NFS at all then?

    Because it's the only uniformly supported file sharing protocol for Unix. And because it works, mostly.

12. How do I stop things hanging when a server goes down

    There are three main NFS behaviors on server down:

    soft

    Your NFS client will report an error to the process concerned if an NFS server doesn't answer after a few retries. If you want to use this you should check that your software handles it gracefully. I would not recommend using this setting.

    hard

    Your NFS client will try forever unless killed off. Operations will be restarted when the NFS server recovers or reboots.

    hard,intr

    As hard but Ctrl-C will interrupt a hung process. In a few cases - notably nfs mounted /usr/spool/mail disks this doesn't help as the shell will be ignoring Ctrl-C when it checks you have mail... I recommend this setting for **all** NFS mounts, including the mail spool.

7 PC-NFS

Some things are best left undocumented :-) But you definitely want to run pcnfsd, if you want to run PC-NFS. Good luck.

I know, that's not exactly friendly, but I'm not familiar with PC NFS. If you are please contact me so we can maybe get in something about PC-NFS here.