

NFS

Dazu vorab einige Bemerkungen.

NFS hat nicht direkt etwas mit Fernsehen und digitalem Satelliten-Empfang zu tun. Es ist auch keineswegs eine Errungenschaft der Dreambox oder davor schon der Dbox und es ist schon wirklich alt und deshalb in vielen verschiedenen Versionen möglich.

Was ich nun beschreibe, gibt einfachste Grundlagen wieder und hat nicht den Anspruch, vollständig zu sein. Vielmehr konzentriere ich mich auf jene, die durch den Kontakt mit Dreambox und Dbox erstmalig mit einem Unix System konfrontiert werden und Schwierigkeiten bei der Umsetzung der Begriffe und Anleitungen haben, weil hier jegliche Vorkenntnis fehlt. Den eigentlichen Dienst NFS und Hintergründe dazu behandle ich nicht.

ALLGEMEINES:

Eine typische Antwort für Anfänger aus einer Unix-FAQ will ich aber zitieren:

„Warum meldet mir mountd auf meinem FreeBSD NFS-Server ständig can't change attributes und bad exports list?“

„Die häufigste Ursache für dieses Problem ist, dass Sie den Aufbau der exports(5) nicht oder nicht richtig verstanden haben. Überprüfen Sie Ihre exports(5) und lesen das Kapitel NFS“

Wer dazu nun den Kopf schüttelt, sollte wissen: es ist das Niveau, auf das man sich begeben muss, wenn man mit Unix arbeiten möchte (was sehr empfehlenswert ist und bleibt). Denn mit Ausdrücken wie „exports(5)“ wird auf entsprechende Hilfetexte verwiesen, die in einem Unix System zum Standard gehören (oder auf Wunsch installiert werden können, so genannte man-pages). Für dieses Beispiel genügt in einem Unix System der einfache Aufruf **man exports** um eine solche Information zu erhalten:

```
EXPORTS(5)                                FreeBSD File Formats Manual                EXPORTS(5)

NAME
  exports -- define remote mount points for NFS mount requests

SYNOPSIS
  exports

DESCRIPTION
  The exports file specifies remote mount points for the NFS mount protocol
  per the NFS server specification; see Network File System Protocol
  Specification, RFC1094, Appendix A and NFS: Network File System Version 3
  Specification, Appendix I.

  Each line in the file (other than comment lines that begin with a #)
  specifies the mount point(s) and export flags within one local server
  file system for one or more hosts.  A long line may be split over several
  lines by ending all but the last line with a backslash (`\`).  A host may
  be specified only once for each local file system on the server and there
  may be only one default entry for each server file system that applies to
  all other hosts.  The latter exports the file system to the ``world'' and
  should be used only when the file system contains public information.

  In a mount entry, the first field(s) specify the directory path(s) within
  a server file system that can be mounted on by the corresponding
  client(s).  There are two forms of this specification.  The first is to
  list all mount points as absolute directory paths separated by white-
  space.  The second is to specify the pathname of the root of the file
  system followed by the -alldirs flag; this form allows the host(s) to
  mount at any point within the file system, including regular files if the
  -r option is used on mountd(8).  The pathnames must not have any symbolic
  links in them and should not have any "." or ".." components.  Mount
  points for a file system may appear on multiple lines each with different
  sets of hosts and export options.
```

The second component of a line specifies how the file system is to be exported to the host set. The option flags specify whether the file system is exported read-only or read-write and how the client uid is mapped to user credentials on the server.

Export options are specified as follows:

`-maproot=user` The credential of the specified user is used for remote access by root. The credential includes all the groups to which the user is a member on the local machine (see `id(1)`). The user may be specified by name or number.

`-maproot=user:group1:group2:...` The colon separated list is used to specify the precise credential to be used for remote access by root. The elements of the list may be either names or numbers. Note that `user:` should be used to distinguish a credential containing no groups from a complete credential for that user.

`-mapall=user` or `-mapall=user:group1:group2:...` specifies a mapping for all client uids (including root) using the same semantics as `-maproot`.

The option `-r` is a synonym for `-maproot` in an effort to be backward compatible with older export file formats.

In the absence of `-maproot` and `-mapall` options, remote accesses by root will result in using a credential of `-2:-2`. All other users will be mapped to their remote credential. If a `-maproot` option is given, remote access by root will be mapped to that credential instead of `-2:-2`. If a `-mapall` option is given, all users (including root) will be mapped to that credential in place of their own.

The `-ro` option specifies that the file system should be exported read-only (default read/write). The option `-o` is a synonym for `-ro` in an effort to be backward compatible with older export file formats.

WebNFS exports strictly according to the spec (RFC 2054 and RFC 2055) can be done with the `-public` flag. However, this flag in itself allows r/w access to all files in the file system, not requiring reserved ports and not remapping uids. It is only provided to conform to the spec, and should normally not be used. For a WebNFS export, use the `-webnfs` flag, which implies `-public`, `-mapall=nobody` and `-ro`. Note that only one file system can be WebNFS exported on a server.

A `-index=file` option can be used to specify a file whose handle will be returned if a directory is looked up using the public filehandle (WebNFS). This is to mimic the behavior of URLs. If no `-index` option is specified, a directory filehandle will be returned as usual. The `-index` option only makes sense in combination with the `-public` or `-webnfs` flags.

Specifying the `-quiet` option will inhibit some of the syslog diagnostics for bad lines in `/etc/exports`. This can be useful to avoid annoying error messages for known possible problems (see EXAMPLES below).

The third component of a line specifies the host set to which the line applies. The set may be specified in three ways. The first way is to list the host name(s) separated by white space. (Standard Internet `'dot'` addresses may be used in place of names.) The second way is to specify a `'netgroup'` as defined in the netgroup file (see `netgroup(5)`). The third way is to specify an Internet subnetwork using a network and network mask that is defined as the set of all hosts with addresses within the subnetwork. This latter approach requires less overhead within the kernel and is recommended for cases where the export line refers to a large number of clients within an administrative subnet.

The first two cases are specified by simply listing the name(s) separated by whitespace. All names are checked to see if they are `'netgroup'` names first and are assumed to be hostnames otherwise. Using the full domain specification for a hostname can normally circumvent the problem of a host that has the same name as a netgroup. The third case is specified by the flag `-network=netname` and optionally `-mask=netmask`. If the mask is not specified, it will default to the mask for that network class (A, B or C; see `inet(4)`). See the EXAMPLES section below.

The `mountd(8)` utility can be made to re-read the exports file by sending it a hangup signal as follows:

```
kill -s HUP `cat /var/run/mountd.pid`
```

After sending the `SIGHUP`, check the `syslogd(8)` output to see whether

mountd(8) logged any parsing errors in the exports file.

FILES

/etc/exports the default remote mount-point file

EXAMPLES

```
/usr /usr/local -maproot=0:10 friends
/usr -maproot=daemon grumpy.cis.uoguelph.ca 131.104.48.16
/usr -ro -mapall=nobody
/u -maproot=bin: -network 131.104.48 -mask 255.255.255.0
/u2 -maproot=root friends
/u2 -alldirs -network cis-net -mask cis-mask
/cdrom -alldirs,quiet,ro -network 192.168.33.0 -mask 255.255.255.0
```

Given that /usr, /u and /u2 are local file system mount points, the above example specifies the following:

/usr is exported to hosts friends where friends is specified in the netgroup file with users mapped to their remote credentials and root mapped to uid 0 and group 10. It is exported read-write and the hosts in ``friends'' can mount either /usr or /usr/local. It is exported to 131.104.48.16 and grumpy.cis.uoguelph.ca with users mapped to their remote credentials and root mapped to the user and groups associated with ``daemon''; it is exported to the rest of the world as read-only with all users mapped to the user and groups associated with ``nobody''.

/u is exported to all hosts on the subnetwork 131.104.48 with root mapped to the uid for ``bin'' and with no group access.

/u2 is exported to the hosts in ``friends'' with root mapped to uid and groups associated with ``root''; it is exported to all hosts on network ``cis-net'' allowing mounts at any directory within /u2.

The file system rooted at /cdrom will be exported read-only to the entire network 192.168.33.0/24, including all its subdirectories. Since /cdrom is the conventional mountpoint for a CD-ROM device, this export will fail if no CD-ROM medium is currently mounted there since that line would then attempt to export a subdirectory of the root file system with the -alldirs option which is not allowed. The -quiet option will then suppress the error message for this condition that would normally be syslogged. As soon as an actual CD-ROM is going to be mounted, mount(8) will notify mountd(8) about this situation, and the /cdrom file system will be exported as intended. Note that without using the -alldirs option, the export would always succeed. While there is no CD-ROM medium mounted under /cdrom, it would export the (normally empty) directory /cdrom of the root file system instead.

SEE ALSO

netgroup(5), mountd(8), nfsd(8), showmount(8)

BUGS

The export options are tied to the local mount points in the kernel and must be non-contradictory for any exported subdirectory of the local server mount point. It is recommended that all exported directories within the same server file system be specified on adjacent lines going down the tree. You cannot specify a hostname that is also the name of a netgroup. Specifying the full domain specification for a hostname can normally circumvent the problem.

Ich habe das nicht alles gelesen und verstanden! Ich habe es nicht einmal gelesen, sondern etliche male und außerdem noch mehr dazu, wie zum Beispiel die von mir eingefärbten Einträge, zu denen es jeweils eigene Hilfetexte gibt. Aber ich würde lügen, wenn ich behauptete, alles verstanden zu haben und deshalb ein Experte zu sein. Hier wollte ich nur ein Beispiel zeigen, was in einem Unix System als selbstverständlich gilt, dass nämlich Programme und Aufrufe dokumentiert sind, so dass jeder sie sich erklären kann und sie nach eigener Vorstellung einsetzt und nutzt, bis zu dem Punkt, wo in OpenSource Systemen sogar die Quellen der Programme selbst offen liegen und verändert werden können.

Leider gilt das nicht für Dreambox und Dbox, die haben keine man und info Seiten.

So ist also jedenfalls im Vorteil, wer auch ein Unix auf seinem PC einsetzt, weil er dort wenigstens die grundlegenden Funktionen verstehen kann, da sie dort gut dokumentiert sind. Deshalb nun eine Erklärung, die mir wichtig erscheint und die doch gar nichts mit dem eigentlichen Thema zu tun hat.

Unix GNU Linux und busybox

Ken Thompson und Dennis Ritchie entwickelten als Angestellte bei Bell Laboratories gemeinsam mit einigen anderen ein neues Betriebssystem, das schließlich Unix genannt wurde. Dies fand in den 60ern statt und Entwicklungen liefen ständig weiter, niemand dachte aber zu dieser Zeit daran, dass Software etwas ist, womit Geld verdient werden kann. Es herrschte allgemein eine Aufbruchstimmung und die meisten Beteiligten bezeichnen sich heute noch als Freaks und wollten mit guten Programmen überzeugen, nicht marktwirtschaftlich glänzen. Ken Thompson nahm deshalb einfach sein neues Betriebssystem mit nach Berkeley, wo er an der Universität als Gast-Professor eine Zeit lang tätig war. Dort wurde das neue Unix fleißig weiter entwickelt und weiter verteilt, bis die inzwischen zu AT&T gehörenden Bell Laboratories ihren Anspruch auf diese Entwicklung geltend machten und eine weitere, freie Verbreitung dieses Systems untersagten. Während der diversen gerichtlichen Auseinandersetzungen darum, gab es also kein freies Unix mehr und das nahm Richard Stallman zum Anlass, GNU zu schaffen und dafür zu sorgen, dass dieses von Anfang an und für alle Zeiten durch eine Lizenz geschützt wird (die General Public License), so dass es immer offen und frei verfügbar bleibt. GNU bedeutet Gnu is Not Unix und meint damit, dass es wirklich nicht Unix ist, weil nämlich alle Programme und Tools vollkommen neu geschrieben wurden, dass es aber alles genauso macht, als wäre es ein Unix.

Ein Merkmal von Unix ist seit Anbeginn, dass es auf den verschiedensten Rechnern laufen kann. Es hat nämlich viele Hardware-nahe Steuerungsaufgaben in den so genannten Betriebssystem-Kernel verlagert. Der Kernel kümmert sich also, grob gesagt, um die Hardware und stellt eine Schnittstelle für das Betriebssystem dar, die für alle Hardware immer gleich ist (im Idealfall). Das gleiche Betriebssystem kann also, mit angepassten Kernen, auf Intel und AMD, auf Sparc und Mips und Alpha und vielen anderen Rechnern laufen, kann 8, 16, 32 oder 64 Bit Prozessoren dienen und so sehr weit angepasst und verbreitet werden. Das ist ein Riesenvorteil für alle Unix-Systeme und vielleicht auch heute noch eines der Hauptargumente für Entwickler von Programmen. Sie brauchen sich nicht um die Hardware zu kümmern. Egal, welcher Prozessor und welche Hardware, im Zweifel wird ihr Programm einfach neu kompiliert und schon kann es auf allen Plattformen eingesetzt werden. Dazu müssen die Kompiler-Programme natürlich entweder beim Endverbraucher vorhanden sein, oder jemand macht sich die Mühe und stellt fertig kompilierte Lösungen für die einzelnen Plattformen bereit.

Damit können wir zurück zu Richard Stallman. Der hatte etwa Mitte der 80er sein GNU soweit fertig und wollte sich nun HURD widmen, jenem Teil seines Projektes, der einen Kernel entwickeln sollte. Es kann gut sein, dass er zunächst GNU entwickelte, weil er darauf spekulierte, wenigstens den Kernel aus dem Unix von AT&T oder der Universität Berkeley verwenden zu können. Doch die geführten Prozesse um BSD (Berkeley Software Distribution) waren nicht einfach und in deren Verlauf war die Verwendung deren Kernel auch blockiert. Jedenfalls hatte Richard Stallman sich gleich zu Anfang der Aufgabe gestellt, einen neuen Kompiler zu schreiben. Ein Programm, das also den Quellcode aller anderen GNU-Programme auf die geforderte Struktur übersetzen konnte. Wie alles, was er und seine Mitstreiter entwickelten, stellten sie auch diesen Kompiler unter GPL, die Lizenz, die garantieren soll, dass Software, die frei für alle entwickelt wurde, auch immer frei für alle bleiben konnte. Und genau diesen Kompiler, der durch seine Lizenz geschützt und frei zugänglich, veränderbar und nutzbar war, nahm ein Student, um damit einen eigenen Kernel zu entwickeln, den er zunächst Freaks nannte und der später als Linux bekannt werden sollte. Dass dieser Kernel gerade nun zur Verfügung stand, wo das GNU-Projekt nach einem solchen suchte, war natürlich sehr verlockend und unglaublich schnell hatte sich GNU/Linux etabliert und wird seither mit rasender Geschwindigkeit weiter entwickelt, vermutlich, weil die jeweiligen Programmierer die Idee freier Software für gut finden und unter der GPL am ehesten Schutz erhoffen. Linux, der Kernel, war dabei zunächst gar nicht durch die GPL geschützt. Es bedurfte einer gewissen Überzeugungsarbeit, doch weil Linus Torvalds schon den Kompiler von Richard Stallman nutzte und der bereits unter GPL veröffentlicht war, wäre jede andere Entscheidung doch merkwürdig gewesen.

Linus Torvalds wurde seither zu einer Frontfigur für die gesamte Bewegung, die sich der freien Verfügbarkeit von Software verschrieben hat und Linux inzwischen zu einem Synonym für komplette Betriebssysteme und Anwendungsprogramme, die von zahlreichen, so genannten Distributoren

zusammen gestellt und geschaffen werden und die alle unter der GPL von Richard Stallman veröffentlicht werden.

Dem gegenüber steht BSD. Dieses entwickelte sich aus dem ursprünglichen Unix, das Ken Thompson einst an die Universität von Berkeley mitgenommen hatte. Es wurde schließlich von allem Code befreit, der noch irgendwie von AT&T stammen könnte und erhielt damit endgültig seine Freiheit und auch auch den Namen Free-BSD. Inzwischen wurde davon abgeleitet Open-BSD, Net-BSD und Dragon-Fly-BSD und einige andere freie Systeme, so wie bekannte Systeme, die entweder von freien, oder von AT&T Systemen abgeleiteten kommerziellen und meist unfreien Systeme, wie Mac OS X, HP-UX, AIX, IRIX und Solaris und viele mehr.

Nicht nur die unterschiedliche Lizenz macht dabei beide große Linien freier Software interessant, auch Unterschiede in Performance und Stabilität bestimmen den Wettstreit, wobei ein normaler User (=Nutzer) vermutlich keine Unterschiede realisieren wird da beide als gleich ausgereift und sicher gelten dürfen und obschon deutlich unterschiedlich, letztlich doch hauptsächlich unterschiedlich zu anderen Systemen erkannt werden, die sich nicht in die Unix-Reihe einfügen wollen und eigene Standards nutzen.

busybox wäre beinahe so etwas.

Nun, da wir die grundlegende Aufteilung von Unix-Systemen in Kernel-Komponenten und Programme kennen, verschafft sich ein neues Modell Aufmerksamkeit. busybox wurde speziell für den Einsatz in Systemen geschaffen, wo nur begrenzter Speicher zur Verfügung steht. Es nutzt die Unix Struktur, das heißt, es braucht einen Kernel, und, wo immer ich es sehe, hat es Linux als solchen gefunden. Anders, als bei GNU und anderen Unix-Systemen, werden aber die möglichen Tools und Programme meist nicht einzeln kompiliert. Ein Befehl, wie etwa date ist sowohl in BSD als in GNU ein eigenständiges Programmpaket, das in diesem Falle die Systemzeit und das Datum darstellt oder ändert und das in einem Unix System zwar üblicherweise vorhanden ist, aber nicht zwingend eingebaut sein muss. busybox ist aber ein einziges, fertig kompiliertes Programmpaket. Ein Befehl, wie date, kann vorhanden sein, ist aber nur ein Kommando, das eine Funktion von busybox aufruft. Genauer hingeschaut, ist es als ein Link nach busybox realisiert, dazu später mehr:

```
root@dreambox:~> ls -al /bin/date
lrwxrwxrwx 1 root root 7 Oct 16 13:07 /bin/date -> busybox
```

busybox ist deshalb nicht so flexibel, einzelne Programmpakete können nicht einfach ausgetauscht werden. Die vorhandenen Befehle müssen beim Bau von busybox bereits in ihrem Umfang festgelegt werden und gerade auf den unterschiedlichen Satelliten-Boxen haben wir es mit einer abgespeckten Version zu tun, die nicht alle Möglichkeiten bietet.

Ein weiterer Grundgedanke von Unix war, alles, was dem User im Betriebssystem begegnet, soll eine Datei sein. Kernel und Betriebssystem müssen irgendwann entsprechende Informationen erhalten und stellen fortan alle Komponenten als Datei dar. Dies hat große Vorteile, verwirrt aber viele Anfänger. Wenn etwa die Festplatte in einem System gesucht wird, gibt es dafür keinen reservierten Eintrag der Art c:\. Wozu auch? Ein Benutzer soll sich gar nicht darum sorgen, ob seine Daten auf einer bestimmten Partition oder Festplatte oder einem Ort in einem Netzwerk oder wo auch immer liegen. Sobald er ausreichend Rechte hat, kann er auf Daten zugreifen und diese ändern und speichern, ohne sich über die physikalische Beschaffenheit seines Speichers sorgen zu müssen. Unix Systeme (und dazu gehören auch GNU und busybox) organisieren Daten in einem hierarchisch gebauten Dateibaum. Im Filesystem Hierarchy Standard ist beschrieben, wo sich welche Daten befinden sollen, aber kein Entwickler muss sich wirklich daran halten. Die Entwickler so genannter Images für die Dreambox oder Dbox bringen eigene Überlegungen dazu ein, wo sich welche Daten finden sollen. Dabei ist stets zu bedenken, dass in diesen Boxen nur ein Teil des Speichers sich einfach verändern lässt, also beschreibbar ist. Dies ist physikalisch gegeben und muss von einem Betriebssystem beachtet sein.

Im Folgenden führe ich Befehle auf der Box direkt aus. Dies geht, indem eine telnet Sitzung gestartet wird. Dazu wird in einem Kommando-Fenster (Bei Windows heißen die meist cmd oder command) **telnet IP.der.Box** aufgerufen. Es folgt ein login screen und hier ist root als User und dreambox bei DM7000, dbox2 bei Dbox2 und bei manchen Boxen auch gar kein Passwort einzugeben. Nebenbei, im folgenden spreche ich von der DM 7000 als Dreambox oder Box.

Der Dateibaum

Ein Blick auf den Dateibaum einer Dreambox, die ein Gemini Image nutzt:

```
BusyBox v1.2.1 (2006.10.11-21:54+0000) Built-in shell (ash)
Enter 'help' for a list of built-in commands.

root@dreambox:~> pwd
/
root@dreambox:~> ls
automount dev hdd mnt root share var
bin etc lib proc sbin tmp var_init
root@dreambox:~>
```

Im Filesystem Hierarchy Standard werden folgende Festlegungen für ein Unix System empfohlen:

```
/bin (grundlegende ausführbare Dateien/Befehle ? zur Verwendung durch alle Benutzer)
/boot (statische Dateien des Bootloaders)
/dev (Gerätedateien)
/etc (spezifische Konfigurationsdateien)
/home (optional, Benutzerverzeichnisse)
/lib (Kernel-Module und dynamische Bibliotheken)
/lib<Spezifikation> (optional, alternative(s) Verzeichnis(se) für grundlegende
dynamische Bibliotheken)
/media (Einhängepunkt für auswechselbare Datenträger)
/mnt (temporärer Einhängpunkt für Dateisysteme)
/opt (zusätzliche Softwarepakete)
/root (optional, "Home"-Verzeichnis für den Systemadministrator)
/sbin (wichtige Systembefehle)
/srv (Daten, die von Diensten angeboten werden)
/tmp (temporäre Dateien)
/usr (2. Verzeichnisebene)
/var (variable Daten)
```

Einige Übereinstimmung zur Dreambox lässt sich unschwer erkennen. Ein wenig mehr Aufschluss über die Organisation des Dateibaumes auf der Box gibt der Befehl `ls -al`, der alle Eigenschaften der Einträge aufzeigt und auch so genannte versteckte Dateien sichtbar macht (welche einen führenden Punkt im Namen haben).

```
root@dreambox:~> ls -al
lrwxrwxrwx 1 root root 9 Oct 16 13:07 .ssh -> /var/.ssh
-rw-r--r-- 1 root root 223 Oct 16 13:07 .version
drwxr-xr-x 1 root root 0 Oct 16 13:06 automount
drwxr-xr-x 1 root root 763 Oct 16 13:07 bin
drwxr-xr-x 1 root root 0 Jan 1 1970 dev
drwxr-xr-x 1 root root 318 Oct 16 13:07 etc
lrwxrwxrwx 1 root root 12 Oct 16 13:07 hdd -> /var/mnt/hdd
drwxr-xr-x 1 root root 809 Oct 16 13:07 lib
lrwxrwxrwx 1 root root 8 Oct 16 13:07 mnt -> /var/mnt
dr-xr-xr-x 59 root root 0 Jan 1 1970 proc
drwxr-xr-x 1 root root 0 Oct 16 13:06 root
drwxr-xr-x 1 root root 520 Oct 16 13:07 sbin
drwxr-xr-x 1 root root 124 Oct 16 13:07 share
drwxr-xr-x 2 root root 0 Jan 1 1970 tmp
drwxr-xr-x 21 root root 0 Jan 1 1970 var
drwxr-xr-x 1 root root 146 Oct 16 13:07 var_init
root@dreambox:~>
```

Es fällt auf, dass manche Einträge so genannte Links darstellen, ein Zeiger auf einen Eintrag an anderer Stelle. Der allererste Eintrag in der Liste erklärt dies: `d = directory`, also Verzeichnis, `l = link`, also ein Zeiger auf einen Eintrag an anderer Stelle. Wohin ein Link zeigt, wird ebenfalls angegeben: `hdd -> /var/mnt/hdd` etwa sagt, dass der Eintrag `hdd` auf einen Ort `/var/mnt/hdd` zeigt. Auf diese Struktur gehe ich gleich noch ein, sie muss uns beschäftigen und klar sein. Zunächst aber ein Hinweis auf die anderen Einträge in der obigen Ausgabe, so weit diese für uns wichtig sind.

Da stehen eine Reihe von `rw-rw-rw-r` in jeder Zeile und manchmal fehlt auch ein Symbol und wird durch ein `-` ersetzt. Dies sind die berechtigten Datei-Rechte in einem Unix System. Jede Datei kann drei Herren dienen. Diese werden User, Group und Others genannt, also Nutzer, Gruppe und Andere, in einfachster Übersetzung. Der User ist der Eigentümer dieser Datei und kann entscheiden, wer außerdem noch welche Rechte erhalten soll. Der erste Name in der Ausgabe in jeder Zeile ist dieser User, in unserem Beispiel `root` und auf der Dreambox gibt es meist nur diesen einen User, der mit Dateien Umgang hat. Der zweite Name gibt die Gruppe an, der diese Datei gehören soll. In unserem Falle etwas fantasielos ebenfalls `root`. Und die Rechte für die einzelnen Herren einer Datei

sind genau in dieser Reihenfolge vergeben und aufgeteilt, für jeden gibt es also ein rwx Päckchen, eines für User, eines für Group und eines für Others. Dabei steht r für read, also lesen, w für write, also schreiben und x für executable, also ausführbar. Wenn bei Others kein w eingetragen ist, dürfen diese Anderen die Datei nicht beschreiben, nur lesen und ausführen.

Der hier gezeigte Nutzer root ist in Unix Systemen ein besonderer Nutzer, der meist auch als Super User bezeichnet wird. Ihm stehen weitgehende Rechte zu und er kann solche Einträge auch verändern, sich selbst über Festlegungen anderer Nutzer hinwegsetzen. Deshalb arbeitet man normalerweise nicht als Nutzer root auf einem Unix System, um Risiken zu minimieren. Bei Dreambox und Dbox gibt es nur root und es bleibt keine Wahl, hier muss man als Super User und Systemadministrator tätig sein.

Aber, nur Mut, das Risiko auf einer Box das System zu zerstören ist relativ gering, denn auch root kann nicht in den so genannten Flash Speicher schreiben. Dieser Speicher ist eine Besonderheit. Daten können in einem bestimmten Vorgang, der flashen heißt, in diesen Bereich gebracht werden, doch sie können nicht einfach im laufenden Betrieb verändert werden. Deshalb gilt dieser Bereich für uns als ROM, nur lesbarer Speicher und folgende Ausgabe des Befehles df zeigt uns vielleicht mehr dazu, welcher physikalische Speicher auf der Box genutzt ist.

```
root@dreambox:~> df -h
Filesystem      Size      Used Available Use% Mounted on
/dev/root        4.2M      4.2M      0 100% /
/dev/mtdblock/1  1.8M      1.0M      724.0k  60% /var
/dev/ide/host0/bus0/target0/lun0/part1 93.1G      72.4G      20.8G  78% /var/mnt/hdd
root@dreambox:~>
```

Einem Eintrag aus /dev werden hier Einträge des Dateibaumes zugeordnet.

In /dev stehen Einträge, die vielleicht am einfachsten als Geräte-Treiber beschrieben werden. Für jedes Gerät muss ein solcher Eintrag existieren, der Hardware-Eigenschaften beschreibt. In Unix Systemen kann hier sehr viel stehen, weil niemand weiß, welche Hardware vielleicht einmal angesprochen werden soll und deshalb vorsorglich alle verfügbare Information hier gelistet ist. In einer begrenzten und bekannten Hardware Umgebung, wie einer Dreambox, kann dieser Bereich klein gehalten werden. /dev/root ist jener Speicher, der einfach nicht zu beschreiben ist und er enthält das komplette System, außer /var und /var/mnt/hdd, denen eigene Geräte zugewiesen wurden. /dev/mtdblock/1 ist ein beschreibbarer Speicher und /dev/ide/host0/bus0/target0/lun0/part1 ist der Eintrag, für eine Festplatte. Beim Booten, dem Systemstart, sorgen gezielte Einträge in einer Startkonfigurationsdatei dafür, dass diese Zuweisung automatisch erfolgt und immer wieder /var den beschreibbaren Speicher und var/mnt/hdd die Festplatte bekommt. Und damit ist auch sicher, dass der link, den wir oben bereits sahen, immer wieder die Festplatte trifft, da er von /hdd nach /var/mnt/hdd zeigt.

Es sollte nun vielleicht schon deutlich geworden sein, wie der Dateibaum zu lesen ist. Als Wurzel wird / angesehen, also ein Ursprungsverzeichnis, von dem alle anderen verzweigen. Die Schreibweise /hdd ist daher eindeutig, es ist der Eintrag hdd direkt unterhalb von / gemeint. Wie bereits gesehen, gibt es auch einen Eintrag /var/mnt/hdd und auch dieser ist eindeutig, hier wird hdd gemeint, das sich in einem Ordner mnt befindet, der unterhalb eines Ordners var liegt, welches wiederum direkt unterhalb der Wurzel liegt. Der Ausdruck hdd indessen ist mehrdeutig, im Zweifel weiß niemand, was damit gemeint ist. Innerhalb des Dateibaumes kann gewandert werden. Der Befehl cd wechselt den Standort des Users. ls listet die Einträge an einem Ort und ls -al zeigt zusätzlich die Eigenschaften dieser Einträge. Ich zeige Beispiele und jeder kann selbst einfach herausfinden, wie das funktioniert. Es wird allgemein von einem absoluten Pfad gesprochen, wenn der komplette Weg durch den Dateibaum angegeben ist, also stets beginnend mit einem /, wie etwa: /irgendwas/zu/irgendwo und der relative Pfad bezieht sich auf den jeweiligen Standort, zu/irgendwo geht also von dort, wo ich mich gerade befinde, erst nach zu, dann dort nach irgendwo und wenn dort, wo ich gerade bin kein zu existiert, wird ein Fehler gemeldet. In diesem Beispiel trifft zu/irgendwo also nur genau, wenn ich bereits in /irgendwas bin. Im Gemini ist die login shell so eingestellt, dass der Ort, wo ich bin, vor der Eingabe angezeigt wird. Ich mache den mal Gelb. ~ diese Linie steht für das home Verzeichnis des Users (in Unix Systemen hat normalerweise jeder Benutzer ein eigenes Heimatverzeichnis, in dem er schalten und walten darf), bei der Box also des Users root und der ist auf der box direkt in / daheim. cd .. wechselt jeweils ins übergeordnete Verzeichnis (das ist cd(Leerzeichen)PunktPunkt). pwd zeigt,

wo man gerade ist. Ich empfehle fleißiges Üben damit, später wird die Kenntnis des Aufbaus des Dateibaumes hilfreich sein.

```

root@dreambox:~> pwd
/
root@dreambox:~> ls
automount  dev          hdd          mnt          root          share          var
bin         etc          lib          proc         sbin          tmp            var_init
root@dreambox:~> cd sbin
root@dreambox:~/sbin> ls
automount          hdparm          loadkmap          pmap_set          start-stop-daemon
chroot             hotplug         logread           portmap           streamapes
dropbear           httpd           lsmod             poweroff          streamsec
dropbearkey        ifconfig        mke2fs            rdate            streamts
dropbearmulti      in.ftpd         mkfs.ext2         reboot            swapoff
e2fsck             in.telnetd     mkfs.ext3         reiserfsck       swapon
exportfs           inetd           mkswap            rmmmod           syslogd
fsck.ext2          init            mountd            route            telnetd
fsck.ext3          insmod          nfsd              scp              udhcpc
halt               klogd          pmap_dump         sfdisk           udpstreamapes
root@dreambox:~/sbin> cd bin
-sh: cd: can't cd to bin
root@dreambox:~/sbin> pwd
~/sbin
root@dreambox:~/sbin> cd /bin
root@dreambox:~/bin> ls
[          chgrp          dvbnet          free          lcdoff         mv            rm            tail          uptime
[[         chmod          dvbsnoop        gdaemon       lcdstuff       nc            rmdir         tar            usleep
ash        chown          echo            grep           ln              netstat       sed           test           vi
awk        clear          enigma          gunzip         logger         nslookup      sh            top            wc
basename   cp             env             gzip           login          passwd         showlogo     touch          wget
boot       date           eraseall        head           ls             pidof         sleep        true           which
bootmenu   dd             etherwake       hostname       md5sum         ping          smbmnt       tty            whoami
bunzip2    df             expr            id             mkdir          prockill     smbmount     udhcpc        xargs
busybox    dmesg          false           inadyn         mknod          ps            sort          umount        yes
bzcat      dream_lpd      find            kill           more           pwd           su            uname          zcat
cat        du             flashtool       killall        mount          reset         sync          uniq
root@dreambox:~/bin> cd ..
root@dreambox:~> pwd
/
root@dreambox:~> cd var
root@dreambox:~/var> ls
ani         etc          lock          mnt          scce          tmp           www
bin         keys         log           run          script        tuxbox
empty       lib          misc          scam         share         uninstall
root@dreambox:~/var> ls mnt
cdrom  cf      cifs   dvd    hdd    musik  nfs    test  usb
root@dreambox:~/var> cd /
root@dreambox:~> ls var/mnt
cdrom  cf      cifs   dvd    hdd    musik  nfs    test  usb
root@dreambox:~> cd bin
root@dreambox:~/bin> ls var/mnt
ls: var/mnt: No such file or directory
root@dreambox:~/bin> ls /var/mnt
cdrom  cf      cifs   dvd    hdd    musik  nfs    test  usb
root@dreambox:~/bin>

```

Wie zu sehen, haben die Entwickler von Gemini im Ordner /var/mnt bereits Einträge angelegt, die auf physikalischen Speicher verweisen können. Hier befindet sich auch ein Eintrag hdd, der als Eintrag für die Festplatte genutzt wird. Gemini hat damit eine gewisse eigene Ordnung, die ich für gut finde und achte. Es spricht aber nichts dagegen, andere Einträge zu verwenden und diese auch an vollkommen anderen Orten anzulegen. Bei anderen Images ist es deshalb unter Umständen nicht, wie hier gezeigt und viele Unix Systeme gehen jeweils eigene Wege. Das Verständnis sollte aber nun ausreichend sein, sich auch auf anderen Systemen zurecht zu finden und bewegen zu können.

Die Zuweisung der Einträge auf einen physikalischen Speicher passiert mit dem Befehl mount und diejenigen Einträge im Dateibaum, welche einem Speicher zugewiesen werden, nennt man mountpoint. Und grundsätzlich können alle Verzeichnis-Einträge, also Ordner im Dateibaum auch mountpoints sein. Wenn etwa alle Befehle, die im Verzeichnis /bin stehen nicht nur auf einem Unix System genutzt werden sollen, sondern von mehreren und deshalb auf einem zentralen Rechner liegen, dann kann dies durchaus realisiert werden. Oder, wenn jemand private Daten auf einem USB-Stick hat, kann er diese immer an die gleiche Stelle im System auf verschiedenen Rechnern einbinden und findet sie also auf jedem Rechner dann am gleichen Ort. Das Konzept hat wirklich große

Vorteile, verwirrt aber stets den Neuling, der erwartet eine konkretere Abbildung der physikalischen Verhältnisse zu finden. Für uns ist dies wichtig und die vorletzte Stufe der Vorbereitung auf den NFS, der sehr einfach ins Konzept eingereiht werden kann und sehr schnell abgehandelt ist, wenn die eben besprochenen Grundlagen beherzigt werden. Mountpoints sind Einträge (irgendwelche Ordner) im Dateibaum, die zunächst nichts enthalten, durch den Befehl mount aber einen physikalischen Speicher zugewiesen bekommen können. Der Befehl umount löst diese Bindung wieder. Einen eigenen Eintrag kann der Befehl mkdir erzeugen und touch kann eine Datei erzeugen. Dies nutze ich im folgenden Beispiel:

```

root@dreambox:~> cd /var/mnt
root@dreambox:/var/mnt> ls
cdrom cf      cifs  dvd   hdd   musik nfs   test  usb
root@dreambox:/var/mnt> mkdir pit234a
root@dreambox:/var/mnt> ls
cdrom cf      cifs  dvd   hdd   musik nfs   pit234a test  usb
root@dreambox:/var/mnt> mount
/dev/root on / type squashfs (ro)
none on /dev type devfs (rw)
proc on /proc type proc (rw,nodiratime)
devpts on /dev/pts type devpts (rw)
/dev/mtdblock/1 on /var type jffs2 (rw,noatime)
none on /tmp type ramfs (rw)
usbfs on /proc/bus/usb type usbfs (rw)
/dev/ide/host0/bus0/target0/lun0/part1 on /var/mnt/hdd type ext3 (rw)
nfsd on /proc/fs/nfsd type nfsd (rw)
root@dreambox:/var/mnt> ls pit234a
root@dreambox:/var/mnt> cd pit234a
root@dreambox:/var/mnt/pit234a> ls
root@dreambox:/var/mnt/pit234a> touch test.pit
root@dreambox:/var/mnt/pit234a> ls
test.pit
root@dreambox:/var/mnt/pit234a> mount /dev/ide/host0/bus0/target0/lun0/part1 /var/mnt/pit234a
root@dreambox:/var/mnt/pit234a> mount
/dev/root on / type squashfs (ro)
none on /dev type devfs (rw)
proc on /proc type proc (rw,nodiratime)
devpts on /dev/pts type devpts (rw)
/dev/mtdblock/1 on /var type jffs2 (rw,noatime)
none on /tmp type ramfs (rw)
usbfs on /proc/bus/usb type usbfs (rw)
/dev/ide/host0/bus0/target0/lun0/part1 on /var/mnt/hdd type ext3 (rw)
nfsd on /proc/fs/nfsd type nfsd (rw)
/dev/ide/host0/bus0/target0/lun0/part1 on /var/mnt/pit234a type ext3 (rw)
root@dreambox:/var/mnt/pit234a> ls
test.pit
root@dreambox:/var/mnt/pit234a> cd ..
root@dreambox:/var/mnt> ls pit234a
2x_mehr_wie Du.mp3      dream_festplatte.pdf   hanna.png              swapfile
AOL.mp3                enigma-sig-11.core.239 kathy.png              sync.sh
Anykey.mp3             enigma-sig-11.core.338 lost+found              tmp_dbox
Blindmann.mp3         for_pit                movie
Laterne.mp3           gemini.epg.dat         musik
akut                  gemini.epg.dat.md5     nfs
root@dreambox:/var/mnt> umount /var/mnt/pit234a
root@dreambox:/var/mnt> ls pit234a
test.pit
root@dreambox:/var/mnt>

```

Interessant ist sicher, dass im farbig markierten Teil die Zuordnung der Festplatte auf /var/mnt/pit234a bereits erfolgte, der Befehl ls aus dem Verzeichnis heraus aber noch den alten Inhalt zeigte. Erst nach Verlassen des Verzeichnisses wirkte nun ls wie erwartet. Der Befehl mount ohne weitere Optionen zeigt einen aktuellen Status der zugewiesenen mountpoints und ihrer Dateisysteme. Dass ich die gleiche Festplatte an zwei unterschiedlichen Orten gleichzeitig einbinden kann, wurde sicherlich auch ersichtlich. Bei solchen Konstruktionen kann es dann lustige Verwirrung geben, wenn diese beiden Orte unterschiedlichen Nutzern mit unterschiedlichen Rechten zugesprochen sind, diese aber nun den gleichen Inhalt serviert bekommen und möglicherweise noch ein Dateisystem finden, welches die Unix Dateirechte gar nicht nutzen kann, wie etwa msdos Dateisysteme. Da wird leicht verständlich, warum nur der Super User root das Recht hat, etwas zu mounten. Ähnlich vorsichtig sollte man bei Löschen, Verschieben oder Kopieren sein. Wenn es mir zum Beispiel einfiel, im obigen Verzeichnis alle Einträge zu löschen und neue mountpoints zu erstellen und dazu ein rm -rf /var/mnt durchführen würde, dann wäre auch alles gelöscht, was dort gerade gemountet ist. Also unter

Umständen nicht nur die lokale Platte, sondern auch welche, die aus dem Netzwerk importiert wurden. Dies ist durchaus nicht ungewöhnlich und ich selbst habe mir auf ähnliche Weise schon manches System zerstört. Es ist dabei einfach sehr wichtig, darauf zu achten, dass die Pfadangabe auch wirklich dorthin trifft, wohin man will. Besser doppelt prüfen, denn es passiert sehr leicht, einen Befehl ungewollt abzusetzen und es gibt keine Rückfrage: ein Unix System führt Befehle aus und diskutiert diese nicht mit dem Anwender. Es meldet auch keine Erfolge, sondern Fehler. Der Erfolg ist selbstverständlich, dafür wurde das System entwickelt und die Kompetenz des Users wird vorausgesetzt.

Und schließlich sollte man sehr vorsichtig sein, mountpoints zu verwenden, die bereits etwas enthalten. Es ist sicher nicht sehr tragisch, wenn ein lokal vorhandenes Verzeichnis /hdd/movie benutzt wird, um eine externe Datenquelle einzuhängen. Die Filme sind einfach weg und nichts weiter passiert. Sind auf der externen Quelle zufällig auch Filme in Dreambox Format drauf, können nun halt diese angeschaut werden, nicht mehr die lokal gespeicherten. Wird indessen ein solcher Ordner auf /bin oder /sbin gemountet, wo wichtige und System-relevante Befehle liegen, werden diese einfach nicht mehr gefunden und das System selbst wird hängen. Vielleicht nicht sofort, aber bald. Aus- und Einschalten würde hier helfen, denn der dramatische mount fände beim nächsten Start keine Anwendung (es sei denn, er würde automatisch ablaufen, was Gott verhüten möge, wenn der Super User es schon nicht konnte).

NFS

Womit wir ungefähr dort sind, wo wir hin wollten. Network File System ist das Unix typische Verfahren, wie entfernte Freigaben in einem System eingebunden werden. Es ist sehr alt und es gibt viele verschiedene Ausführungen, doch allen ist der grundlegende Aufbau gemeinsam.

Es gibt einen Server, der Freigaben exportiert und es gibt Clients, die auf diese zugreifen dürfen. Ein Server kann auch selbst Client sein und zwar sowohl bei anderen, als auch bei sich selbst. Dazu später noch was. Die Client Seite ist meist schnell erklärt und abgehandelt. Ist dieser Dienst in einem Unix System installiert und auf den Boxen ist dies allermeist automatisch der Fall, dann kann er genutzt werden und Freigaben ins lokale System einhängen. Analog zu dem oben gezeigten mounten einer lokalen Festplatte, geht dies mit dem Befehl mount und hier nutze ich nun das vorher angelegte Verzeichnis /var/mnt/pit234a um eine NFS Freigabe aus meinem Netzwerk dorthin zu mounten:

```
root@dreambox:~> mount
/dev/root on / type squashfs (ro)
none on /dev type devfs (rw)
proc on /proc type proc (rw,nodiratime)
devpts on /dev/pts type devpts (rw)
/dev/mtdblock/1 on /var type jffs2 (rw,noatime)
none on /tmp type ramfs (rw)
usbfs on /proc/bus/usb type usbfs (rw)
/dev/ide/host0/bus0/target0/lun0/part1 on /var/mnt/hdd type ext3 (rw)
nfsd on /proc/fs/nfsd type nfsd (rw)
root@dreambox:~> mount -t nfs -o rw 192.168.0.103:/dreamold/movie /var/mnt/pit234a
root@dreambox:~> mount
/dev/root on / type squashfs (ro)
none on /dev type devfs (rw)
proc on /proc type proc (rw,nodiratime)
devpts on /dev/pts type devpts (rw)
/dev/mtdblock/1 on /var type jffs2 (rw,noatime)
none on /tmp type ramfs (rw)
usbfs on /proc/bus/usb type usbfs (rw)
/dev/ide/host0/bus0/target0/lun0/part1 on /var/mnt/hdd type ext3 (rw)
nfsd on /proc/fs/nfsd type nfsd (rw)
192.168.0.103:/dreamold/movie on /var/mnt/pit234a type nfs
(rw,v3,rsize=32768,wsiz=32768,hard,udp,lock,addr=192.168.0.103)
root@dreambox:~> ls /var/mnt/pit234a
66
CHASSE _ PECHE - Chasse traditionnelle du sangl - 09_05_05.ts
CHASSE _ PECHE - Chasse traditionnelle du sangl - 09_05_05.ts.idx
Das Erste - die story - Der Griff nach dem Äl - Ein riskanter Wettlauf - 13_07_05.ts
Das Erste - die story - Der Griff nach dem Äl - Ein riskanter Wettlauf - 13_07_05.ts.idx
SR SÄDWEST Ferns_ - Aktueller Bericht am Sonntag - 07_08_05.eit
SR SÄDWEST Ferns_ - Aktueller Bericht am Sonntag - 07_08_05.ts
ZDFinfokanal - Lafers Culinarium - 17_05_05.eit
ZDFinfokanal - Lafers Culinarium - 17_05_05.ts
bilder
eit.txt
eitsweg.sh
oma_video
recordings.epl
sor
ts.txt
root@dreambox:~>
```

mount -t nfs -o rw 192.168.0.103:/dreamold/movie /var/mnt/pit234a mountet eine solche Freigabe aus meinem Netzwerk auf genau das Verzeichnis, wohin wir zuvor mal die lokale Festplatte eingebunden hatten. Dass dort normalerweise eine Datei test.pit liegt, ist dann egal. Davon kann nun nichts mehr gesehen werden, denn durch den mount Befehl wird unser Verzeichnis zum mountpoint und stellt den Inhalt der entfernten Freigabe dar. Die Option **-t nfs** gibt an, dass es sich um ein nfs Dateisystem handelt. Mit **-o rw** wird versucht, das Verzeichnis mit Lese- und Schreibrechten einzubinden. Das gelingt aber nur, wenn der NFS-Server entsprechend konfiguriert ist und dies auch erlaubt. Der Client ist damit bereits abgehandelt. Ein beliebter Fall ist nun, dass eine Dbox die Festplatte einer Dreambox benutzen soll. Es gilt das oben gesagte, nur, dass viele Images genau ein bestimmtes Verzeichnis wollen, wohin sie Filme speichern und woher sie diese auch lesen wollen. Auf Server-Seite muss dieses Verzeichnis zur Verfügung stehen und auf Client Seite genau dort eingebunden werden, wo das Image es erwartet.

Es ist einem Unix System vollkommen egal, ob es einen Stick oder eine entfernte oder eine lokale

Platte ins System einbindet und einen mountpoint innerhalb des Dateibaumes zuweist. Erst die Zuweisung, das mounten, bindet die jeweilige Ressource ein und macht sie nutzbar. Ansonsten bleibt nur ein Eintrag im Dateibaum, der unter Umständen leer ist. Deshalb wieder: die meisten Probleme bereitet das Verständnis des mountens, weniger der NFS Dienst selbst. Doch weiter in unserem Beispiel mit unterschiedlichen Boxen.

Ist eine Dreambox eine DM7000 und hat eine Platte, so wird das Verzeichnis `/hdd/movie` erwartet. Dieses movie liegt also auf der Festplatte selbst. Wird nun die gesamte Platte, also `/hdd exportiert`, dann muss darauf geachtet werden, dass eventuell auf der Client Seite auch nach `/hdd gemountet` wird, denn `/movie` liegt ja auf der Platte und wird mit exportiert und somit findet sich nach dem mount von `/hdd` nach `/hdd` auch `/hdd/movie` auf der Import-Box, eventuell eine DM7000 ohne Platte. Wird hingegen nur das Verzeichnis `/hdd/movie` exportiert, so muss auf der Client Box dieses nach `/hdd/movie gemountet` werden und entsprechend dieser mountpoint existieren, vielleicht extra angelegt sein. Dazu zeige ich später auch ein Beispiel mit einem USB Stick.

Client Seite, wie gesagt einfach. Ein Befehl (`mount -t nfs IP.der:freigabe:/Pfad/zur/Freigabe /Pfad/zu/mountpoint`) genügt und NFS geht.

Server Seite macht viel mehr Probleme. Aus der Erklärung zu Free-BSD:

Der Server benötigt folgende Daemonen:

Daemon	Beschreibung
<code>nfsd</code>	Der NFS-Daemon. Er bearbeitet Anfragen der NFS-Clients.
<code>mountd</code>	Der NFS-Mount-Daemon. Er bearbeitet die Anfragen, die <code>nfsd(8)</code> an ihn weitergibt.
<code>rpcbind</code>	Der Portmapper-Daemon. Durch ihn erkennen die NFS-Clients, welchen Port der NFS-Server verwendet.

Und eine Erläuterung aus http://de.wikipedia.org/wiki/Network_File_System, wo einiges zum Thema gefunden werden kann.

Schematischer Ablauf der Datenübertragung

Im Folgenden ist der prinzipielle Ablauf einer NFS-Kommunikation des zustandslosen NFS beschrieben. Szenario: Ein Nutzer des Client-Rechners möchte ein entferntes Verzeichnis ("`/directory`") öffnen und eine darin befindliche Datei ("`test`") anzeigen lassen.

Damit ein Datenaustausch zwischen NFS-Server und Client stattfinden kann, muss der NFS-Server gestartet und beim Portmapper registriert sein.

1. Client kontaktiert Portmapper und fragt nach dem Port des Mount-Daemons (`mountd`)
2. Portmapper gibt Portnummer für `mountd` heraus
3. Client kontaktiert `mountd` und fragt nach einem Filehandle für "`/directory`"
4. `mountd` gibt ein Filehandle 0 zurück
5. Client kontaktiert Portmapper und fragt nach dem Port für NFS (`nfsd`)
6. Portmapper gibt Portnummer für `nfsd` heraus
7. Client führt LOOKUP-Prozedur aus mit den Parametern Filehandle 0 und dem Dateinamen ("`test`")
8. `nfsd` gibt Filehandle 1 für Datei ("`test`") heraus
9. Client führt READ-Prozedur aus mit dem Parameter Filehandle 1
10. `nfsd` gibt Inhalt der Datei ("`test`") zurück (Daten)

OK, das kann sich vermutlich niemand merken, doch prinzipiell sollte bekannt sein, dass mehrere Dienste laufen müssen und auch passende Einstellungen brauchen. Insbesondere muss bekannt sein, was zu exportieren ist und mit welchen Rechten, bzw, wer Zugriff haben darf.

In einem Unix System ist die Hauptkonfigurationsdatei die `/etc/exports` und im ersten Zitat habe ich dazu die man page eingefügt. Die Dienste werden dann gestartet, indem ein entsprechender Aufruf in einer Startkonfigurationsdatei erfolgt. Hier wird auch für die richtige Reihenfolge gesorgt. Wird dann später in der `/etc/exports` etwas geändert, müssen die Dienste neu gestartet werden, entweder von Hand, oder durch Neustart des Systems. Die Startkonfiguration ist meist in der `/etc/rc.conf` oder einer ähnlichen Datei beschrieben. Hier nochmal ein Auszug aus einer Beschreibung zu NFS, die vielleicht schon alles dazu wesentliche erklärt, bevor wir dann auf der Box landen werden:

<http://www.virtualuniversity.ch/guides/unix/68.html>

NFS - Network File System

NFS erlaubt das Einbinden einer beliebigen Platten-Partition eines Computers in das Dateisystem eines anderen Rechners. Dazu wird entweder per `mount`-Befehl oder über die Datei `/etc/fstab` bzw. `/etc/vfstab` angegeben, welches Verzeichnis eines fernen Rechners auf welches lokale Verzeichnis gemountet werden soll.

Die einzige Konfigurationsdatei für den NFS-Dämon des NFS-Servers ist die Datei `/etc/exports` mit folgendem Format:

```

Verzeichnis-Pfad Rechnernamen (Optionen)
Links steht das Verzeichnis, das der NFS-Server exportieren soll, beispielsweise /home/public
oder /cdrom. In der Mitte stehen die Rechner, die Zugriff auf das Verzeichnis haben sollen, und
danach in Klammern die Optionen. ACHTUNG: Beachten Sie das Leerzeichen zwischen den Rechnernamen
und den Optionen! Nach jeder Änderung der Datei müssen Sie den Portmapper neu starten und dann den
NFS-Dämon neu einlesen lassen.
Die zugriffsberechtigten Client-Rechner können Sie auf drei Arten definieren:
Ein einzelner Rechnername oder eine einzelne IP-Nummer. Damit gestatten Sie nur diesem einen
Rechner den Zugriff auf das Verzeichnis. Wenn Sie den Rechnernamen angeben, sollte in der Datei
/etc/hosts seinem Namen eine IP-Adresse zugeordnet sein.
Domain-Eintrag mit Jokerzeichen (* oder ?). Damit können Sie allen Rechnern einer Domain den
Zugriff gestatten, z. B.: *.netzmafia.de
IP-Netzwerknummern. Durch Eingabe eines Subnetzes mit Netzmaske. Wenn Sie z. B.
192.168.253.255/255.255.255.255 angeben, haben alle Rechner im Subnetz 192.168.253.0 Zugriff auf
das Verzeichnis.
Die gebräuchlichsten Optionen sind:
rw: Read-Write gibt den Clients Lese- und Schreibrechte im Verzeichnis.
ro: Read-Only gibt den Clients nur Leserecht (Voreinstellung).
noaccess: Verbieht Clients den Zugriff auf Unterverzeichnisse.
root-squash: Dateien mit User/Group root werden bei den Clients einem anonymen Eigentümer und
einer anonymen Gruppe zugeordnet.
no-root-squash: Das Gegenteil zu obiger Option.
Im folgenden Beispiel sollen folgende Zugriffe möglich sein: Auf die erste CD-ROM bekommen
die Clients nur Lesezugriff. Der Rechner boss.netzmafia.de benötigt root-Zugriff auf /install
knecht.netzmafia.de darf ebenfalls auf /install zugreifen, allerdings ohne daß Dateien des
Benutzers root als solche erscheinen. Die Home-Verzeichnisse aller Benutzer auf dem Server
exportiert der Server an alle Rechner im Subnetz, damit die Benutzer auf allen Clients das gleiche
Home-Verzeichnis bekommen:
# /etc/exports
#
/cdrom      *.netzmafia.de (ro)
/install   boss.netzmafia.de (rw,no-root-squash) \
          knecht.netzmafia.de (ro,root-squash)
/home      192.168.253.255/255.255.255.255

Beim Client werden die Verzeichnisse unter Angabe des Servernamens (durch Doppelpunkt
getrennt) eingebunden. Das kann entweder per mount-Kommando geschehen, z. B. durch:
mount nfs.netzmafia.de:/cdrom -t nfs /opt/cdrom

oder in der Datei /etc/fstab, z. B.:
.
.
nfs.netzmafia.de:/cdrom /opt/cdrom nfs ro 0 0
nfs.netzmafia.de:/home /home nfs defaults 0 0

```

Wer die wenigen Zeilen von oben nun innig liest, bekommt alles zu NFS erklärt, was in der Praxis von Bedeutung ist. Allermeist ist die Hürde für einen Neuling in Sachen Unix eben gar nicht der NFS-Server oder Client, sondern das Verständnis um das freie mounten, das Zuweisen von einem Speichermedium zu einem beliebigen Verzeichniseintrag im lokalen Dateibaum. Deshalb haben wir dazu viel mehr geübt, als über NFS geredet.

Auf der Dreambox ist dies alles ganz anders. Zwar existiert eine Startkonfigurationsdatei in /etc/init.d/rcS und diese ist bei einem Gemini Image ein echt ausgefeiltes Stück und kümmert sich um manchen Spezialfall, aber einen NFS Server startet sie nicht direkt, sondern sie ruft dazu eine separate Startdatei auf, die sich nun ihrerseits um den Start der nötigen Dienste mit den erforderlichen Einstellungen kümmert. Diese Datei ist **/var/script/nfs_server_script.sh** und wie die Endung verrät, ein script. Ein script ist etwas ähnliches, wie ein Programm. In ihm werden Befehle und Entscheidungen ausgeführt, dazu muss es ausführbar sein. Es kann auch über die Fernbedienung ein Menü erreicht werden, das auf diese Datei zugreift. Über Blue-Panel und Extras Einstellungen kommt man zu einem Punkt, indem Einstellungen zum NFS Server gemacht werden können und dieser dann auch gestartet werden kann. Was dabei passiert, ist folgendes. Mit diesen Daten wird die /var/script/nfs_server_script.sh gefüttert und ausgeführt. Die Fernbedienung kann dabei nicht mehr Einstellungen vornehmen, als die Entwickler planten. Wir können dies schon, wenn wir wollen. Dazu muss nur ein Texteditor benutzt werden, der Unix-konform arbeitet und die geänderte Datei muss später unbedingt wieder ausführbar sein. In unserem Falle sieht sie so aus:

```
-rwxr-xr-x 1 root root 3740 Dec 28 18:39 /var/script/nfs_server_script.sh
```

und das entspricht `chmod 755 /var/script/nfs_server_script.sh` und darauf möchte ich an dieser Stelle nicht weiter eingehen, nur so viel: `chmod` ändert die Dateirechte für die angegebene Datei und

755 entspricht dem Wert, den rwxrwxrwx angibt, wenn jeweils eine Dreiergruppe als Zahl im Binärsystem angesehen wird und jeweils ein Buchstabe einer Eins und ein Minus einer Null entspricht ($rwx = 111 = 2^2 + 2^1 + 2^0 = 4 + 2 + 1 = 7$ | $r-x = 4 + 1 = 5$ | $rwxr-xr-x = 755 \Rightarrow 7$ für User, 5 für Group, 5 für Others).

Wird dieses script direkt aufgerufen, zeigt es verschiedene Optionen:

```
root@dreambox:~> /var/script/nfs_server_script.sh
Usage: nfs-kernel-server & portmap {start|stop|reload|force-reload|restart}
```

ich zeige mal einen restart, der allerdings missglückte und nicht alle meine Änderungen eingelesen hatte, stattdessen war ein Neustart der Box erfolgreich.

```
root@dreambox:~> /var/script/nfs_server_script.sh restart
Stopping portmap daemon: portmap.
Stopping NFS kernel daemon: mountd
  nfsd
.
Unexporting directories for NFS kernel daemon...
done.
Starting portmap daemon: portmap.
exportfs: could not open /var/lib/nfs/etab for locking
Exporting directories for NFS kernel daemon...
done.
Starting NFS kernel daemon:
  nfsd
  mountd
.
not registered:      100005      1      udp      690      mountd
not registered:      100005      1      tcp      693      mountd
not registered:      100005      2      udp      690      mountd
not registered:      100005      2      tcp      693      mountd
not registered:      100005      3      udp      690      mountd
not registered:      100005      3      tcp      693      mountd
root@dreambox:~> Connection closed by foreign host.
```

Die letzte Zeile ist ein Indikator dafür, dass die Box ausgeschaltet wurde (nicht standby), denn Unix Systeme erwarten normalerweise eine ständige Verfügbarkeit, auch der Netzwerk Ressourcen und erleiden dabei gelegentlich Schiffbruch, wenn einfach mal ein Rechner ausfällt. Auch dieses gilt es zu bedenken: wenn Netzwerkressourcen später automatisch eingebunden werden sollen, aber gar nicht vorhanden sind, dann kann ein echtes Problem entstehen, wenn dieser Zustand nicht durch ein entsprechend intelligentes script vorab geprüft wird. In der Regel ist es so, dass nichts und niemand prüft, sondern erwartet wird, dass die Ressource vorhanden ist. Schließlich gab ein autorisierter User, nämlich der Super User root, den Befehl und auf dessen Kompetenz wird vertraut.

Jedenfalls, wenn unser NFS Server erfolgreich gestartet wurde, finden sich mit ps einige laufenden Prozesse, die zu ihm gehören.

```
219 1          440 S    /sbin/portmap
233 root        SW    [nfsd]
234 root        SW    [nfsd]
235 root        SW    [nfsd]
237 root        SW    [lockd]
238 root        SW    [rpciod]
239 root        792 S    /sbin/mountd
```

Wenn diese Namen auftauchen, sieht es schon sehr gut aus, dann wird es vermutlich laufen.

In ähnlicher Manier kann dann der Server auch wieder gestoppt werden, eine Änderung an der script Datei vorgenommen und neu gestartet werden, manchmal durch einen Neustart der Box.

Nun zu diesem script, das den NFS-Server auf der Box mit der gewünschten Konfiguration versorgt und startet. Da steht dieses bei mir drin:

```
#!/bin/sh
#

set -e

# Read config
NFSSERVER_ON=1
NFS_CL_IP=192.168.0.0
NFSNETMASKE=255.255.255.0
DIRECTORY=/hdd/movie
DIRECTORY2=/hdd/musik
MOUNTD_PORT_ON=1
MOUNTD_PORT=2049
OPTIONS=rw
# end

DESC="NFS kernel daemon"
LIB=/lib/modules/2.6.9/kernel/fs
RPCNFSDCOUNT=3

if [ -d /usr/sbin ] ; then
    PORTMAP_BIN=/usr/sbin/portmap
    NFSD_BIN=/usr/sbin/nfsd
    MOUNTD_BIN=/usr/sbin/mountd
    EXPORTFS_BIN=/usr/sbin/exportfs
else
    PORTMAP_BIN=/sbin/portmap
    NFSD_BIN=/sbin/nfsd
    MOUNTD_BIN=/sbin/mountd
    EXPORTFS_BIN=/sbin/exportfs
fi

#####_NFSSERVER_#####
# Exit if required binaries are missing.
[ -x $NFSD_BIN ] || exit 0
[ -x $MOUNTD_BIN ] || exit 0
[ -x $EXPORTFS_BIN ] || exit 0
#####_NFSSERVER_#####
#####_PORTMAP_#####
[ -x $PORTMAP_BIN ] || exit 0
#####_PORTMAP_#####

if [ $NFSSERVER_ON -ne 0 ]; then
    if [ ! -d /var/lib ] ; then mkdir /var/lib; fi
    rm -R /var/lib/nfs 2> /dev/null || /bin/true
    ln -sf /tmp /var/lib/nfs
    touch /var/lib/nfs/rmtab
fi

# See how we were called.
case "$1" in
    start)
        if [ $NFSSERVER_ON -ne 0 ]; then
            #####_PORTMAP_#####
            echo -n "Starting portmap daemon:"
            echo -n " portmap"
            start-stop-daemon --start --quiet --exec $PORTMAP_BIN
            echo "."

            if [ -f /var/run/portmap.upgrade-state ]; then
                echo -n "Restoring old RPC service information..."
                sleep 1 # needs a short pause or pmap_set won't work. :(
                pmap_set </var/run/portmap.upgrade-state
                rm -f /var/run/portmap.upgrade-state
                echo "done."
            fi
            #####_PORTMAP_#####
            sleep 1
            #####_NFSSERVER_#####
            $EXPORTFS_BIN -i -o $OPTIONS $NFS_CL_IP/$NFSNETMASKE:$DIRECTORY
            $EXPORTFS_BIN -i -o $OPTIONS $NFS_CL_IP/$NFSNETMASKE:$DIRECTORY2
            if ! grep -qs exportfs /proc/modules ; then
                insmod $LIB/exportfs/exportfs.ko
            fi
        fi
    ;;
)
```

```

        if ! grep -qs nfsd /proc/modules ; then
            insmod $LIB/nfsd/nfsd.ko
        fi

        if ! grep -qs /proc/fs/nfs /proc/mounts ; then
            mount -t nfsd nfsd /proc/fs/nfsd
        fi

        echo "Exporting directories for $DESC..."
        echo "done."
        echo "Starting $DESC:"
        echo " nfsd"
        start-stop-daemon --start --quiet --exec $NFSD_BIN -- $RPCNFSDCOUNT
        echo " mountd"
        #/sbin/rpcinfo -u localhost nfs 3 >/dev/null 2>&1
        if [ $MOUNTD_PORT_ON -ne 0 ]; then
            start-stop-daemon --start --quiet --exec $MOUNTD_BIN -p $MOUNTD_PORT
        else
            start-stop-daemon --start --quiet --exec $MOUNTD_BIN
        fi

        echo "."
        #####_NFSSERVER_#####
    fi
;;
stop)
    #####_PORTMAP_#####
    echo -n "Stopping portmap daemon:"
    echo -n " portmap"
    start-stop-daemon --stop --quiet --exec $PORTMAP_BIN
    echo "."
    #####_PORTMAP_#####
    sleep 1
    #####_NFSSERVER_#####
    echo "Stopping $DESC: mountd"
    start-stop-daemon --stop --quiet --name mountd --user 0
    sleep 1
    echo " nfsd"
    start-stop-daemon --stop --quiet --name nfsd --user 0 --signal 2
    echo "."

    echo "Unexporting directories for $DESC..."
    $EXPORTFS_BIN -au
    #rm -rf /var/lib/nfs
    umount /proc/fs/nfsd
    #rmmod nfsd.ko
    #rmmod exportfs.ko
    rm -rf /tmp/etab
    rm -rf /tmp/rmtab
    rm -rf /tmp/xtab
    echo "done."
    #####_NFSSERVER_#####
;;
reload | force-reload)
    #####_NFSSERVER_#####
    echo "Re-exporting directories for $DESC..."
    $EXPORTFS_BIN -r
    echo "done."
    #####_NFSSERVER_#####
;;
restart)
    pmap_dump >/var/run/portmap.state
    $0 stop
    sleep 1
    $0 start
    if [ ! -f /var/run/portmap.upgrade-state ]; then
        sleep 1
        pmap_set </var/run/portmap.state
    fi
    rm -f /var/run/portmap.state
;;
*)
    echo "Usage: nfs-kernel-server & portmap {start|stop|reload|force-reload|restart}"
    exit 1
;;
esac
exit 0

```

Wer noch nie ein solches script gesehen hat, erschrickt vielleicht erstmal und denkt, dass dies wirklich nur was für ausgemachte Experten und Programmierer ist. Ich selbst verstehe davon auch fast nichts. Was ich farbig markiert habe, sind wesentliche Einstellungen. `NFS_CL_IP=192.168.0.0` ist ziemlich am Anfang gesetzt und am Anfang passiert überhaupt zunächst mal, dass verschiedene Variablen gesetzt werden. Diese können hernach im script benutzt werden, was einfacher ist, als ständig an allen Stellen des scripts, wo diese Werte vorkommen sollen, etwas ändern zu müssen. Die Variable `NFS_CL_IP` steht für die IP des zugelassenen Client-Rechners. Hier habe ich es mit 0 am Ende versucht und das Ergebnis war, wie gewünscht, dass nämlich alle Rechner in meinem Netz Zugriff auf die Freigaben haben. Andere Versionen habe ich nicht getestet, aber sicher lassen sich so auch unterschiedliche Verzeichnisse für unterschiedliche IP bereit halten. Mehrere IP Adressen sind im Vorbild, der `/etc/exports` durch Leerzeichen getrennt, ob es hier auch so gehandhabt wird, habe ich nicht getestet.

Ein wenig möchte ich zu diesem script erklären. In meinem Beispiel habe ich nämlich nicht nur das übliche `/hdd/movie` exportiert, sondern auch ein weiteres Verzeichnis auf der Platte dieser Box, das `/hdd/musik` heißt und meine gesammelten Lieder enthält, die ich so von allen Rechnern im Netzwerk einfach erreichen und etwa auch von anderen Boxen abspielen kann. Dazu benutze ich die vorgegebene Struktur des scripts und fügte eine Variable `DIRECTORY2=/hdd/musik` ein, die später im Aufruf des NFS Servers eingesetzt wird. Da habe ich einfach die Zeile des `export`-Befehls kopiert und unverändert wieder eingefügt, dann den Variablennamen angepasst. Das hat funktioniert. Ob es auch funktioniert hätte, im ersten Aufruf die zweite Variable zusätzlich einzufügen, weiß ich nicht.

Alternativ hätte ich natürlich einfach `/hdd` komplett exportieren können, doch dazu hätte ich, wie bereits erwähnt, einiges auf meinen Clients ändern müssen und letztlich wollte ich auch andere Bereiche der Platte gar nicht exportiert wissen, sondern eben nur `/hdd/movie` und `/hdd/musik`.

Ein Hinweis noch. `mount -t nfsd nfsd /proc/fs/nfsd` ist ein eigenartiger `mount` Befehl den ich nicht näher beschreiben will. Er sorgt dafür, dass ein laufender NFS Server auf einem System ein bestimmten Eintrag erhält. Deshalb ist die Zeile in der Ausgabe des Befehles `mount`

```
nfsd on /proc/fs/nfsd type nfsd (rw)
```

ein echter Hinweis darauf, dass der Dienst gestartet ist.

Und damit sollte auch der Server grundsätzlich behandelt sein, denn auf die vielen möglichen Optionen will ich nicht näher eingehen. Wird versucht, in einem Unix System einen NFS Server einzusetzen, dann sollte man der dort gelieferten Dokumentation folgen, um genauere Einzelheiten herauszufinden. Soll ein NFS Server auf einem anderen Image als einem Gemini gestartet werden, so ist daran zu denken, dass nicht alle Verzeichnisse beschreibbar sind. In einem laufenden Image kann deshalb ein solcher Server nicht überall eingebunden werden. Für die Macher von Gemini war die Wahl, ausführbare Dateien, also Programme, in das Verzeichnis `/sbin` zu legen. Zusammen mit einigen System-Programmen, die unsere bekannten Links auf `busybox` enthalten, stehen dort verschiedene andere Programme, die in einem anderen Image möglicherweise nicht zu finden sind. Für den NFS werden dabei benötigt: `exportfs`, `mountd`, `nfsd` und `portmap`. Auch diese Fälle will ich nicht weiter verfolgen, nur erklären, dass `/sbin` normalerweise nicht beschreibbar ist und daher nicht in Frage kommt, wenn jemand in einem anderen Image einen NFS Server einsetzen will. Da vermutlich nur `/var` und `/hdd` beschreibbar ist, müsste alles hier untergebracht werden, vielleicht in einem Verzeichnis `NFS` und dann die verschiedenen Aufrufe mit ihren Pfadangaben angepasst werden. Etwas Arbeit, aber machbar.

Schließlich existieren auch NFS Server und Client Programme für Windows und dazu kann ich nichts weiter sagen, weil ich das gar nicht kenne. Es ist aber klar, dass diese Programme nicht anders können, als sich an die Unix Vorgaben zu halten und sofern dürfte nun jeder damit umgehen können. Der NFS Dienst, wie er von der Firma SUN auf deren Solaris verwendet wird, ist etwas unterschiedlich und die neueste Version, NFS4 weicht auch in wichtigen Details von den hier gemachten Feststellungen ab.

Fehlersuch-Strategie

Wenn nun ein Server läuft und Clients nicht darauf zugreifen können, so kann dies an verschiedenen Einstellungen und Problemen liegen. Wichtig ist, dass alle Änderungen an Textdateien und scripten mit einem Unix-konformen Editor durchgeführt werden. Für ein Unix System ist das ohnehin so, für Windows müssen entsprechende Programme gefunden und angewendet werden. Ich kann dazu aber wieder mal nichts sagen, denn mit Microsoft ihrem Zeugs will ich mich nicht auch noch beschäftigen. Weiters ist wichtig, dass in Unix Systemen Groß- und Kleinschreibung beachtet wird. Es ist also nicht egal, ob da /bin oder /BIN steht, das wären zwei vollkommen unterschiedliche Einträge. Und schließlich noch ein Hinweis für Windows Nutzer: Dateirechte kann Windows nicht so, wie dies von Unix erwartet wird. Werden Dateien erst auf ein Windows System übertragen, dort geändert und später zurückgespeichert, gehen in der Regel die Dateirechte verloren. Dies kann sehr wichtig sein. Trotz dieser Hinweise, können sich Fehler einschleichen. Dass NFS ein Netzwerkdienst ist und deshalb ein richtig konfiguriertes Netzwerk voraussetzt, ist klar. Schließlich ging ich auch davon aus, dass telnet funktioniert und da gilt diese Voraussetzung ja ebenso. NFS soll aber fast immer nur im internen Netz funktionieren und dazu ist nur wichtig, dass diese Einstellungen passen. Ein Gateway oder Router oder DNS Server muss hierfür nicht gefunden werden. Wenn sich die betroffenen Rechner mit einem ping finden können, geht in der Regel auch NFS. Es gibt vielleicht die Ausnahme, dass bestimmte NFS Dienste (hier zählen vor allem die Windows Programme dazu) nicht das modernere TCP Protokoll nutzen, sondern UDP und dazu muss dieser Dienst eventuell erst noch auf einem System freigeschaltet werden. Bei den Dreamboxen und den meisten Unixen sind diese Dienste beide in Betrieb. Ein ping auf einem Unix Rechner funktioniert ähnlich, wie bei einem Windows Rechner. Allerdings sendet bei Windows der Befehl immer nur eine gewisse Anzahl von pings, während in der Grundeinstellung bei Unix endlos pings abgesetzt werden. Eine legale Möglichkeit, dies zu beenden, ist das gleichzeitige Drücken von STRG+C.

```
root@dreambox:~> ping 192.168.0.103
PING 192.168.0.103 (192.168.0.103): 56 data bytes
64 bytes from 192.168.0.103: icmp_seq=0 ttl=64 time=0.9 ms
64 bytes from 192.168.0.103: icmp_seq=1 ttl=64 time=0.3 ms
64 bytes from 192.168.0.103: icmp_seq=2 ttl=64 time=0.4 ms
64 bytes from 192.168.0.103: icmp_seq=3 ttl=64 time=0.3 ms
64 bytes from 192.168.0.103: icmp_seq=4 ttl=64 time=0.4 ms
64 bytes from 192.168.0.103: icmp_seq=5 ttl=64 time=0.3 ms
64 bytes from 192.168.0.103: icmp_seq=6 ttl=64 time=0.4 ms
64 bytes from 192.168.0.103: icmp_seq=7 ttl=64 time=0.3 ms
64 bytes from 192.168.0.103: icmp_seq=8 ttl=64 time=0.4 ms

--- 192.168.0.103 ping statistics ---
9 packets transmitted, 9 packets received, 0% packet loss
round-trip min/avg/max = 0.3/0.4/0.9 ms
root@dreambox:~>
```

Wenn es einmal nicht geht, ist dieser Test der erste und einfachste, der durchgeführt werden sollte..

Hilfreich kann es weiter sein, erst mal die Freigabe der Clients für alle zu ermöglichen und wie in meinem Beispiel die NFS_CL_IP so zu setzen, dass der letzte Wert eine Null ist. Auch die möglichen Optionen sollten so weit wie machbar reduziert werden, also eventuell nur das rw setzen, ebenfalls wie in meinem Beispiel. Nicht alle Clients verstehen alle möglichen Optionen und wenn erst mal sicher ist, dass der Dienst grundsätzlich läuft und genutzt werden kann, dann kann auch optimiert und probiert werden.

Ob der NFS Server läuft, dafür haben wir bereits einige Kenntnisse erworben, die uns weiter helfen. Ob die verschiedenen Dienste überhaupt gestartet sind, zeigt uns der Befehl ps. Ob der NFS Server auch läuft, kann die Zeile in der Ausgabe des Befehls mount deutlich machen und was tatsächlich exportiert wird und wie, das zeigt der Befehl exportfs -v

```
root@dreambox:~> exportfs -v
/var/mnt/hdd/musik
    192.168.0.0/255.255.0.0 (rw,wdelay,root_squash)
/var/mnt/hdd/movie
    192.168.0.0/255.255.0.0 (rw,wdelay,root_squash)
```

Wenn diese Dinge auf dem Server gefunden werden, ist es höchst wahrscheinlich, dass er richtig funktioniert und läuft. Es kann dann auch einfach ein Test erfolgen und ein exportiertes Verzeichnis lokal wieder gemountet werden. Das ist eine an und für sich unsinnige Konstruktion, die aber der Fehlersuche dienen kann. Funktioniert es nämlich, dass der Server sein eigener Client werden kann, dann muss ein Problem auf der entfernten Client Seite bestehen, wenn es trotzdem nicht klappt. Dazu muss der eigene Server die Berechtigung haben, als Client zu wirken und die NFS_CL_IP im Beispiel des Gemini Image entsprechend gesetzt sein, wie schon erwähnt, empfehle ich hier zunächst eh die Null am Ende der IP-Adresse, damit alle innerhalb des Netzwerks als Client arbeiten können. Der mount Befehl wird dann akkurat identisch eingesetzt, als wolle ein entferntes Verzeichnis gemountet werden, nur lasse ich hierbei jegliche Optionen weg.

```
root@dreambox:~> mount -t nfs 192.168.0.96:/hdd/musik /var/mnt/pit234a
root@dreambox:~> mount
/dev/root on / type squashfs (ro)
none on /dev type devfs (rw)
proc on /proc type proc (rw,nodiratime)
devpts on /dev/pts type devpts (rw)
/dev/mtdblock/1 on /var type jffs2 (rw,noatime)
none on /tmp type ramfs (rw)
usbfs on /proc/bus/usb type usbfs (rw)
/dev/ide/host0/bus0/target0/lun0/part1 on /var/mnt/hdd type ext3 (rw)
nfsd on /proc/fs/nfsd type nfsd (rw)
192.168.0.96:/hdd/musik on /var/mnt/pit234a type nfs
(rw,v3,rsize=32768,wsiz=32768,hard,udp,lock,addr=192.168.0.96)
root@dreambox:~> ls /var/mnt/pit234a/*.mp3
/var/mnt/pit234a/All I Want To Do Is Make Love To You - Heart.mp3
/var/mnt/pit234a/All Right Now - Free.mp3
/var/mnt/pit234a/Always Something There To Remind Me - Tin Tin Out.mp3
/var/mnt/pit234a/Angels - Robbie Williams.mp3
/var/mnt/pit234a/As Long As You Love Me - Backstreet Boys.mp3
/var/mnt/pit234a/Axel F - Clock.mp3
/var/mnt/pit234a/Baby Baby - Corona.mp3
/var/mnt/pit234a/Bat Out Of Hell - Meatloaf.mp3
/var/mnt/pit234a/Better of Alone - DJ Jurgen.mp3
/var/mnt/pit234a/Blaze Of Glory - Bon Jovi.mp3
/var/mnt/pit234a/Broken Wings - Mr Mister.mp3
/var/mnt/pit234a/Could you be Loved - Bob Marley & The Wailers.mp3
/var/mnt/pit234a/Dancing Queen - ABBA.mp3
/var/mnt/pit234a/Don't Stop Movin' - Livin' Joy.mp3
/var/mnt/pit234a/Eye Of The Tiger - Survivor.mp3
/var/mnt/pit234a/I Knew You Were Waiting - George Michael.mp3
/var/mnt/pit234a/I'll Be Missing You - Puff Daddy Featuring faith Evans.mp3
/var/mnt/pit234a/It's Alright - East 17.mp3
/var/mnt/pit234a/Land Of The Rising Son.mp3
/var/mnt/pit234a/Stand By Me - Ben E King.mp3
/var/mnt/pit234a/The Final Countdown - Europe.mp3
/var/mnt/pit234a/Whisky In The Jar - Thin Lizzy.mp3
/var/mnt/pit234a/Wild Wild West - Will Smith.mp3
/var/mnt/pit234a/You Might Need Somebody - Shola Ama.mp3
/var/mnt/pit234a/You're Still The One - Shania Twain.mp3
root@dreambox:~> umount /var/mnt/pit234a
root@dreambox:~> ls /var/mnt/pit234a/*.mp3
ls: /var/mnt/pit234a/*.mp3: No such file or directory
root@dreambox:~>
```

Das Weglassen jeglicher Optionen schadet also für das mounten nicht. Verstehen sich Client und Server, handeln sie diese untereinander aus. Nur manchmal ist es nötig, bestimmte Rechte auch anzufordern.

Über die vorgestellten Möglichkeiten hinaus, gibt es sehr komplexe Zusammenhänge. Weil aber meiner Einschätzung nach bereits weit mehr als 95% aller Fälle mit dem hier gezeigten Wissen zum Erfolg gebracht werden können, möchte ich nicht auf weitere Details eingehen.

In einem Gemini Image wird der NFS Server mit den gefundenen Einstellungen bei jedem Bootvorgang automatisch gestartet. In anderen Fällen muss hierfür vielleicht Sorge getragen werden. Meist hilft ein Eintrag in die Datei /var/etc/init. Automatisches mounten kann durch einen Eintrag in der /var/tuxbox/config/enigma/conf erfolgen oder bei Gemini kann der automount zum Einsatz kommen. Dies aber nur als Hinweise, ich denke, das ist dann für jeden einfach zu bewältigen, wenn die hier vermittelten Techniken zum Einsatz kommen. Zur Verdeutlichung füge ich hier mal alle Zeilen ein, die mit solch einem Eintrag aus Enigma, also mittels Fernbedienung gefunden, in die /var/tuxbox/config/enigma/config eingetragen werden.

```

root@dreambox:~> cat /var/tuxbox/config/enigma/config | grep nfs
i:/elitedvb/network/nfs0/automount=00000000
i:/elitedvb/network/nfs0/fstype=00000000
i:/elitedvb/network/nfs0/options=00000002
u:/elitedvb/network/nfs0/ip=c0a80061
s:/elitedvb/network/nfs0/extraoptions=nolock,rsize=8192,wsiz=8192
s:/elitedvb/network/nfs0/ldir=/mnt/97movie
s:/elitedvb/network/nfs0/password=
s:/elitedvb/network/nfs0/sdir=/hdd/movie
s:/elitedvb/network/nfs0/username=
root@dreambox:~>

```

c0a80061 ist eine zunächst vielleicht ungewohnte Darstellung der IP als Zahlenreihe in Hexadezimalschreibweise. Jeweils zwei Zeichen bilden einen Tripel der IP(4) Adresse. c0 = 192 , a8 = 168, 00 = 000 und 61 = 097. ldir=/mnt/97movie zeigt das lokale Verzeichnis, also den mountpoint und sdir=/hdd/movie das Verzeichnis auf dem Server, also die Freigabe. Was der Eintrag options=00000002 bedeutet, erinnere ich nicht mehr, doch wenn automount auf 1 gestellt wird, wird dieser mount automatisch bei Systemstart versucht.

Diese Zeilen können einfach in die Sektionen der config eingefügt werden, also i: zu anderen Zeilen, die so beginnen und entsprechend weiter. Dies kann dann Enigma beim Start lesen und ausführen.

Dabei gibt es eine kleine Hürde zu meistern. Wird die Box einfach wie normal neu gestartet, verschwindet diese Einstellung immer wieder. Die Box überschreibt dann das config mit dem aktuellen Inhalt aus dem Speicher und fügt so zum Beispiel auch den gerade laufenden Sender ein, damit dieser nächstes Mal wieder gestartet wird. Deshalb muss Enigma dazu gebracht werden, nicht die config mit Speicherinhalt zu überschreiben. Am einfachsten geschieht dies mit einem Befehl in telnet: killall enigma. killall enigma führt nicht nur dazu, dass Enigma schlagartig beendet wird, es startet auch gleich wieder neu und so kann auch direkt die Wirkung der Änderung beobachtet werden.

An dieser Stelle ein weiterer Hinweis auf die Funktionsweise der Box: Enigma ist ein weiteres Programm, das die Steuerung der Box übernimmt und Befehle der Fernbedienung entgegennimmt. In Enigma werden die Menüs angelegt, die dem Benutzer zur Verfügung stehen und es kann zahlreiche Funktionen ausführen und starten. Enigma wird als ein fest geschnürtes Programm auf das Image einer Box geschnallt und die Funktionen können sehr unterschiedlich geordnet sein. Gemini macht vieles anders, als andere. Die /var/tuxbox/config/enigma/config ist die globale Konfigurationsdatei und hier finden sich viele Dinge, die Enigma beim Start wissen will. Es ist also kein script, wie zuvor behandelt. In /var/tuxbox/config/enigma/ liegen auch die Senderlisten, Timer Einträge, Bouquets und solche Dinge. Diese können hier direkt verändert, gespeichert oder zurückgeschrieben werden. Es gilt aber wie eben erklärt, dass killall enigma ausgeführt werden muss, damit nicht alle Änderungen durch den Inhalt des Speichers überschrieben werden.

Aus diesem Grund ist es auch ratsam, nicht ein einziges derartiges Verzeichnis für mehrere Boxen zu betreiben, die als NFS-share freigegeben und gemountet werden könnten. Ansonsten wäre das eine Idee, alle Einstellungen der Boxen und Senderlisten gleich zu halten. Es wird aber immer die Box, die ausschaltet, den Inhalt der temporären Informationen bestimmen. Beim nächsten Einschalten der anderen Box, würde hier der Sender gestartet, den die zuletzt ausgeschaltete laufen hatte und auch die eventuell geänderte Senderliste dieser Box dargestellt werden. Auch Timereinstellungen könnten sich mächtig ins Gehege kommen, doch für derartige Probleme kann vielleicht auch eine Konstruktion mit Links ins System Abhilfe schaffen. Hier gibt es also einiges an Betätigungsmöglichkeiten für den angehenden Unix Nutzer. Die Boxen sind schöne Spielzeuge und nicht nur zum Fernseh-glutzen gut. Besonders der Unix Nutzer kann sich freuen und auf jene Hilfsprogramme verzichten, die für Windows Nutzer mit der Box unverzichtbar scheinen.

Zusätzliches Beispiel:

Weil das Konzept des freien mountens meist sehr viel schwieriger für den Neuling ist, als der Umgang mit dem NFS, möchte ich durch ein weiteres Beispiel die neu gewonnene Kenntnis festigen. Ein recht amüsanter Versuch, der gerade Dreambox Besitzern vielleicht nützlich ist, aber generell Neulingen in Unix eine Hilfe sein mag: Ich stecke nun meinen neuen Aldi 2GB USB-Stick in die Box und will darauf eine kurze Filmszene aufzeichnen, danach nehme ich ihn weg und setze ihn in diesem Rechner ein und sehe mir diese Aufnahme an. Was ich bereits weiß, ist, dass der Stick in einem DOS üblichen FAT oder VFAT formatiert ist und ich denke, dass die Box und mein Rechner dieses Format beherrschen. Ansonsten könnte es nicht gelingen, bevor ich nicht ein geeignetes Dateisystem auf dem Speichermedium installiert hätte.

Zunächst, unmittelbar nach dem Anstecken des USB-Sticks, gebe ich dmesg ein um zu sehen, was der Kernel als Gerät erkannt hat. Ich zeige nun nur die letzten Zeilen, denn diese sind ohnehin chronologisch angeordnet und somit interessieren uns genau diese, die am Ende der Ausgabe stehen.

```
root@dreambox:~> dmesg
...die letzten Zeilen:...
<6>usb 1-1: new full speed USB device using address 2
<6>scsi0 : SCSI emulation for USB Mass Storage devices
<5> Vendor: Tevion Model: USB Drive Rev: PMAP
<5> Type: Direct-Access ANSI SCSI revision: 02
<5>SCSI device sda: 4030464 512-byte hdwr sectors (2064 MB)
<5>sda: Write Protect is off
<7>sda: Mode Sense: 23 00 00 00
<5>SCSI device sda: drive cache: write through
<6> /dev/scsi/host0/bus0/target0/lun0: p1
<5>Attached scsi removable disk sda at scsi0, channel 0, id 0, lun 0
<5>Attached scsi generic sg0 at scsi0, channel 0, id 0, lun 0, type 0
<7>USB Mass Storage device found at 2
root@dreambox:~>
```

Tatsächlich steht da was von USB und sogar das Modell und der Herstellername tauchen auf, kein Zweifel, mein neuer Stick wurde erkannt. Es erscheint sogar die Zuordnung zu einer der Treiberdateien in /dev. Ich finde hier die Mitteilung interessant, dass der Stick als p1, also Partition 1 zu mounten sein wird. Sticks werden oft nicht als disc, also Festplatte behandelt, sondern wie eine Partition auf einer Festplatte. Warum das so ist, weiß ich nicht und manche USB-Geräte und Sticks erfordern auch ein disc und manchmal werden auch lokale Platten als part gemountet. Gemini berücksichtigt beides beim Booten und zwar wird nämlich beides in der /etc/init.d/rcS versucht, wenn beim Start des Systems ein Gerät gefunden wird. Wir ließen nicht Booten und daher zeigt uns mount auch keinerlei eingebundenes Gerät. Gemini würde ein USB-Gerät nach /var/mnt/usb mounten.

```
root@dreambox:~> mount
/dev/root on / type squashfs (ro)
none on /dev type devfs (rw)
proc on /proc type proc (rw,nodiratime)
devpts on /dev/pts type devpts (rw)
/dev/mtdblock/1 on /var type jffs2 (rw,noatime)
none on /tmp type ramfs (rw)
usbfs on /proc/bus/usb type usbfs (rw)
/dev/ide/host0/bus0/target0/lun0/part1 on /var/mnt/hdd type ext3 (rw)
nfsd on /proc/fs/nfsd type nfsd (rw)
```

Nun versuche ich zunächst als disc zu mounten, obwohl ich zuvor schon sah, dass p1 erwartet wird.

```
root@dreambox:~> mount /dev/scsi/host0/bus0/target0/lun0/disc /var/mnt/usb
mount: Mounting /dev/scsi/host0/bus0/target0/lun0/disc on /var/mnt/usb failed: Invalid
argument
root@dreambox:~> mount /dev/scsi/host0/bus0/target0/lun0/part1 /var/mnt/usb
root@dreambox:~> umount /var/mnt/usb
```

Wie erwartet schlägt der erste Versuch fehl. Im zweiten Anlauf gelingt der mount, denn es erscheint keine Fehlermeldung. Allerdings wollte ich gar nicht nach /var/mnt/usb mounten und deshalb mache ich dies wieder rückgängig.

Ich will ja eine Aufnahme auf den Stick machen und da muss er den Ort einnehmen, wo meine Aufnahmen gemäß der Software des Images landen. Das ist normalerweise /hdd/movie.

```

root@dreambox:~> mount /dev/scsi/host0/bus0/target0/lun0/part1 /hdd
root@dreambox:~> mount
/dev/root on / type squashfs (ro)
none on /dev type devfs (rw)
proc on /proc type proc (rw,nodiratime)
devpts on /dev/pts type devpts (rw)
/dev/mtdblock/1 on /var type jffs2 (rw,noatime)
none on /tmp type ramfs (rw)
usbfs on /proc/bus/usb type usbfs (rw)
/dev/ide/host0/bus0/target0/lun0/part1 on /var/mnt/hdd type ext3 (rw)
nfsd on /proc/fs/nfsd type nfsd (rw)
/dev/scsi/host0/bus0/target0/lun0/part1 on /var/mnt/hdd type vfat
(rw,nodiratime, fmask=0022, dmask=0022)

```

Tatsächlich ist nun also mein Stick an dem Ort gemountet, wo die lokale Festplatte erwartet wird. Eine Aufnahme, mit der Fernbedienung gestartet, misslingt aber. Klar, es gibt nun kein Verzeichnis /hdd/movie, denn der Stick ist ja noch leer und somit steht in /hdd nun gar nichts drin.

```

root@dreambox:~> ls hdd
hdd
root@dreambox:~> ls hdd/
root@dreambox:~> umount /hdd

```

Ich hätte also die Möglichkeit, mit `mkdir /hdd/movie` das benötigte Verzeichnis anzulegen, entscheide mich aber anders. Ich will den leeren Stick direkt als /hdd/movie mounten, dann sollte es ebenfalls funktionieren.

```

root@dreambox:~> mount /dev/scsi/host0/bus0/target0/lun0/part1 /hdd/movie
root@dreambox:~> mount
/dev/root on / type squashfs (ro)
none on /dev type devfs (rw)
proc on /proc type proc (rw,nodiratime)
devpts on /dev/pts type devpts (rw)
/dev/mtdblock/1 on /var type jffs2 (rw,noatime)
none on /tmp type ramfs (rw)
usbfs on /proc/bus/usb type usbfs (rw)
/dev/ide/host0/bus0/target0/lun0/part1 on /var/mnt/hdd type ext3 (rw)
nfsd on /proc/fs/nfsd type nfsd (rw)
/dev/scsi/host0/bus0/target0/lun0/part1 on /var/mnt/hdd/movie type vfat
(rw,nodiratime, fmask=0022, dmask=0022)

```

Das ist recht interessant. Jeder neue mount überschreibt einen bestehenden, weshalb eben in /hdd nichts zu sehen war, nach einem umount aber wieder die alte Beziehung deutlich wird. Es erscheint nun /var/mnt/hdd als lokale Festplatte wieder und auf ein Verzeichnis, das auf dieser Platte angelegt ist, wird nun der Stick gemountet. Das geht, denn sobald die Platte gemountet ist erscheinen auch die darauf befindlichen Ordner im Dateibaum und können als mountpoint dienen.

Um es nochmal deutlich zu machen: wenn /hdd/movie gebraucht wird, kann es entweder als leerer Eintrag erzeugt und im Dateibaum angelegt werden, somit als mountpoint dienen (wenn keine lokale Platte da ist), es kann ein Eintrag movie auf dem Speichermedium erzeugt und dieses nach /hdd gemountet werden (was normalerweise bei einer lokalen Platte gemacht wird) oder es kann auch der Eintrag movie auf dem in /hdd eingebundenen Speichermedium als mountpoint benutzt werden, wie hier gezeigt.

Nun gibt es also ein /hdd/movie (genauer gesagt, ein /var/mnt/hdd/movie und weil /hdd ein Link nach /var/mnt/hdd ist, existiert auch /hdd/movie) und dahinter befinden sich nach dem mount nicht mehr die Daten aus dem entsprechenden Verzeichnis der eingebundenen Festplatte, sondern unter diesem mountpoint liegt nun der Inhalt des Stick am USB Port der Box.

Nun eine Aufnahme und ls zeigt, ob sich etwas tut.

```

root@dreambox:~> ls /hdd/movie
06-12-29 - Phoenix - Pompeji - Der letzte Tag.eit
06-12-29 - Phoenix - Pompeji - Der letzte Tag.ts
recordings.epl
root@dreambox:~>

```

Wahrhaftig, der Stick füllt sich mit der Aufnahme und dem begleitenden .eit File und mit der recordings.epl, welche die Liste der aufgenommenen Filme enthält und nun will ich nur kurz das Problem erwähnen, dass diese recordings.epl im RAM Bereich der Box parat ist und wenn keine solche Datei auf /hdd/movie gefunden wird, kopiert sich die Datei aus dem Speicher in das unvollständige Verzeichnis und enthält nun Einträge, die gar nicht gültig sind. Es gibt die Filme nicht

mehr, die von recordings.epl erwähnt werden, denn diese lagen auf der lokalen Platte, nicht auf dem Stick. Erkennbar ist dies, weil alle diese Filme in Größe 0B angezeigt werden. Nur der letzte Eintrag, welcher dem Film auf dem Stick entspricht, hat eine reale Größenangabe.

Genug, ich habe die Aufnahme angehalten und kann sie tadellos auf der Box abspielen, als wäre sie lokal auf der Platte vorhanden. Nun will ich den Stick loswerden und umounten und damit hatte ich meine Not, was ich hier nicht verschweigen will. Es schadet sicher nicht, die Probleme, die auftreten beim Namen zu nennen anstatt sie zu verschweigen.

```
root@dreambox:~> umount /hdd/movie
umount: Couldn't umount /var/mnt/hdd/movie: Invalid argument
root@dreambox:~> umount /hdd/movie
umount: Couldn't umount /var/mnt/hdd/movie: Invalid argument
root@dreambox:~> umount -f /hdd/movie
umount: forced umount of /var/mnt/hdd/movie failed!
root@dreambox:~> umount -f /hdd/movie
umount: forced umount of /var/mnt/hdd/movie failed!
root@dreambox:~> umount -f /dev/scsi/host0/bus0/target0/lun0/part1
umount: forced umount of /dev/scsi/host0/bus0/target0/lun0/part1 failed!
root@dreambox:~> umount -f /dev/scsi/host0/bus0/target0/lun0/part1
umount: forced umount of /dev/scsi/host0/bus0/target0/lun0/part1 failed!
root@dreambox:~> umount -f /hdd/movie
umount: forced umount of /var/mnt/hdd/movie failed!
```

An dieser Stelle habe ich die Idee, dass es möglicherweise deshalb nicht funktioniert, weil ja /hdd/movie vom NFS Server exportiert wird und es vielleicht eine Schutzfunktion ist, diesen Eintrag nicht lösen zu können, weil dann entfernte Teilnehmer, die darauf zugreifen, plötzlich überrascht würden. Deshalb stoppe ich erst den NFS Server und wirklich funktionierte dann der umount und anschließend startete ich den NFS Server wieder:

```
root@dreambox:~> /var/script/nfs_server_script.sh stop
Stopping portmap daemon: portmap.
Stopping NFS kernel daemon: mountd
nfsd
.
Unexporting directories for NFS kernel daemon...
done.
root@dreambox:~> umount -rf /hdd/movie
root@dreambox:~> /var/script/nfs_server_script.sh start
Starting portmap daemon: portmap.
exportfs: could not open /var/lib/nfs/etab for locking
Exporting directories for NFS kernel daemon...
done.
Starting NFS kernel daemon:
nfsd
mountd
.
root@dreambox:~>
```

Dass ich den Stick nun hier, auf diesem PC wo ich gerade schreibe, mounten und den Film abspielen konnte, mag man mir einfach glauben. Ich zeige keine Beweise dafür. Es geht, vollkommen problemlos. Die Box läuft nun auch wieder wie gewohnt weiter und der zuletzt aufgenommene Film, der auf dem Stick und nicht auf der Platte ist, wird mit Größe 0B angezeigt, während alle anderen wieder eine reale Größe anzeigen. Das ist im Verhalten gleich, wenn ein Film direkt auf der Festplatte gelöscht wird und nicht über das Menü der Fernbedienung (also vielleicht: rm über telnet). Es ist einfach in diesem Falle Abhilfe zu schaffen: die vier Zeilen, die den Film in der /hdd/movie/recordings.epl beschreiben, können einfach gelöscht werden. Anschließend killall enigma. Gemini kann einfach über Webif: recover movies und bildet aus gefundenen, spielbaren Filmen, eine neue recordings.epl.

Ob nun ein USB Stick oder eine entfernte Festplatte gemountet wird, ist doch der gleiche Vorgang. Deshalb habe ich so viel Zeit hierauf verwandt, denn der NFS ist relativ einfach zu behandeln, wenn diese Grundlagen verstanden sind.

Obwohl es stets so beschrieben wird, als funktionierte ein Restart der einzelnen Anwendungen ohne Probleme, sind meine Erfahrungen auf der Dreambox anders. Häufig hilft hier nur ein Neustart des Systems, was für Unix Rechner sehr unüblich ist. Die Hintergründe kenne ich nicht, gebe es nur als Hinweis oder Tip weiter, wenn mal nichts funktionieren will, wie es eigentlich sollte.