62 - NFS - Network File System

NFS Server:
Paket `nfsserv` Serie `n`
Konfigurationsdatei: (`rc.config` through `yast`)
`NFS_SERVER` und `NF_SERVER_UGID` auf »yes« setzen
Die datei `/etc/exports`
Beispiel:

```
/home    (rw)
/                    (ro,root_squash)
/cdrom   (ro)
/root    *.domaine.com(rw,no_root_squash,map_daemon)
```

`rcnfsserver restart`
oder `/sbin/init.d/nfsserver restart`

NFS Client
Konfigurationsdatei: START_PORTMAP (Remote Procedure Call Portmapper) auf »yes« setzen

```
/sbin/init.d/rpc stop
/sbin/init.d/rpc start
```

`Mount -t nfs` *`RECHNER`*`:`*`/Remote-Pfad`* *`/Lokaler-Pfad`*

NFS (Network File System) Remote Mounting

- **Mounting and NFS:**
  - `/etc/fstab` Default mounted devices
  - `/etc/mtab` Already mounted devices. Changes are dynamic

- **User Space NFS Server setup**
  - check for the presence of daemons: `ps -axOt | less`
    1. `portmap`
    2. `rpc.mountd`
    3. `rpc.nfsd` or multi `[nfsd]` for kernel based NFS
    4. or do a `rpcinfo -p` for list of runnning `rpc` programs
  - If not then **yast** and **network services** (item 3) RPC and NFS enable
  - Set NFS_SERVER and NFS_SERVER_UGID to yes in `/etc/rc.config`
  - `chmod 7??` to shared directories
  - Edit `/etc/exports` file (see also `man exports` document below)

```
eg.   /*.berlin.de     (rw,no_root_squash,map_daemon)
      /cdrom           0.0.0.0/0.0.0.0(ro)
      /home            192.168.0.0/255.255.0.0(rw)
      /public          *(ro)
```

  - Restart NFS system with `rcnfsserver restart`

- **Kernet Space NFS Server Setup:**

  - check for the presence of daemons: `ps -axOt | less`
    1. `portmap`
    2. `rpc.mountd`
    3. multiple `[nfsd]`
    4. or do a `rpcinfo -p` for list of runnning **rpc** programs
    5. For file locking support:
       1. `rpc.statd`
          Files used: `/var/lib/nfs/sm/*`
       2. `rpc.lockd`
  - Set the NFS Boot autostart In Yast-1 (`/etc/rc.config`)

```
START_PORTMAP = yes
USE_KERNEL_NFSD = yes (till SuSE 71 only)
NFS_SERVER= yes
```

  - Edit `/etc/exports` file (see also `man exports` document below)

```
eg.   /            *.berlin.de(rw,no_root_squash)
      /cdrom       *(ro) 192.168.10.100(rw)
      /home        192.168.0.0/255.255.0.0(rw)
      /public      *(rw,sync)
      /transfer    192.168.0.0/24(ro,intr)
```

Note: NFS server when started, uses `exportfs` command to read this file.
`man exports` - Shows the format and options of the `/etc/exports`

Important: NO space between IP addr/Netmask and the access rights.
If there is a space, then the IP addr/Netmask hosts will be denied access and the access rights will be assigned to all other hosts.
eg. `/public achow(rw) diamond (ro)`

Means: achow will have read/write access, diamond hosts denied access and all others are allowed access with read permissions.
- chmod 7?? *Dirname* (to shared directories as desired)
- Always reload the NFS server after any change to /etc/exports with:

```
rcnfsserver restart
```

NFS commands:

cat /var/lib/nfs/xtab	- List of presently mounted clients on an NFS server.
rm /var/lib/nfs/rmtab*	- Erase old nfs connections to avoid waiting on reboot
showmount -e *servername*	- List the exports of any NFS Server
showmount -e localhost	- List exports of local NFS Server
exportfs	- List exports of local NFS Server
exportfs -v	- List exports of local NFS Server (incl options)

exportfs -o *export_options clients*:*exported_dir*

Adds a new export resource to the list taken from /etc/exports into the kernel without having to change the /etc/exports
exportfs is dynamic: DO NOT restart/reload the NFS server.
The default exportfs options are async,ro,root_squash,no_delay
examples:
Dynamically adding exports:

```
exportfs -o ro,umask=000,user 192.168.10.0/24:/data
exportfs -o rw,no_root_squash :/transfer
```

Dynamically removing exports:

```
exportfs -u rw,no_root_squash :/transfer
```

Shutting down NFS server:

```
exportfs -ua
```

(un-export **all** exported resources: NFS stops running)

Updating of NFS server exports according to /etc/exports:
(re-reading of /etc/exports) It is done automatically at start of NFS server.

```
exportfs -a
```

- **Files in /var/lib/nfs directory:**

rmtab — List of current and previously mounted (and unmounted!!!) clients. This is the list shown by the command showmount -a
Recommended to empty it(NOT reasing) before restarting NFS to avoid reconnecting old clients nfs mountings.

etab — Content of /etc/exports translated into nfs readable configuration. (same format but more complete)

xtab — List of presently mounted clients with mounted directories and their options.(Not really actual) Can be erased.
For actual list of mounted clients you need to do the following sequence:

```
rm /var/lib/nfs/xtab
rcnfsserver reload
cat /var/lib/nfs/xtab
```

- **NFS Settings for Clients:**

- Check for the presence of daemons:
  1. - portmap
  2. - For file locking support:
     1. rpc.statd
        Files used: /var/lib/nfs/sm/*

2. `rpc.lockd`

   `START_PORTMAP=yes` and `rcportmap start`

Optional: Edit `/etc/fstab` file and enter the device as *IP-No:/path/* and the rest like a normal mounting entry.

- Manual Mountig Command:

  `mount [-t nfs]` *remoteserver*`:`*/remotedir /mount/point*

EXPORTS

NFS Server configuration file

NAME

`exports` - NFS file systems being exported

LOCATION

`/etc/exports`

DESCRIPTION

The file /etc/exports serves as the access control list for file systems which may be exported to NFS clients. It is used by both the NFS mount daemon, mountd(8) and the NFS file server daemon nfsd(8).

The file format is similar to the SunOS exports file, except that several additional options are permitted.

Each line contains a mount point and a list of machine or netgroup names allowed to mount the file system at that point. An optional parenthesized list of mount parameters may follow each machine name. Blank lines are ignored, and a # introduces a comment to the end of the line.

Entries may be continued across newlines using a backslash.

Machine Name Formats

NFS clients may be specified in a number of ways:

single host

This is the most common format. You may specify a host either by an abbreviated name recognizued be the resolver, the fully qualified domain name, or an IP address.

netgroups

NIS netgroups may be given as @group. Only the host part of all netgroup members is extracted and added to the access list. Empty host parts or those containing a single dash (-) are ignored.

wildcards

Machine names may contain the wildcard characters * and ?. This can be used to make the exports file more compact; for instance, *.cs.foo.edu matches all hosts in the domain cs.foo.edu. However, these wildcard characters do not match the dots in a domain name, so the above pattern does not include hosts such as a.b.cs.foo.edu.

IP networks

You can also export directories to all hosts on an IP (sub-) network simultaneously. This is done by specifying an IP address and netmask pair as address/netmask.

=public

This is a special ``hostname'' that identifies the given directory name as the public root directory (see the section on WebNFS in nfsd(8) for a discussion of WebNFS and the public root handle). When using this convention, =public must be the only entry on this line, and must have no export options associated with it. Note that this does not actually export the named directory; you still have to set the exports options in a separate entry.

The public root path can also be specified by invoking nfsd with the --public-root option. Multiple specifica tions of a public root will be ignored.

General Options

mountd and nfsd understand the following export options:

`secure` (`default=secure`)

This option requires that requests originate on an internet port less than IPPORT_RESERVED (1024). This option is on by default. To turn it off, spec ify insecure.

`rw` (`default=ro`)

Allow the client to modify files and directories.The default is to restrict the client to read-only request, which can be made explicit by using the ro option.

`noaccess` (`default = access is granted`)

This makes everything below the directory inacces sible for the named client. This is useful when you want to export a directory hierarchy to a client, but exclude certain subdirectories. The client's view of a directory flagged with noaccess is very limited; it is allowed to read its

attributes, and lookup `.' and `..'. These are also the only entries returned by a readdir.

`link_relative` (`default = NOT link_relative`)

Convert absolute symbolic links (where the link contents start with a slash) into relative links by prepending the necessary number of ../'s to get from the directory containing the link to the root on the server. This has subtle, perhaps questionable, semantics when the file hierarchy is not mounted at its root.

`link_absolute` (`default = link_absolute`)

Leave all symbolic link as they are. This is the default operation.

Anonymous Entries

Entries where hosts are not specified are known as anonymous entries. They have different default settings compared to normal entries. The differences include all_squash, no_secure, and ro.

User ID Mapping

nfsd bases its access control to files on the server machine on the uid and gid provided in each NFS RPC request. The normal behavior a user would expect is that she can access her files on the server just as she would on a normal file system. This requires that the same uids and gids are used on the client and the server machine.

This is not always true, nor is it always desirable.

Very often, it is not desirable that the root user on a client machine is also treated as root when accessing files on the NFS server. To this end, uid 0 is normally mapped to a different id: the so-called `anonymous` or `nobody` uid. This mode of operation (called `root squash ing') is the default, and can be turned off with `no_root_squash`.

By default, nfsd tries to obtain the `anonymous` uid and gid by looking up user `nobody` in the password file at startup time. If it isn't found, a uid and gid of -2 (i.e. 65534) is used. These values can also be overridden by the anonuid and anongid options.

In addition to this, nfsd lets you specify arbitrary uids and gids that should be mapped to user `nobody` as well.

Finally, you can map all user requests to the `anonymous` uid by specifying the `all_squash` option.

For the benefit of installations where uids differ between different machines, nfsd provides several mechanism to dynamically map server uids to client uids and vice versa: static mapping files, NIS-based mapping, and ugidd-based mapping.

ugidd-based mapping is enabled with the map_daemon option, and uses the UGID RPC protocol. For this to work, you have to run the `ugidd(8)` mapping daemon on the client host. It is the least secure of the three methods, because by running `ugidd`, everybody can query the client host for a list of valid user names. You can protect yourself by restricting access to `ugidd` to valid hosts only. This can be done by entering the list of valid hosts into the `hosts.allow` or `hosts.deny` file. The service name is ugidd. For a description of the file's syntax, please read hosts_access(5).

Static mapping is enabled by using the `map_static` option, which takes a file name as an argument that describes the mapping. NIS-based mapping queries the client's NIS server to obtain a mapping from user and group names on the server host to user and group names on the client.

Here's the complete list of **mapping options**:

`root_squash`
Map requests from uid/gid 0 to the anonymous uid/gid. Note that this does not apply to any other uids that might be equally sensitive, such as user bin.

`no_root_squash`
Turn off root squashing. This option is mainly use ful for diskless clients.

`squash_uids` and
`squash_gids`
This option specifies a list of uids or gids that should be subject to anonymous mapping. A valid list of ids looks like this:

```
squash_uids=0-15,20,25-50
```

Usually, your squash lists will look a lot simpler.

`all_squash`
Map all uids and gids to the anonymous user. Useful for NFS-exported public FTP directories, news spool directories, etc. The opposite option is no_all_squash, which is the default setting.

`map_daemon`

This option turns on dynamic uid/gid mapping. Each uid in an NFS request will be translated to the equivalent server uid, and each uid in an NFS reply will be mapped the other way round. This option requires that rpc.ugidd(8) runs on the client host.

The default setting is map_identity, which leaves all uids untouched. The normal squash options apply regardless of whether dynamic mapping is requested or not.

`map_static`

This option enables static mapping. It specifies the name of the file that describes the uid/gid mapping, e.g.

```
map_static=/etc/nfs/foobar.map
```

The file's format looks like this

```
# Mapping for client foobar:
#    remote     local
uid  0-99       -       # squash these
uid  100-500    1000    # map 100-500 to 1000-1400
gid  0-49       -       # squash these
gid  50-100     700     # map 50-100 to 700-750
```

`map_nis`
This option enables NIS-based uid/gid mapping. For instance, when the server encounters the uid 123 on the server, it will obtain the login name associated with it, and contact the NFS client's NIS server to obtain the uid the client associates with the name.

In order to do this, the NFS server must know the client's NIS domain. This is specified as an argu ment to the map_nis options, e.g.

```
map_nis=foo.com
```

Note that it may not be sufficient to simply spec ify the NIS domain here; you may have to take addi tional actions before nfsd is actually able to contact the server. If your distribution uses the NYS library, you can specify one or more NIS servers for the client's domain in /etc/yp.conf. If you are using a different NIS library, you may have to obtain a special ypbind(8) daemon that can be con figured via yp.conf.

`anonuid` and
`anongid`
These options explicitly set the uid and gid of the anonymous account. This option is primarily useful for PC/NFS clients, where you might want all requests appear to be from one user. As an example, consider the export entry for `/home/joe` in the example section below, which maps all requests to uid 150 (which is supposedly that of user joe).

EXAMPLE

```
# sample /etc/exports file
/               master(rw) trusty(rw,no_root_squash)
/projects       proj*.local.domain(rw)
/usr            *.local.domain(ro) @trusted(rw)
/home/joe       pc001(rw,all_squash,anonuid=150,anongid=100)
/pub            (ro,insecure,all_squash)
/pub/private    (noaccess)
```

The first line exports the entire filesystem to machines master and trusty. In addition to write access, all uid squashing is turned off for host trusty. The second and third entry show examples for wildcard hostnames and net groups (this is the entry `` `@trusted' ``). The fourth line shows the entry for the PC/NFS client discussed above.

Line 5 exports the public FTP directory to every host in the world, executing all requests under the nobody account. The insecure option in this entry also allows clients with NFS implementations that don't use a reserved port for NFS. The last line denies all NFS clients access to the private directory.

CAVEATS
Unlike other NFS server implementations, this nfsd allows you to export both a directory and a subdirectory thereof to the same host, for instance `/usr` and `/usr/X11R6`. In this case, the mount options of the most specific entry apply. For instance, when a user on the client host accesses a file in `/usr/X11R6`, the mount options given in the `/usr/X11R6` entry apply. This is also true when the latter is a wildcard or netgroup entry.

FILES
`/etc/exports`

DIAGNOSTICS
An error parsing the file is reported using `syslogd(8)` as level NOTICE from a DAEMON whenever `nfsd(8)` or `mountd(8)` is started up. Any unknown host is reported at that time, but often not all hosts are not yet known to `named(8)` at boot time, thus as hosts are found they are reported with the same `syslogd(8)` parameters.

SEE ALSO

`mountd(8), nfsd(8)`

MAN NFS

NAME :

nfs - nfs fstab format and options

SYNOPSIS

`/etc/fstab`

DESCRIPTION

The fstab file contains information about which filesys tems to mount where and with what options. For NFS mounts, it contains the server name and exported server directory to mount from, the local directory that is the mount point, and the NFS specific options that control the way the filesystem is mounted.

Here is an example from an `/etc/fstab` file from an NFS mount.

```
server:/usr/local/pub   /pub   nfs  rsize=8192,wsize=8192,timeo=14,intr
```

Options

rsize=n The number of bytes NFS uses when reading files from an NFS server. The default value is dependent on the kernel, currently 1024 bytes. (However, throughput is improved greatly by asking for rsize=8192.)

wsize=n The number of bytes NFS uses when writing files to an NFS server. The default value is dependent on the kernel, currently 1024 bytes. (However, throughput is improved greatly by asking for wsize=8192.)

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 retransmissions 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 gateways.

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 console.

acregmin=n

The minimum time in seconds that attributes of a regular file should be cached before requesting fresh information from a server.
The default is 3 seconds.

acregmax=n

The maximum time in seconds that attributes of a regular file can be cached before requesting fresh information from a server. The default is 60 seconds.

acdirmin=n
The minimum time in seconds that attributes of a directory should be cached before requesting fresh information from a server. The default is 30 seconds.

acdirmax=n
The maximum time in seconds that attributes of a directory can be cached before requesting fresh information from a server. The default is 60 seconds.

actimeo=n
Using actimeo sets all of acregmin, acregmax, acdirmin, and acdirmax to the same value. There is no default value.

retry=n The number of minutes to retry an NFS mount operation in the foreground or background before giving up. The default value is 10000 minutes, which is roughly one week.

namlen=n
When an NFS server does not support version two of the RPC mount protocol, this option can be used to specify the maximum length of a filename that is supported on the remote filesystem. This is used to support the POSIX pathconf functions. The default is 255 characters.

port=n The numeric value of the port to connect to the NFS server on. If the port number is 0 (the default) then query the remote host's portmapper for the port number to use. If the remote host's NFS daemon is not registered with its portmapper, the standard NFS port number 2049 is used instead.

mountport=n
The numeric value of the mountd port.

mounthost=name
The name of the host running mountd .

mountprog=n
Use an alternate RPC program number to contact the mount daemon on the remote host. This option is useful for hosts that can run multiple NFS servers. The default value is 100005 which is the standard RPC mount daemon program number.

mountvers=n
Use an alternate RPC version number to contact the mount daemon on the remote host. This option is useful for hosts that can run multiple NFS servers. The default value is version 1.

nfsprog=n
Use an alternate RPC program number to contact the NFS daemon on the remote host. This option is useful for hosts that can run multiple NFS servers. The default value is 100003 which is the standard RPC NFS daemon program number.

nfsvers=n
Use an alternate RPC version number to contact the NFS daemon on the remote host. This option is useful for hosts that can run multiple NFS servers. The default value is version 2.

nolock Disable NFS locking. This has to be used with some old NFS servers that don't support locking.

bg If the first NFS mount attempt times out, retry the mount in the back ground. After a mount operation is backgrounded, all subsequent mounts on the same NFS server will be backgrounded immediately, without first attempting the mount. A missing mount point is treated as a timeout, to allow for nested NFS mounts.

fg If the first NFS mount attempt times out, retry the mount in the foreground. This is the complement of the bg option, and also the default behavior.

soft If an NFS file operation has a major timeout then report an I/O error to the calling program. The default is to continue retrying NFS file operations indefinitely.

hard If an NFS file operation has a major timeout then report "server not responding" on the console and continue retrying indefinitely. This is the default.

intr If an NFS file operation has a major timeout and it is hard mounted, then allow signals to interupt the file operation and cause it to return EINTR to the calling program. The default is to not allow file operations to be interrupted.

posix Mount the NFS filesystem using POSIX semantics. This allows an NFS filesystem to properly support the POSIX pathconf command by querying the mount server for the maximum length of a filename. To do this, the remote host must support version two of the RPC mount protocol. Many NFS servers support only version one.

nocto Suppress the retrieval of new attributes when creating a file.

noac Disable all forms of attribute caching entirely. This extracts a server performance penalty but it allows two different NFS clients to get reason able good results when both clients are actively writing to common filesystem on the server.

nolock Do not use locking. Do not start lockd.

tcp Mount the NFS filesystem using the TCP protocol instead of the default UDP protocol. Many NFS servers only support UDP.

udp Mount the NFS filesystem using the UDP protocol. This is the default.

All of the non-value options have corresponding nooption forms. For example, nointr means don't allow file operations to be interrupted.

FILES

`/etc/fstab`

SEE ALSO

`fstab(5), mount(8), umount(8), exports(5)`

BUGS

The posix, and nocto options are parsed by mount but currently are silently ignored. The tcp and namlen options are implemented but are not currently supported by the Linux kernel. The umount command should notify the server when an NFS filesystem is unmounted.