NFS HOWTO

Table of Contents

Table of Contents

NFS HOWTO

Nicolai Langfeldt `janl@linpro.no`

v1.0, 1 October 1999

HOWTO set up NFS clients and servers.

1.Preamble

2.README.first

3.Setting up a NFS server

4.Setting up a NFS client

5.NFS over slow lines

6.Security and NFS

7.Mount Checklist

8.FAQs

9.Exporting filesystems

10.NFS under Linux 2.2

11.NFS server on a floppy

12.PC−NFS

1.Preamble

1.1 Legal stuff

 The FAQ section is based on a NFS FAQ compiled by Alan Cox. The Checklist section is based on a mount problem checklist compiled by the IBM Corporation. The nfs−server−on−a−floppy section was written by Ron Peters.

1.2 Disclaimer

Neither Nicolai Langfeldt, Ron Peters, nor their employers, or anyone else, take any responsebility for what might happen if you follow the instructions in this document. If you choose to follow the instructions in any case, good luck!

1.3 Feedback

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,or rpeters@hevanet.com in the case of NFS server on a floppy things. If you send E−mail and want an answer please show the simple courtecy of *making sure* that the return address is correct and working. You have no idea how many answers have bounced.

1.4 Other stuff

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 :−)

1.5 Dedication

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

---

2.README.first

NFS, the Network File System has three important characteristics:

- It makes sharing of files over a network possible.
- It mostly works well enough.
- 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 security section of this HOWTO, and you will be vulnerable to fewer silly security risks. 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 ;–)

There are two sections to help you troubleshoot NFS, called *Mount Checklist* and *FAQs*. Please refer to them if something dosen't work as advertized.

The home–site for the Linux 2.0 nfsd is ftp.mathematik.th–darmstadt.de:/pub/linux/okir, in case you want/need to get it and compile it yourself.

For information about NFS under Linux 2.2 please see the Linux 2.2 section.

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−3 HOWTO and set up networking properly.

3.2 First step

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 setting up a NFS client

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. There is a separate section in this HOWTO on how to do it on many different systems. 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.

If you're running please see the Linux 2.2 section before you continue reading this.

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 hole you'll open reading this HOWTO. Description of how to close one of the holes is in the security section. 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` and then `rpcinfo −p`. It does? Good.

Oh, one thing. Remote access to your portmapper is regulated by the contents of your `/etc/hosts.allow` and `/etc/hosts.deny` files. If `rpcinfo -p` fails, but your portmapper is running please examine these files. See the security section for details on these files.

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. There is a separate section in this HOWTO about other Unixes `exports` files.

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. The traditonal way is to run `exportfs`. 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`, `RPC_PROG_NOT_REGISTERED` or something similar instead then the portmapper isn't running. OR you might have something in `/etc/hosts.{allow,deny}` that forbids the portmapper from answering, please see the security section for details on these files. 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. Or you have the `hosts.{allow,deny}` problem mentioned earlier.

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 Mount options

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

The NFS client will report an 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, it is a recepie 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 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, this is now your fstab entry:

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

4.2 Optimizing NFS

Normally, if no rsize and wsize options are specified NFS will read and write in chunks of 4096 or 8192 bytes. Some combinations of Linux kernels and network cards cannot handle that large blocks, 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 (which should be large enough that caching is no significant part of any performance perceived, use a larger file if you have a lot of memory). 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 probably be multiples of 1024, and not larger than 16384 bytes since that's the maximum size in NFS version 2. 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', and 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, 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. Please note that this increases the risk of data loss.

5.NFS over slow lines

Slow lines include Modems, ISDN and quite possibly other long distance connections.

This section is based on knowledge about the used protocols but no actual experiments. Please let me hear from you if try this ;−)

The first thing to remember is that NFS is a slow protocol. It has high overhead. Using NFS is almost like using kermit to transfer files. It's *slow*. Almost anything is faster than NFS. FTP is faster. HTTP is faster. rcp is faster. ssh is faster.

Still determined to try it out? Ok.

NFS' default parameters are for quite fast, low latency, lines. If you use these default parameters over high latency, slow, lines it can cause NFS to report errors, abort operations, pretend that files are shorter than they really are, and act mysteriously in other ways.

The first thing to do is *not* to use the `soft` mount option. This will cause timeouts to return errors to the software, which will, most likely not handle the situation at all well. This is a good way to get mysterious failures. Instead use the `hard` mount option. When `hard` is active timeouts causes infinite retries instead of aborting whatever it was the software wanted to do. This is what you want. Really.

The next thing to do is to tweak the `timeo` and `retrans` mount options. They are described in the nfs(5) man page, but here is a copy:

```
       timeo=n        The  value  in  tenths  of  a second before
                      sending the first retransmission  after  an
                      RPC timeout.  The default value is 7 tenths
```

```
                              of a second.  After the first timeout,  the
                              timeout  is  doubled  after each successive
                              timeout until a maximum timeout of 60  sec-
                              onds  is  reached or the enough retransmis-
                              sions have occured to cause a  major  time-
                              out.   Then,  if  the  filesystem  is  hard
                              mounted, each new timeout cascade  restarts
                              at  twice the initial value of the previous
                              cascade, again doubling at each retransmis-
                              sion.   The  maximum  timeout  is always 60
                              seconds.  Better overall performance may be
                              achieved  by  increasing  the  timeout when
                              mounting on  a  busy  network,  to  a  slow
                              server, or through several routers or gate-
                              ways.

               retrans=n      The number of minor timeouts  and  retrans-
                              missions  that  must  occur  before a major
                              timeout occurs.  The default is 3 timeouts.
                              When a major timeout occurs, the file oper-
                              ation is either aborted or  a  "server  not
                              responding"  message is printed on the con-
                              sole.
```

In other words: If a reply is not received within the 0.7 second (700ms) timeout the NFS client will repeat the request and double the timeout to 1.4 seconds. If the reply does not appear within the 1.4 seconds the request is repeated again and the timeout doubled again, to 2.8 seconds.

A lines speed can be measured with ping with the same packet size as your rsize/wsize options.

```
$ ping -s 8192 lugulbanda
PING lugulbanda.uio.no (129.240.222.99): 8192 data bytes
8200 bytes from 129.240.222.99: icmp_seq=0 ttl=64 time=15.2 ms
8200 bytes from 129.240.222.99: icmp_seq=1 ttl=64 time=15.9 ms
8200 bytes from 129.240.222.99: icmp_seq=2 ttl=64 time=14.9 ms
8200 bytes from 129.240.222.99: icmp_seq=3 ttl=64 time=14.9 ms
8200 bytes from 129.240.222.99: icmp_seq=4 ttl=64 time=15.0 ms

--- lugulbanda.uio.no ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 14.9/15.1/15.9 ms
```

The time here is how long the ping packet took to get back and forth to lugulbanda. 15ms is quite fast. Over a 28.000 bps line you can expect something like 4000−5000ms, and if the line is otherwise loaded this time will be even higher, easily double. When this time is high we say that there is 'high latency'. Generally, for larger packets and for more loaded lines the latency will tend to increase. Increase `timeo` suitably for your line and load. And since the latency increases when you use the line for other things: If you ever want to use FTP and NFS at the same time you should try measuring ping times while using FTP to transfer files and increase `timeo` to match your line latency.

6.Security and NFS

I am by no means a computer security expert. But I do 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. It 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.
```

6.1 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.

6.2 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 and NFS. 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.

root_squash is in fact the default with the Linux NFSd, to grant root access to a filesystem use `no_root_squash`.

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.

6.3 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 that most Linux distributions use is relatively secure against this attack, and can be made more secure by configuring up access lists in two files.

Not all Linux distributions were created equal. Some seemingly up−to−date distributions does *not* include a securable portmapper, even today, many years since the vulnerability became common knowledge. At least one distribution even contains the manpage for a securable portmapper but the actual portmapper is *not* secureable. The easy way to check if your portmapper is good or not is to run strings(1) and see if it reads the relevant files, `/etc/hosts.deny` and `/etc/hosts.allow`. Assuming your portmapper is `/usr/sbin/portmap` you can check it with this command: `strings /usr/sbin/portmap | grep hosts`. On my machine it comes up with this:

```
/etc/hosts.allow
/etc/hosts.deny
@(#) hosts_ctl.c 1.4 94/12/28 17:42:27
@(#) hosts_access.c 1.20 96/02/11 17:01:27
```

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

```
portmap: ALL
```

which will deny access to *everyone*. While it is closed thus run `rpcinfo −p` just to check that your portmapper really reads and obeys this file. rpcinfo should give no output, or possebly a errormessage. Restarting the portmapper should *not* be necessary.

Closing the portmapper for everyone is a bit drastic, 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.

6.4 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.

6.5 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 read your mail when `/home` or `/var/spool/mail` is NFS exported. 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.

7.Mount Checklist

This section is based on IBM Corp.s ``NFS mount problem checklist''. Many thanks to them for making it available for this HOWTO. If you experience a problem mounting a NFS filesystem please refer to this list before posting your problem. Each item describes a failure mode and the fix.

1. Mount keeps saying `RPC: Program not registered`

   Is the portmapper running?

   Fix: Start it.

   Is mountd running?

   Fix: Start it.

   Is nfsd running?

   Fix: Start it.

   Is the portmapper forbidden to answer by `/etc/hosts.deny`?

   Fix: Either remove the rule in `hosts.deny` or add a rule to `hosts.allow` such that the portmapper is allowed to talk to you.
2. File system not exported, or not exported to the client in question.

   Fix: Export it
3. Name resolution doesn't jibe with the exports list.

   e.g.: export list says export to `johnmad` but `johnmad`'s name is resolved as `johnmad.austin.ibm.com`. mount permission is denied.

   Fix: Export to both forms of the name.

   It can also happen if the client has two interfaces with different names for each of the two and the export file only specifies one.

Fix: export both interfaces.

This can also happen if the server can't do a lookuphostbyname or lookuphostbyaddr (these are library functions) on the client. Make sure the client can do host <name>; host <ip_addr>; and that both shows the same machine.

Fix: straighten out name resolution.

4. The file system was mounted after NFS was started (on that server). In that case the server is exporting underlying mount point, not the mounted filesystem.

   Fix: Shut down NFSd and then restart it.

   Note: The clients that had the underlying mount point mounted will get problems accessing it after the restart.
5. The date is wildly off on one or both machines (this can mess up make)

   Fix: Get the date set right.

   The HOWTO author recommends using NTP to synchronize clocks. Since there are export restrictions on NTP in the US you have to get NTP for Debian, Red Hat or Slackware from `ftp://ftp.hacktic.nl/pub/replay/pub/linux` or a mirror.
6. The server can not accept a mount from a user that is in more than 8 groups.

   Fix: decrease the number of groups the user is in or mount via a different user.

8.FAQs

This is the FAQ section. It is partly based on a old NFS FAQ by Alan Cox.

If you have a problem mounting a filesystem please see if your problem is described in the ``Mount Checklist'' section.

1. I get a lot of ``stale nfs handle'' errors when using Linux as a nfs server.

   This is caused by a bug in some old nfsd versions. It is fixed in nfs−server2.2beta16 and later.
2. 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.

3. 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.
4. My NFS files are all read only

   The Linux NFS server defaults to read only. Please read the section about ``Mountd and nfsd'' and ``Exporting filesystems'' in this HOWTO, and refer to the ``exports'' and ``nfsd'' manual pages. You will need to alter `/etc/exports`.
5. 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`.
6. 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. This does not happen with 2.0 and 2.2 kernels. As far as I recall there is no problem with 1.2 either.
7. Can Linux do NFS over TCP

   No, not at present.
8. 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.
9. 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`.
10. Linux NFS clients are very slow when writing to Sun and 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..
```

11. When I connect many clients to a Linux NFS server the performance suddenly drops.

    The NFS protocol uses fragmented UDP packets. The kernel has a limit of how many fragments of incomplete packets it can have before it starts throwing away packets. In 2.2 this is runtime tuneable via the /proc filesystem: `/proc/sys/net/ipv4/ipfrag_high_thresh` and `ipfrag_low_thresh`. In 2.0 these are compile−time constants defined in `.../linux/net/ipv4/ip_fragment.c`, `IPFRAG_HIGH_THRESH` and `IPFRAG_LOW_THRESH`. The meaning of these values is that once the memory consumption of unassembled UDP fragments reaches the ``ipfrag_high_thresh'' in bytes (256K by default in 2.2.3 and 2.0.36) it is cut down to ``ipfrag_low_tresh'' at once. This is done by throwing away fragments. This will look almost like packet loss, and if the high threshold is reached your server performance drops a lot.

    256K is enough for up to 30 clients. If you have 60, double it. And double the low threshold also.

12. I'm using Linux 2.2 (or later) with knfsd and I can't get my AIX, IRIX, Solaris, DEC−Unix, ... machine to mount it.

    Knfsd announces that it implements NFS version 3. It does not. There is an option to stop it from announcing it. Use it. Or you can put "`vers=2`" in the mount option list on the clients.

13. My AIX 4 machine cannot mount my Linux NFS server. It says

    ```
    mount: 1831−011 access denied for server:/dir
    mount: 1831−008 giving up on:
    server:/dir
    The file access permissions do not allow the specified action.
    ```

    or something like that instead.

    AIX 4.2 used reserved ports (<1024) for NFS. AIX 4.2.1 and 4.3 are not constrained to reserved ports. Also, AIX 4.2.1 and 4.3 try to mount using NFS3, then NFS/TCP, then fiannly NFS/UDP.

    Adding

    ```
    nfso −o nfs_use_reserved_ports=1
    ```

to the end of `rc.tcpip` will force it to use reserved ports again. (This tip was supplied by Brian Gorka)

9.Exporting filesystems

The way to export filesytems with NFS is not completely consistent across platforms of course. In this case Linux and Solaris 2 are the deviants. This section lists, superficially, the way to do it on most systems. If the kind of system you have is not covered you must check your OS man−pages. Keywords are: nfsd, system administration tool, rc scripts, boot scripts, boot sequence, /etc/exports, exportfs. I'll use one example throughout this section: How to export /mn/eris/local to apollon read/write.

9.1 IRIX, HP−UX, Digital−UNIX, Ultrix, SunOS 4 (Solaris 1), AIX

These OSes use the traditional Sun export format. In `/etc/exports` write:

```
/mn/eris/local -rw=apollon
```

The complete documentation is in the `exports` man page. After editing the file run `exportfs -av` to export the filesystems.

How strict the exportfs command is about the syntax varies. On some OSes you will find that previously entered lines reads:

```
/mn/eris/local apollon
```

or even something degenerate like:

```
/mn/eris/local rw=apollon
```

I recommend being formal. You risk that the next version of `exportfs` if much stricter and then suddenly everything will stop working.

9.2 Solaris 2

Sun completely re−invented the wheel when they did Solaris 2. So this is completely different from all other OSes. What you do is edit the file `/etc/dfs/dfstab`. In it you place share commands as documented in the `share`(1M) man page. Like this:

```
share −o rw=apollon −d "Eris Local" /mn/eris/local
```

After editing run the program `shareall` to export the filesystems.

10.NFS under Linux 2.2

As I write this Linux 2.2.12 is the current kernel version and to use NFS under it can be a bit of a chore. Or not.

What the status of NFS in Linux 2.4 will be i unknown.

The new big thing in Linux 2.2 is support for a in−kernel nfs server demon, called knfsd in 2.2. This way of implementing nfsd has some advantages, the main one is speed. A Linux 2.2 machine with knfsd is a respectable nfs server. You can still use the old nfsd with Linux 2.2 though, and there are some advantages to using this, mainly simplicity.

If you use a kernel source or binary package made by someone like RedHat (6.0 and later), SuSE (6.1 or later, I belive) or some other professional system integrator they have likely integrated full "knfsd" functionality in their kernel and you need not worry, it will work. Mostly. Until you want to compile a kernel yourself. If you use a stock Linux 2.2 kernel (up to 2.2.12 at least) knfsd will break.

To get this on the air yourself you need to get H.J. Lus knfsd package. This is a collection of patches, and the needed utilities for 2.2 that Lu is maintaining in his spare time. You can get it from your local kernel mirror, the master site is ftp.kernel.org:/pub/linux/devel/gcc/. **This is not meant for general consumption**. If you find this package confusing please don't try to do this yourself. Wait until a kernel package from your favourite system integrator (e.g., Red Hat, SuSE or ...) appears.

Also, please don't send me questions about this, I can't help you. I do not have any knfsd based servers running. If you find errors or omissions in this documentation, please write to me and I'll revise this HOWTO and release it again.

Still reading? Ok. H.J.Lu posts about new versions of this package on the linux−kernel mailing list. Other issues pertaining to NFS in 2.2 is also posted about there. Read it.

There is one interesting thing to note about the knfsd package. It announces that it supports NFS version 3. However it does not support it. There is an option you can give to stop it from announcing NFS3, or on the clients you can specify "`vers=2`" in the mount option list.

10.1 The client

The client is almost simple. To get propper locking you need to get `statd` (from the knfsd package) compiled, installed and started from your boot−scripts. Do that. Statd needs a directory called `/var/lib/nfs` to function otherwise it will just abort with no error message, so that directory needs to be created before it will run.

Once statd is running you can use the `testlk` program (in `tools/locktest` to test if locking of a file on a NFS mounted filesystem works. It should. If it prints *No locks available* statd is not working.

Actually, you can also avoid locking entierly (not that I recomend this), by giving "`nolock`" in the mount option list.

As far as I know this is all that's needed to get the client working.

Oh, if you have a Sparc or Alpha NFS server you will find that the nfs client in Linux 2.2 absolutely sucks. The transfer rates to and from the server is so bad that ... you can't imagine. It's far worse than under Linux 2.0. Far. But there is a fix for this of course. The Alan Cox series of 2.2 kernels (which are a bit more experimental than the normal 2.2 kernels from Linus) include a patch to make Linux 2.2 perform when used with Alpha and Sparc servers. If you want to use the Alan Cox 2.2 kernels you should be reading the linux−kernel mailing list and if you do you know where the patch can be found. There home site of this patch is http://www.uio.no/~trondmy/src/, in case you want to try to apply it to a stock 2.2 kernel. This patch will probably not be in Linux 2.4 either, because it requires too many changes in the kernel to be accepted in the current development cycle. Wait for Linux 2.5.

`trondmy` also has patches to make Linux use NFS version 3, this will also enable you to use tcp as transport mechanism instead of UDP. NFSv3 is is very good for long−haul networks and other networks where the packet loss is non−zero or the latencies are high.

The reason you should read the linux−kernel mailing list to use these patches is that sometimes there are bad bugs discovered in them. Bugs that eat your files. So please **beware**.

10.2 The server

The nfs server demon under Linux 2.2 and later is called "`knfsd`". It is tricky to set it up. You have to figure this out all by yourself, or stick to what SuSE, Red Hat and others are releasing in the way of 2.2 kernel packages. Sorry. You can still use the old nfsd under Linux 2.2 though. It's slow but easy to set up.

11.NFS server on a floppy

This section was written by Ron Peters, rpeters@hevanet.com It explains how to set up an NFS server when booting up from floppy. It was originally devised to be able to NFS share a cdrom from another non−Linux/UNIX machine to install Linux on a machine that does not have a cdrom.

11.1 Introduction

This document is being created for those who will run into the same problem I had recently. I was building a Linux server on a machine that didn't have a cdrom and has no facility for adding one except for possibly an external SCSI or the like. Now that it is getting less and less likely that you will be installing on a machine like that, this document may not be that valuable. However, I would have appreciated it when I was trying to build my machine.

Since my machine didn't have a cdrom drive, I thought I would go find an NFS server for Win95 and share the cdrom for long enough to install the box and get it on my network. Of the two products I found, (I'm not mentioning names but one was freeware and the other was a 14 day limited license), one didn't work out of the box, and the other couldn't handle the Linux naming convention well enough to complete the install.

I then settled on trying to boot my Win95 machine with the boot/root set of disks and then use a suplimentary floppy to set up the NFS server.

This was remarkably simple, and the procedure is probably easier than reading this introduction but I believe that putting the whole procedure in one place will be value added.

11.2 Expectations

This document was derived using the boot/root disks from one of the current InfoMagic developer distributions of Slackware. I used kernel version 2.0.34 for the boot/root disks, but the NFS server programs were taken from a 2.0.30 server. I have always used the Slakware installation method, not because it is any easier or better or worse, just that I am comfortable with it and I haven't taken the time to try another method.

I don't believe that there will be many problems using this document in relation to OS version. I would recommend using something relatively current. Since it is likely that this will be used for installation, a current boot/root set will likely be used.

Your mileage may vary.

11.3 Requirements

- Network capable system and boot disk. The system that is to be the NFS server must have a network card and it must be recognized by the during the boot process. More information on this can be found in the Networking HOWTO.
- Secondary floppy that contains rpc.portmap, rpc.mountd and rpc.nfsd. These files should be easily found from an ftpsearch off the web.
- Slackware (or other) source media (assumed to be cd).

11.4 Server Setup

Boot the temporary NFS server

Boot the NFS server system from boot floppy and make sure the network card is recognized. It is also necessary that the CDROM be recognized. I will use eth0 as the example network card.

Mount the floppy and cdrom

Once the system is booted up, the boot/root floppies are not needed. The system is fully contained in RAM.

Replace the root floppy with the suplimentary disk. Mount the floppy:

```
mount /dev/fd0 /floppy
```

This assumes that the floppy is an ext2 file system type. I imaging that the suplimentary disk could be a DOS floppy with the files on it, but I haven't tried that yet. I imagine that this would be easier that a disk image. In this case, it would be a `mount -t msdos ...etc`. This should probably be put in the todo section.

Mount the cdrom:

```
mount -t iso9660 /dev/hdc /cdrom
```

The floppy and cdrom devices are the ones I used. These may be different depending on application. The mount points /floppy and /cdrom exist on the root floppy disk image so they can be used. If they don't, create them or you could use any mount points you like.

Set up networking on the temporary server.

This is where the temporary NFS server is set up to talk on the network. There are only a few commands to run. There are a few items of information that you will need before running the commands (values are examples):

IPADDR:172.16.5.100 #This is the address of the temporary server.

NETMASK:255.255.255.0 #This is the netmask.

BROADCAST:172.16.5.255 #The last number (255) is significant from IPADDR.

ETHNETWORK:172.16.5.0 #Once again, slightly different from IPADDR.

GATEWAY:172.16.5.251 #Only needed if you have a gateway. You will probably know. Most home networks won't have a gateway.

The commands to get on the network. Insert values from above:

```
ifconfig eth0 inet IPADDR arp netmask NETMASK broadcast BROADCAST
```

```
route add -net ETHNETWORK netmask NETMASK eth0
```

Only use next command if you have a gateway and need to go through it:

```
route add default gw GATEWAY netmask 0.0.0.0 eth0
```

If all goes well, you are now on the network and should be able to ping other nodes.

Set up the NFS share.

Determine the directory that you want to NFS share. In the case of the my example, I used the /cdrom/slakware directory. Put this directory in the /etc/exports file:

```
echo "/cdrom/slakware" > /etc/exports
```

11.5 Run the NFS server

Go to /floppy/usr/sbin and run:

```
./rpc.portmap
```

```
./rpc.mountd
```

```
./rpc.nfsd
```

Complete, start the install.

This should share the "/cdrom/slakware" directory in the /etc/exports file. Once this is done, you can now boot up the machine to be installed from boot/root floppies (I used same ones that I booted NFS server with) and start the installation.

Once you are ready to choose the media source location, choose the NFS server option. It will ask about the ip address of the server. Give it the IP address that you used as IPADDR for the server. It will also ask for the directory to be mounted. This is the directory you put in the /etc/exports on the NFS server.

The system will then NFS mount the server. Watch for any error messages. All should be complete and you can continue the installation.

11.6 Troubleshooting

Nothing Here Yet.

I don't have any troubleshooting info yet. Perhaps as people use this procedure, there will be more tips and hints available.

11.7 To Do

DOS Disk.

Check out a DOS disk for the suplimentary disk.

rpc commands.

Check out specific order of running rpc.* commands and if all or just some of the command needs to be run.

12.PC−NFS

You don't want to run PC−NFS. You want to run samba.

samba is far better than PC−NFS and it works with Windows 3 for Workgroups and later versions of

Windows. It's faster and more secure too. Use it. Really.