

Debian GNU/Linux  
Systemadministration

Lukas Beeler  
Huber+Monsch  
Interne Schulung

---

1. November 2003  
*Rev* : 17

## Vorwort

Dies stellt das erste Dokument der H+M internen Schulung mit dem Betriebssystem Debian GNU/Linux dar. Es widmet sich an Einsteiger auf dem Gebiet GNU/Linux mit Vorkenntnissen im Bereich Computer.

Dieses Dokument ist nicht vollständig. Nicht sämtliche darin vorkommenden Begriffe und Befehle werden erklärt. Es ist die Aufgabe des Lernenden, sich fehlende Informationen selbst zu besorgen. Dies zu erlernen ist Teil der Übung.

## Inhaltsverzeichnis

<b>1</b>	<b>Einführung</b>	<b>4</b>
1.1	Allgemeines	4
1.2	Philosophie	4
1.2.1	KISS	4
1.2.2	RTFM	4
1.2.3	Everything is a file	4
1.3	Verwendung	5
1.4	Lerntipps	5
<b>2</b>	<b>Scripten mit Shell</b>	<b>6</b>
2.1	Einführung	6
2.2	Beispiel	6
2.3	Lösung	7
2.4	Ad-Hoc Scripting	8
2.5	File Descriptors	9
2.5.1	Was ist ein fd?	9
2.5.2	Ändern von fd's	9
2.6	Quoting	10
2.6.1	Wie gebe ich das Zeichen > aus?	10
2.7	Variablen	10
2.8	Tools	11
2.9	Resumee	11
<b>3</b>	<b>Das Dateisystem</b>	<b>12</b>
3.1	Allgemeines	12
3.2	Der Aufbau	12
3.3	Inode	12
3.3.1	Hardlinks	13
3.3.2	Softlinks	14
3.4	Hierarchie	15
3.5	mount	16
3.6	fsck	17
<b>4</b>	<b>Das Rechtesystem</b>	<b>18</b>
4.1	Owner	18
4.2	Group	18
4.3	Mode	19
4.4	Ändern des Mode	19
4.5	Ändern von Owner und Group	20

4.5.1	chown	20
4.5.2	chgrp	20
4.5.3	User und Group gleichzeitig	21
<b>5</b>	<b>Paketverwaltung</b>	<b>22</b>
5.1	aptitude	22
5.2	apt	24
5.2.1	/etc/apt/sources.list	24
5.2.2	/etc/apt/apt.conf	24
5.2.3	apt-get	24
5.2.4	apt-cache	24
5.3	dpkg	25
<b>6</b>	<b>Userverwaltung</b>	<b>26</b>
6.1	/etc/passwd	26
6.2	/etc/group	26
6.3	/etc/shadow	26
6.4	Bearbeitung	26
<b>7</b>	<b>Verwaltung von Netzwerkinterfaces</b>	<b>27</b>
7.1	ip	27
7.1.1	lo konfigurieren	27
7.1.2	Interfaces anzeigen	27
7.1.3	Routen erstellen	28
7.1.4	Routen anzeigen	28
7.1.5	Hilfe anzeigen	28
7.2	tc	28
7.3	/etc/network/interfaces	29
7.3.1	ifup	29
7.3.2	ifdown	29
7.4	iptables	30
<b>8</b>	<b>Verwaltung von Software Raids</b>	<b>31</b>
8.1	Einrichten der Partitionen	31
8.2	Erstellen des RAIDs	32
8.3	Erstellen des Dateisystemen	32
8.4	Deaktivieren einer Disk	33
<b>9</b>	<b>Verwaltung von Diensten</b>	<b>34</b>
9.1	sysvinit	34
9.1.1	Starten	34
9.1.2	Stoppen	34
9.1.3	Neustarten	34
9.2	runit	34
9.2.1	Starten	34
9.2.2	Stoppen	34
9.2.3	Neustarten	34
<b>10</b>	<b>Signale</b>	<b>35</b>

<i>INHALTSVERZEICHNIS</i>	3
<b>11 Logfiles</b>	<b>36</b>
11.1 Dateinamen . . . . .	36
11.2 Timestamps . . . . .	37
11.3 Konfiguration, Rotation . . . . .	37
<b>12 strace</b>	<b>38</b>
<b>13 Netiquette/Problemlösung</b>	<b>39</b>
<b>14 Software bei Huber+Monsch</b>	<b>40</b>
14.1 Service-Management . . . . .	40
14.2 HTTP/FTP Proxy . . . . .	40
14.3 SMTP Proxy . . . . .	40
14.4 Paketfilter . . . . .	40
14.5 IP-IP Tunneling . . . . .	40
<b>A Tipps</b>	<b>41</b>
<b>B Credits</b>	<b>42</b>
<b>C Legal Stuff</b>	<b>42</b>

# 1 Einführung

## 1.1 Allgemeines

Linux ist ein UNIX-artiger Kernel. Es stellt jedoch kein UNIX Derivat dar, da es von Grund auf neu geschrieben wurde. Wichtig ist zu wissen, das Linux lediglich ein Kernel, nicht aber ein komplettes Betriebssystem ist. Dies ist wohl der radikalste Unterschied zu kompletten Betriebssystemen wie z.B. {Free,Net,Open}BSD, Solaris, etc. pp.

Dieser Unterschied zieht Vor- und Nachteile mit sich. Z.b. ist es möglich, ein System zu bauen das exakt den eigenen Bedürfnissen entspricht. Dies ist jedoch meist ein ziemlicher Aufwand, da man sich die einzelnen Programme zusammensuchen darf.

Der Aufwand, aus Linux ein komplettes Betriebssystem zu machen übernehmen sogenannte Distributoren. Auch hier kann wieder zwischen Kommerziellen Distributionen wie SuSE, RedHat, Mandrake und nicht-kommerziellen Distributionen wie Debian, Slackware, Gentoo unterschieden werden. Technisch gesehen haben die nicht-kommerziellen Distributionen eigentlich immer die Nase vorne. Allerdings verfügen sie nicht über die Zertifizierungen, die man z.B. beim Betrieb einer Oracle Datenbank benötigen würde.

## 1.2 Philosophie

Folgendes dürfte für sämtliche UNIX-ähnlichen Betriebssysteme gelten.

### 1.2.1 KISS

Keep it small and simple.

Es werden viele kleine Tools erstellt, welche in sich gesehen nichts besonderes können. Diese können jedoch im Normalfall miteinander verbunden werden, um am Ende ein Gesamtprodukt zu ergeben. Das bedeutet, das es im Normalfall keine Schlüsselfertigen All-in-One Produkte gibt, und zur Lösung von Problemen die Intelligenz des Admins gefordert ist. Diesem Prinzip verdankt \*nix seine enorme Flexibilität.

### 1.2.2 RTFM

Read the fine manual.

Von einem \*nix-Admin wird erwartet, das er sich selbst fortbilden kann, und über ein gutes Auffassungsvermögen verfügt. Die Dokumentation zu den einzelnen Programmen begleitet den Admin täglich, ebenso wie die Suchmaschine Google und das Usenet. Hier ist klar Eigeninitiative gefragt, und nicht die Weiterbildung in Kursen.

### 1.2.3 Everything is a file

Unter \*nix-Betriebssystemen wird im Normalfall sämtliche Hardware durch ein File dem Userspace (Gegensatz zu Kernelspace) zur Verfügung gestellt. Dies hat den Vorteil, das z.B. ein ISO Image einer CD exakt gleich behandelt wird wie eine CD in einem CD Laufwerk. Hierzu später mehr.

### 1.3 Verwendung

Wofür kann Linux bei H+M benutzt werden?

- Server jeglicher Art, mit Ausnahme von:
  - Exchange–Groupware–Funktionalität
  - TAPI-basiertes CTI
- Debugging von Netzwerkproblemen (Laptop, Ethernet–Bridge)
- Paketfilterung
- Application Level Gateway

### 1.4 Lerntipps

Dieser Stoff deckt nicht alles ab, was man können muss. Es wäre auch ziemlich gewagt, dies von einem Stoff zu behaupten. Auch enthält dieser Stoff Fehler. Als Schüler ist also ihre Wachsamkeit und Eigeninitiative gefordert. Benutzen sie Google, denken sie selbst nach, stellen sie bei Unklarheiten sofort fragen.

Denken sie nicht, das sie nach diesem Kurs ein System problemlos administrieren können. Sie werden dann wohl über die theoretischen Grundlagen verfügen, aber noch nicht über die Erfahrungen, was diese Grundlagen alles für Auswirkungen haben. Eine stetige Weiterbildung (man kann sich durchaus auch ohne Kurse weiterbilden) und auch Arbeit auf diesem Gebiet ist für das Erlangen von Erfahrung unerlässlich.

## 2 Scripten mit Shell

### 2.1 Einführung

Wieso dieses Thema als erstes? Ich bin der Meinung das man bei einem stark fordernden Unterricht besser lernen kann, als bei einem dem man ohne Mühe folgen kann, aber sich am Ende doch nichts gemerkt hat.

Ziel ist es hier, von Anfang an eine Verbindung von diesem Stoff zur Realität herzustellen. Als Lernender ist ihre vollste Aufmerksamkeit und Mitarbeit gefordert. Beachten sie das sie für dieses Kapitel die theoretischen Grundlagen noch nicht haben. Ziel ist es, zuerst einmal ein Feeling zu entwickeln, welches danach noch mit passender Theorie verstärkt wird.

Als Administrator für \*nix Systeme ist man meistens auch, wenn auch nur zu einem kleinen Teil, Programmierer. Das schreiben von kleinen Scripts, aber auch kleinen Programmen, gehört zum Alltag eines jeden Sysadmins. Meist werden Programme geschrieben, die genau die Anforderungen erfüllen, welche gestellt sind.

### 2.2 Beispiel

Fullbackup einer Maschine. Dieses Fullbackup soll es ermöglichen, die Kiste wieder problemlos booten zu können, durch Auswechseln der Festplatte.

**Setup** Maschine A ist, von welcher nächtlich ein Backup erstellt werden soll. Maschine B ist unser Backup Rechner, auf dem sämtliche Daten gespeichert werden.

#### Maschine A

```
[ [Disk 0]
  [Disk 1] ] RAID1, System
```

#### Maschine B

```
[ [Disk 0]
  [Disk 1] ] RAID1, System
[ [Disk 0]
  [none ] ] RAID1, failed disk 1, Backup Ziel
```

**Umsetzung** Wie wird nun so ein Backup nächtlich durchgeführt?

Mit dem uns zur Verfügung stehenden Baukasten, ist so eine Aufgabe leicht zu lösen.

Als erstes brauchen wir eine Netzwerkverbindung zwischen den beiden Maschinen. Diese soll verschlüsselt sein, und Authentifikation soll ebenfalls stattfinden.

Welches Tool wählen Sie? .....

Als zweites brauchen wir ein Tool, das über diese Netzwerkverbindung unsere Daten überträgt. Dies muss natürlich so geschehen, das Symlinks, Hardlinks und Devices erhalten bleiben.

Welches Tool wählen Sie? .....

Als drittes brauchen wir ein Tool, welches die obigen zwei Tools jeweils jede Nacht aufruft.

Welches Tool wählen Sie? .....

### 2.3 Lösung

Damit haben wir bereits eine simple Backuplösung, die sich für komplette Maschinen, aber nicht für Daten eignet. Vergleichen sie nun Ihre Lösung mit meiner, und ob die endgültige Funktion ungefähr gleich ist. (Zu diesem Problem existieren zahlreiche Lösungen, es gibt kein richtig/falsch).

#### Maschine A

- SSH2 Key, 2048bit, kein Passwort
- Mittels uschedule wird nun jede Nacht ein rsync job aufgerufen.

```
crontab -e
0 0 * * * rsync -a -e ssh / maschineb:/var/backup/hosts/A/
```

#### Maschine B

- Public Key von Maschine A nach /root/.ssh/authorized\_keys

Ziemlich simpel, oder? Die Festplatte aus Maschine B kann nun einfach ausgebaut werden, und in einen beliebigen Computer eingebaut werden. Dieser erfüllt dann sämtliche Aufgaben von Maschine A.

## 2.4 Ad-Hoc Scripting

Als Eingabeumgebung hat man ständig eine Shell vor sich. Diese kann sehr viel mehr, als bloss Tab-Completion<sup>1</sup> und Globbing<sup>2</sup>. Shell ist eine Turing-Vollständige Programmiersprache, mit der sehr viele Probleme vollständig gelöst werden koennen.

Unter Ad-Hoc Programmen/Scripten versteht man kurze Codeschnipsel, welche meist nur ein oder zweimalig verwendet werden. Aus diesem Grund ist es wichtig, das diese Möglichst schnell geschrieben werden können. Lesbarkeit und Erweiterbarkeit ist bei solchen Programmen sekundär. Shell eignet sich hierfür ideal, da sie für das ausführen von Alltags-Tasks optimiert ist.

```
for file in `find / -name run` ; do
    cat $file
done
```

Dieses Beispiel zeigt den Inhalt sämtlicher Dateien mit dem Namen 'run' an. Selbiges Beispiel kann auch anders geschrieben werden, diesmal in einer Whitespace-Sicheren Version:

```
find / -name run -print0 | xargs -0 cat
```

An diesen beiden Beispielen haben wir die am häufigsten verwendeten Elemente von Shell bereits gesehen.

- |  
Die Pipe lenkt den Output eines Programmes in den Input eines anderen Programmes um
- for  
Der for Loop funktioniert anders als bei anderen Programmiersprachen (er ist eigentlich ein for each Loop). Mit ihm kann über eine Liste von Dingen iteriert werden, die standardmässig mit Whitespace seperatiert werden.
- find  
Findet Dateien, sehr umfangreich
- xargs  
Wandelt STDIN in Argumente eines Programmes um
- cat  
Gibt den Inhalt einer Datei aus

---

<sup>1</sup>Vervollständigung

<sup>2</sup>Auflösen von Wildcards wie \*

## 2.5 File Descriptors

### 2.5.1 Was ist ein fd?

Ein File-Descriptor ist entweder ein Ein- oder Ausgabekanal. Er kann auf eine Konsole, auf ein File, oder auf eine Netzwerkverbindung zeigen. Selbstverständlich auch auf einen anderen fd (Pipe!). Im Normalfall hat ein Programm drei fd's.

- fd 0 (STDIN)  
über diesen fd nehmen die meisten Programme Input entgegen
- fd 1 (STDOUT)  
über diesen fd geben die meisten Programme Output aus
- fd 2 (STDERR)  
über diesen fd geben die meisten Programme Fehlermeldungen aus

### 2.5.2 Ändern von fd's

Shell kann diese fd's umleiten:

```
cat foobar | less
```

Dies lenkt STDOUT von `cat` nach STDIN von `less` um. Dies ist funktional äquivalent zu:

- `less foobar`  
`less` liest *foobar* selbst
- `less < foobar`  
*STDIN* von `less` ist *foobar*
- `cat < foobar | less`  
*STDIN* von `cat` ist *foobar*, *STDOUT* von `cat` ist *STDIN* von `less`

Diese vier Möglichkeiten sind alle äquivalent. Die zwei Besten sind jedoch: `less foobar` und `less < foobar`, da diese kein `cat` benötigen (useless use of cat).

Momentan haben wir ausschliesslich STDIN und STDOUT betrachtet. Doch auch STDERR lässt sich auf die gleiche Weise manipulieren.

```
cat foobar 2>file
```

Dies lenkt *fd2* (STDERR) in *file* um. Fehlermeldungen werden nun also in *file* geschrieben, und nicht ausgegeben.

```
cat foobar 2>&1
```

Dies lenkt *fd2* nach *fd1* um.

```
cat foobar 2>&1 1>file
```

Dies lenkt *fd2* nach *fd1* um, und *fd1* zu *file*.

**ACHTUNG:**

*fd2* zeigt nun dorthin, wo *fd1* ursprünglich zeigte, und nicht auf *file*. Dies bedeutet, dass bei vertauschter Reihenfolge *fd2* ebenfalls in *file* umgelenkt wird.

Komplett äquivalent ist folgende Schreibweise:

```
cat foobar 2>&1 >file
```

## 2.6 Quoting

Auch bekannt als Escaping. Viele selten verwendete Zeichen sind Steuerzeichen für die Shell. Manchmal jedoch braucht man diese Zeichen trotzdem als reguläre, und nicht als Steuerzeichen. In diesem Fall muss man diese Zeichen quoten oder escapen.

### 2.6.1 Wie gebe ich das Zeichen > aus?

```
echo >
```

Funktioniert nicht, das Zeichen wird von der Shell interpretiert.

```
echo \>
echo ">"
echo '>'
```

Sind in diesem Beispiel äquivalent. Mittels \ kann ein einzelner Charakter von der Shell escaped werden. Bei einem String, der in " " gesetzt ist, werden nicht mehr alle Shell-Steuerzeichen ausgeführt (ACHTUNG: einige werden doch!), Variablensubstitution wird normal durchgeführt. Bei einem String, der in ' ' gesetzt ist, wird überhaupt nichts mehr geändert.

## 2.7 Variablen

Variablen zu benutzen ist ziemlich einfach.

```
lb@naru:test % var=foobar
lb@naru:test % echo $var
foobar
```

Es ist auch möglich, die Ausgabe eines Programmes einer Variable zuzuweisen.

```
lb@naru:test % var=`date`
lb@naru:test % date
Tue Jun  3 12:14:40 CEST 2003
lb@naru:test % echo $var
Tue Jun  3 12:14:19 CEST 2003
```

Will man nun, dass eine Variable nicht substituiert wird, muss man zu Quoting greifen, das weiter oben bereits beschrieben wurde.

```
lb@naru:test % var=foobar
lb@naru:test % echo '$var'
$var
lb@naru:test % echo \$var
$var
lb@naru:test % echo "$var"
foobar
```

## 2.8 Tools

- `awk` Textmanipulationen in Flatfile DB's (komplex)
- `cut` Textmanipulationen in Flatfile DB's (leicht)
- `sed` (Stream Editor) Textmanipulationen
- `fgrep` Suchen nach Strings
- `grep` Suchen mittels regex (posix)
- `diff` Vergleichen von Dateien
- `tail` Ausgabe des Dateiendes
- `head` Ausgabe des Dateianfangs
- `ln` (link) Erstellen von Links (hard und soft)
- `sort` Zeilenweises Sortieren von Ausgaben
- `tar` (tape archiver) Erstellen von Archiven
- `cpio` (copy files to and from archives) Erstellen von Archiven
- `wc` (word count) Zählen von Zeichen, Wörtern und Zeilen
- `gzip` Komprimieren
- `date` Anzeigen des Datums
- `tr` (translate) Ersetzen, Löschen von Buchstaben
- `cat` (concatenate) Ausgeben von Dateien nach STDOUT

## 2.9 Resümee

Anstelle von Shell koennen auch andere Programmiersprachen für die gleichen Aufgaben verwendet werden. Perl und Shell sind häufigsten Sprachen, die für kleine Administrationsarbeiten eingesetzt werden. Beides sind grösstenteils 'write only' Sprachen. D.h., sie zu lesen ist ziemlich schwer, schreiben jedoch ziemlich einfach. Für Shell und Perl muss man ein Feeling entwickeln. Alternativen zu Shell und Perl sind C (für umfangreichere Aufgaben, die Programmentwicklung ist hier wesentlich aufwändiger), `exectline` (sehr spezialisiert), `python`, `ruby`. `Execline` ist sehr leicht les- und schreibbar, jedoch erfordert es einen durchdachteren Programmaufbau als Shell/Perl. Ich persönlich setze Shell für Quick-Hacks und evtl. Startscripts, und `exectline` für service run-Scripts. Anfängern empfehle ich, sich zumindest ansatzweise mit Shell anzufreuden (Ad-Hoc Programme, zur schnellen Problemlösung), und sich dann entweder in Shell, Perl, `python` oder `ruby` zu vertiefen.

## 3 Das Dateisystem

Unix-Dateisysteme kennen traditionell keine ACL's, wie sie bei Windows vorhanden sind. Vielfach befinden sich jedoch ACL Extensions zu den gängigen Filesystemen in den meisten \*nix Systemen. Bei einigen sind diese bereits ausgereift (Solaris, AIX, HP-UX), bei anderen im späten Teststadium (Linux, \*BSD). Die meisten "alten" \*nix Sysadmins mögen ACL's jedoch nicht allzusehr, weswegen sie praktisch nie zum Einsatz kommen (das traditionelle Rechtesystem ist einiges einfacher als ACLs, und in 99% der Fälle ausreichend).

### 3.1 Allgemeines

Unter Linux gibt es momentan drei Kategorien von Filesystemen:

Die Ausgereiften:

- ext2  
Schon ein paar Jahre auf dem Buckel, benötigt bei unsachgemässen umount von einem 120GB Volume durchaus 20–30 Minuten für den fsck. Stabiler, ausgereifter fsck, der fast nie versagt.

Die Verbesserten:

- ext3  
ext3 ist, wie der Name impliziert, ein verbessertes ext2. Es unterstützt Journaling, was bei Stromausfall einen fsck spart, und wird in Linux 2.6 auch hashed directory lookups haben.

Die Neuen:

- reiser  
B+Tree Lookup, Tail-Merging, Journalling etc. Viele Features, sehr schnell auch bei vielen kleinen Dateien. Sehr anfällig auf einzelne, defekte Sektoren, fsck nicht ausgereift, viele Leute berichten von Datenausfall.
- xfs  
Port von Irix nach Linux. Ähnliche Features wie reiserfs, Bedarf ebenfalls noch einiges an Arbeit.
- jfs  
Port von AIX nach Linux. Selbiges wie xfs.

Im Normalfall wird man ext3 oder ext2 einsetzen.

### 3.2 Der Aufbau

EVERYTHING IS A FILE.

Dieser Grundsatz trifft auch auf Verzeichnisse zu. Ein Verzeichnis ist lediglich eine Datei, welche Verweise auf alle Dateien enthält, welche in ihm sind. In älteren Unices kann man sich mittels `cat dir` sogar den Inhalt der Inode anschauen. In neueren \*nix gibt das jedoch einen EISDIR.

### 3.3 Inode

Eine Datei wird eindeutig gekennzeichnet durch ihre Inode, die pro Filesystem eindeutig ist. In dieser Inode sind sämtliche Metadaten wie Rechte, Timestamps, Typ etc. abgelegt, und natürlich auch der Inhalt der Datei selbst. Eine Datei erhält einen Namen, indem man von einem Directory aus auf eine Inode verweist (Hardlink). Selbstverständlich können mehrere Verweise auf die selbe Inode existieren. Im Normalfall hat man als Admin nichts mit der Inode zu tun.

### 3.3.1 Hardlinks

Mittels dem Befehl `ln` können zusätzliche Hardlinks angelegt werden. Ein Beispiel:

```
lb@teletha:test % echo file > file
lb@teletha:test % ls -i
 537601 file
lb@teletha:test % echo file2 > file2
lb@teletha:test % ln file file3
lb@teletha:test % ls -i
 537601 file  537602 file2  537601 file3
```

Die Zahl ist hier die Inode eines Files. Beachten sie, das die Rechte für eine Datei in ihrer Inode, und nicht im Directory gespeichert wird:

```
lb@teletha:test % ls -li file file3
537601 -rw-r--r--    2 lb      lb      4 Jun  2 14:26 file
537601 -rw-r--r--    2 lb      lb      4 Jun  2 14:26 file3

lb@teletha:test % chmod 600 file
lb@teletha:test % ls -li file file3
537601 -rw-----    2 lb      lb      4 Jun  2 14:26 file
537601 -rw-----    2 lb      lb      4 Jun  2 14:26 file3
lb@teletha:test % cat file
file
lb@teletha:test % cat file3
file
lb@teletha:test % echo file3 > file3
lb@teletha:test % cat file3
file3
lb@teletha:test % cat file
file3
lb@teletha:test % echo file > file3
```

Achtung: Da Hardlinks auf Inodes basieren, koennen sie nur auf dem selben Filesystem angelegt werden:

```
lb@teletha:test % ln file /tmp/file
ln: creating hard link '/tmp/file' to 'file': Invalid cross-device link
```

Als zusätzliche Einschränkung können Hardlinks nur auf Dateien angelegt werden, nicht aber auf Verzeichnisse.

```
lb@may:test % mkdir foo
lb@may:test % ln foo bar
ln: 'foo': hard link not allowed for directory
```

### 3.3.2 Softlinks

Softlinks arbeiten nicht mit Inodes, sondern nach Namen. Sie werden ebenfalls mit dem Befehl `ln` erzeugt. Sie sind Filesystem-Uebergreifend:

```
lb@teletha:test % ln -s file /tmp/file
lb@teletha:test % ls -l /tmp/file
lrwxrwxrwx    1 lb          lb          4 Jun  2 14:30 /tmp/file -> file
```

Aber Achtung:

```
lb@teletha:test % cat /tmp/file
cat: /tmp/file: Too many levels of symbolic links
```

Dies ist ein häufiger Fehler! (Ich spreche da aus eigener Erfahrung) Symlinks (andere Name für Softlinks) arbeiten anders als Inodes, nach Namen. Der Pfad muss also relativ zur Position des Symlinks, und nicht relativ zur momentanen Position angegeben werden:

```
lb@teletha:test % ln -s ../home/lb/test/file /tmp/file
lb@teletha:test % cat /tmp/file
Hallo, ich bin file
lb@teletha:test % ls -al /tmp/file
lrwxrwxrwx    1 lb          lb          18 Jun  2 14:37 /tmp/file -> ../[../]test/file
```

Andere Möglichkeit:

```
lb@teletha:test % ln -s /home/lb/test/file /tmp/file
lb@teletha:test % cat /tmp/file
file
lb@teletha:test % ls -al /tmp/file
lrwxrwxrwx    1 lb          lb          18 Jun  2 14:37 /tmp/file -> /[../]test/file
```

Symlinks werden einiges öfter eingesetzt als Hardlinks.

Symlinks können auch auf Directories gemacht werden:

```
lb@may:test % mkdir foo
lb@may:test % ln -s foo bar
lb@may:test % cd bar
lb@may:bar %
```

### 3.4 Hierarchie

/	Root, ein kleines Filesystem auf dem sich nur das wichtigste befindet
/boot	Kernel und Bootloader befinden sich hier. Meist ein eigenes Filesystem
/bin	Wichtige Binaries, die gebraucht werden, wenn /usr noch nicht da ist
/dev	Device Files, oft ein Pseudo-Filesystem (devfs)
/etc	Konfigurationsdateien
/lib	Wichtige Libraries
/mnt	Temporärer Mountpoint z.B. für Wechseldatenträger
/proc	Pseudo-Filesystem das Informationen über Prozesse und Kernel enthält
/root	Home-Directory von root
/usr	Hier befinden sich sämtliche Programme, die während dem Betrieb verwendet werden
/tmp	Temporaere Dateien (meist Pseudo-Filesystem)
/sbin	System Binaries, die während des Bootvorganges gebraucht werden
/usr/bin	Binaries, die während des Bootvorganges nicht benötigt werden
/usr/sbin	System Binaries, die während des Bootvorganges nicht benötigt werden
/usr/share/man	Manpages
/usr/include	Include-Dateien (Compiler)
/usr/lib	Libraries, die während des Bootvorganges nicht benötigt werden
/var/log	Logdateien
/var/spool	Spools, Queues, Warteschlange
/var/cache	Diverse Caches
/var/qmail	Installationsverzeichnis von qmail (einem MTA)
/package	Ein alternatives Package Managment

### 3.5 mount

Das aktivieren von Filesystemen, und das attachen<sup>3</sup> an den Verzeichnisbaum nennt man *mounten*. Hierzu dient der Befehl `mount`. Ein Aufruf von `mount` ohne Parameter veranlasst `mount` dazu, alle momentan gemounteten Filesysteme anzuzeigen.

```
lb@may:lb % mount
/dev/md/0 on / type ext3 (rw)
/dev/md/1 on /usr type ext3 (rw)
/dev/md/2 on /var type ext3 (rw)
/dev/md/3 on /home type ext3 (rw)
/dev/md/4 on /var/data/1 type ext3 (rw)
tmpfs on /tmp type tmpfs (rw)
tmpfs on /dev/shm type tmpfs (rw)
```

`mount` holt seine Informationen über Filesysteme aus der `/etc/fstab`. Diese sieht ungefähr so aus:

# Device	Mountpoint	Filesystem	Options	dump	fsck
/dev/md/0	/	ext3	defaults	0	2
/dev/md/1	/usr	ext3	defaults	0	2
/dev/md/2	/var	ext3	defaults	0	2
/dev/md/3	/home	ext3	defaults	0	2

Diese Filesysteme werden beim booten automatisch gemountet. Bei Wechseldatenträgern ist dieses Vorgehen jedoch unpraktisch, weswegen das hier deaktiviert wird:

```
/dev/floppy/0 /floppy auto user,noauto,unhide 0 0
```

Die Option `noauto` bewirkt, dass das Filesystem nicht automatisch beim booten gemountet wird, während die Option `user` bewirkt, dass das Filesystem auch von Usern gemountet werden kann.

```
lb@naru:klog % sudo mount /dev/md/0 /scratch -t ext3
lb@naru:klog % mount | grep /scratch
/dev/md/0 on /scratch type ext3 (rw)
lb@naru:klog % sudo umount /scratch
lb@naru:klog % mount | grep /scratch
lb@naru:klog %
```

Der Befehl zum unmounten von Filesystemen heisst `umount` und nicht `unmount`. Existiert ein Eintrag in der `/etc/fstab` so kann Filesystem und Devicefile weggelassen werden (`mount /floppy`).

---

<sup>3</sup>einhängen

### 3.6 fsck

fsck steht für filesystem check. Es überprüft ein Filesystem auf eventuelle Fehler, und korrigiert diese dann. Ein fsck wird bei normalen Filesystemen nur bei einem unsauberen umount ausgeführt, bei Journaling-Filesystemen ist ein fsck nur nach Systemabsturz notwendig.

Im Normalfall reicht der fsck, der beim booten des Systems automatisch gemacht wird. Jedoch kann es vorkommen, das ein manueller fsck noetig wird.

```
lb@naru:lb % sudo fsck -f /dev/md/0
fsck 1.33 (21-Apr-2003)
e2fsck 1.33 (21-Apr-2003)
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/dev/md/0: 11/915712 files (0.0% non-contiguous), 36953/1831376 blocks
```

Dies war ein erfolgreicher fsck. Er hat keine Fehler im Filesystem festgestellt, und beendet seinen Check mit einem kleinen Rapport. Die Prozentzahlen sind im übrigen die Fragmentierung (Sämtliche \*nix Filesysteme defragmentieren sich selbst, während des Betriebes).

Hier habe ich mir ein Filesystem von Hand zerschossen, und probiere es nun zu fixen:

```
lb@naru:/ % sudo fsck -f /dev/md/0
fsck 1.33 (21-Apr-2003)
e2fsck 1.33 (21-Apr-2003)
Pass 1: Checking inodes, blocks, and sizes
Inode 264 is in use, but has dtime set. Fix<y>? yes
Inode 264 has imagic flag set. Clear<y>? yes
Inode 717 has INDEX_FL flag set but is not a directory.
Clear HTree index<y>? yes
Inode 664 has compression flag set on filesystem without compression support.
Clear<y>? yes
Inode 664, i_blocks is 1238922751, should be 0. Fix<y>? yes
Inode 312 has illegal block(s). Clear<y>? yes
Illegal block #0 (3757866759) in inode 312. CLEARED.
Illegal block #1 (2005194246) in inode 312. CLEARED.
Too many illegal blocks in inode 312.
Clear inode<y>?
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
Free inodes count wrong for group #0 (217, counted=253).
Fix? yes
Free inodes count wrong (837576, counted=837612).
Fix? yes
/dev/md/0: ***** FILE SYSTEM WAS MODIFIED *****
/dev/md/0: 78100/915712 files (0.0% non-contiguous), 29048/1831376 blocks
```

## 4 Das Rechtssystem

Die Rechte einer Datei werden normalerweise durch drei Attribute bestimmt:

- Owner
- Group
- Mode

### 4.1 Owner

Der Owner einer Datei wird über dessen UID<sup>4</sup> bestimmt, welche über */etc/passwd* einem Benutzernamen zugewiesen wird. Gehört ein File der UID 1000, und ist in der */etc/passwd* ein User 'lb' mit UID 1000 definiert, so wird `ls lb` als Owner anzeigen, ansonsten die UID. Ändert man nun in der */etc/passwd* die UID von lb, so gehört das File immer noch der UID 1000. Der Owner einer Datei, und root sind die einzigen, welche die Rechte auf einer Datei ändern koennen. Lediglich root kann den Owner ändern. Der Owner kann jedoch die Group ändern.

### 4.2 Group

Die Group einer Datei wird über deren GID<sup>5</sup> bestimmt, welche über */etc/passwd* einem Gruppennamen zugewiesen wird. Ein File mit der GID 1000, kann von allen Usern welche die GID 1000 haben, bearbeitet werden. `ls` macht auch hier einen Name Lookup, um die GID einem Gruppennamen zuordnen zu können. Ein User kann Mitglied in mehreren Gruppen sein, und mehrere User können Mitglied in einer Gruppe sein. Ein User hat eine primäre GID (wird beim erstellen einer Datei verwendet), und sekundäre GIDs. Er kann auf Dateien zugreifen, welcher irgendeiner dieser Gruppen gehören.

Unsere eigenen Gruppen können wir so anzeigen:

```
lb@teletha:lb % id
uid=1000(lb) gid=1000(lb) groups=1000(lb) 900(audio) 800(lb.irssi)
801(lb.gift) 802(lb.mutt) 803(lb.web) 804(lb.lina)
```

Wie man sieht hat unser User 'lb' die UID 1000, die primäre GID 1000 und die sekundären GIDs 900, 800, 801, 802, 803 und 804.

---

<sup>4</sup>User Identification

<sup>5</sup>Group Identification

### 4.3 Mode

Der Mode ist das Komplizierteste, was man in der Unix Rechteverwaltung jemals finden wird. Gehen wir zuerst auf die letzten 3 Oktette ein, welche alle gleich aufgebaut sind:

```
-rw-r--r--    2 lb        lb        20 Jun  2 14:36 file
```

Hier sehen wir ein normales File. Der erste Abschnitt stellt seinen Mode in human-readable Form dar.

- - Das Erste der 4 Oktette (wird später betrachtet)
- rw- Das Owner-Oktett
- r-- Das Group-Oktett
- r-- Das World/Other/Any-Oktett

r steht für read, w für write, und x für execute. Soweit, so logisch. Ein anderes File:

```
lb@teletha:lb % ls -l /bin/cat
-rwxr-xr-x    1 root      root 14284 Dec  9 03:14 /bin/cat
```

Ebenfalls logisch, oder? Jeder darf cat anschauen, oder ausführen. Dies war die human readable Form, es gibt nun noch eine machine readable, die etwas komplexer ist:

- 1 Execute
- 2 Write
- 4 Read

Durch Zusammenzählen erhalten wir nun die Zahl, welche die aktuellen Rechte beschreibt: Unser erstes Beispiel sieht in dieser Schreibweise so aus:

644

Unser zweites Beispiel so:

755

### 4.4 Ändern des Mode

Man kann also fein einstellen, wer wie auf eine Datei zugreifen darf. Geändert werden kann das mittels des Befehles chmod:

```
lb@teletha:lb % ls -al file
-rw-r--r--    1 lb        lb  14123 May 29 12:23 file
lb@teletha:lb % chmod 755 file
lb@teletha:lb % ls -al file
-rwxr-xr-x    1 lb        lb  14123 May 29 12:23 file
lb@teletha:lb % chmod 000 file
lb@teletha:lb % ls -al file
-----      1 lb        lb  14123 May 29 12:23 file
lb@teletha:lb % chmod 600 file
lb@teletha:lb % ls -al file
-rw-----    1 lb        lb  14123 May 29 12:23 file
lb@teletha:lb % chmod 006 file
lb@teletha:lb % ls -al file
-----rw-    1 lb        lb  14123 May 29 12:23 file
```

Soweit so logisch? Nun, betrachten wir noch das erste Oktett:

```
lb@teletha:lb % chmod 4755 file
lb@teletha:lb % ls -al file
-rwsr-xr-x    1 lb          lb   14123 May 29 12:23 file
```

Dies ist das Setuid Bit. Damit wird die Datei immer als User lb gestartet. Egal wer dieses File ausführt. Achtung, hiermit kann man sich sehr gut in den Fuss schiessen. Häufig wird von sogenannten 'setuid-root' Programmen gesprochen. Ein Beispiel hierfür dürfte 'sudo' sein:

```
lb@teletha:lb % ls -al /sw/bin/sudo
---s---x--x   1 root      root 86128 May  5 07:09 /sw/bin/sudo
```

sudo wird immer automatisch als root gestartet. Ein Sicherheitsrisiko? Ja. Programme, die setuid root sind, müssen extrem sorgfältig geschrieben werden. Der Vorteil ist, das auf diese Art Kontrollsysteme von dem Kernel in den Userspace ausgelagert werden koennen, und dort leichter zu verwalten sind.

Im ersten Oktett können neben dem Setuid Bit noch andere Bits gesetzt werden. Dies lassen wir erstmal aussen vor, da dies nicht allzu häufig verwendet wird.

## 4.5 Ändern von Owner und Group

### 4.5.1 chown

Mittels chown kann der Owner einer Datei geändert werden:

```
lb@teletha:test % chown postfix file
chown: changing ownership of `file': Operation not permitted
```

War ja zu erwarten.

```
lb@teletha:test % sudo chown postfix file
Password:
lb@teletha:test % ls -al file
-rw-----   2 postfix lb      20 Jun  2 14:36 file
```

Jetzt gehts.

### 4.5.2 chgrp

Mittels chgrp kann die Group einer Datei geändert werden:

```
lb@teletha:test % ls -al file
-rw-----   2 lb          lb      20 Jun  2 14:36 file
\begin{verbatim}
lb@teletha:test % chgrp postfix file
chgrp: changing group of `file': Operation not permitted
```

Hmm, aber als Owner darf ich doch die Group ändern?

```
lb@teletha:test % id
uid=1000(lb) gid=1000(lb) groups=1000(lb) 900(audio) 800(lb.irssi)
801(lb.gift) 802(lb.mutt) 803(lb.web) 804(lb.lina)
```

Hmm, ich bin nicht Mitglied in der Gruppe postfix. Probieren wir mal 'audio'.

```
lb@teletha:test % chgrp audio file
lb@teletha:test % ls -al file
-rw-----  2 lb          audio    20 Jun  2 14:36 file
```

Tut.

```
lb@teletha:test % sudo chgrp postfix file
lb@teletha:test % ls -al file
-rw-----  2 postfix  postfix  20 Jun  2 14:36 file
```

Na also.

### 4.5.3 User und Group gleichzeitig

Will man Group und Owner gleichzeitig ändern, müssen die durch seinen '.' voneinander getrennt werden:

```
lb@teletha:test % ls -al file
-rw-----  2 lb          lb          20 Jun  2 14:36 file
lb@teletha:test % sudo chown postfix.postfix file
lb@teletha:test % ls -al file
-rw-----  2 postfix  postfix  20 Jun  2 14:36 file
```

Auch der : kann als Trennzeichen verwendet werden.

Anstelle von Namen kann man chown/chgrp auch direkt eine UID/GID angeben:

```
lb@teletha:test % sudo chown 3483.4834 file
Password:
lb@teletha:test % ls -al file
-rw-----  2 3483      4834      20 Jun  2 14:36 file
lb@teletha:test % grep 3483 /etc/passwd
lb@teletha:test % grep 4834 /etc/group
lb@teletha:test %
```

(gibts nicht)

```
lb@teletha:test % sudo chown 1000.1000 file
lb@teletha:test % ls -al file
-rw-----  2 lb          lb          20 Jun  2 14:36 file
```

## 5 Paketverwaltung

Debian verfügt über eine der ausgeklügeltsten Paketverwaltungen, und eine der umfangreichsten Paketsammlungen. Über 3000 Software ist bereits vorab in Pakete abgepackt. Natürlich deckt das nicht alle vorhandene Software ab, und so werden wohl auch noch ein paar H+M eigene Pakete kreiert werden müssen. (Die sich dann aber unter Kontrolle des Debian Paketmanagements befinden werden. Die Debian Paketverwaltung setzt sich aus drei Stücken zusammen:

- `aptitude`  
GUI<sup>6</sup> und CLI<sup>7</sup> zur Installation und Verwaltung von Paketen
- `apt`  
Rudimentäres CLI zur Installation und Verwaltung, Backend von `aptitude`.
- `dpkg`  
Backend der Debian Paketverwaltung. Verwaltet sämtliche Packages etc. Muss nur selten von Hand verwendet werden.

### 5.1 `aptitude`

`aptitude` ist das favoritierte Frontend zur Verwaltung eines Debian-Systems. Es verfügt sowohl über ein komfortables GUI als auch ein sehr komfortables CLI. Wird `aptitude` ohne irgendwelche Parameter aufgerufen, so wird es im GUI-Modus gestartet. Mittels diesem lassen sich Pakete updaten, installieren, verwalten, etc. Die selbe Funktionalität bietet das CLI. Wichtigste Befehle:

```
lb@nene:lb % aptitude search mail | head -5
p  aeromail          - Web-based e-mail client.
p  archivemail       - archive and compress your old email
p  asmail            - AfterStep mail monitor
p  bbmail            - Mail Utility for X
p  cgiemail          - CGI Form-to-Mail converter
lb@nene:lb % aptitude install bbmail
Reading Package Lists... Done
Building Dependency Tree... Done
E: Could not open lock file /var/lib/dpkg/lock - open (13 Permission denied)
E: Unable to lock the administration directory (/var/lib/dpkg/), are you root?
lb@nene:lb % sudo aptitude install bbmail
Reading Package Lists... Done
Building Dependency Tree... Done
The following NEW packages will be automatically installed:
  blackbox
The following packages have been kept back:
  cpp-2.95 g++-2.95 gcc-2.95 libnetpbm9 libstdc++2.10-dev
  libstdc++2.10-glibc2.2 lpr netpbm samba samba-common smbclient smbfs
  tcpdump
The following NEW packages will be installed:
  bbmail blackbox
0 packages upgraded, 2 newly installed, 0 to remove and 13 not upgraded.
Need to get 268kB of archives. After unpacking 995kB will be used.
```

---

<sup>6</sup>Graphical User Interface

<sup>7</sup>Commandline Interface

Do you want to continue? [Y/n/e/d/v/action/?]  
 Do you want to continue? [Y/n/e/d/v/action/?] ?

Commands:

y: continue with the installation  
 n: abort and quit  
 d: toggle display of dependency information  
 v: toggle display of versions  
 e: enter the full visual interface

You may also specify modifications to the set of package actions.

To do so, type an action character followed by one or more package names or patterns. The action will be applied to all packages. (you may specify additional actions; each will be applied to all packages that follow it)

Actions:

+ : Install  
 - : Remove  
 \_ : Purge  
 = : Hold

Do you want to continue? [Y/n/e/d/v/action/?] v

Versions will be shown.

The following NEW packages will be automatically installed:

blackbox [0.62.1-1] (D: bbmail)

The following packages have been kept back:

cpp-2.95 [1:2.95.4-11 -> 1:2.95.4-11woody1]  
 g++-2.95 [1:2.95.4-11 -> 1:2.95.4-11woody1]  
 gcc-2.95 [1:2.95.4-11 -> 1:2.95.4-11woody1]  
 libnetpbm9 [2:9.20-8 -> 2:9.20-8.2]  
 libstdc++2.10-dev [1:2.95.4-11 -> 1:2.95.4-11woody1]  
 libstdc++2.10-glibc2.2 [1:2.95.4-11 -> 1:2.95.4-11woody1]  
 lpr [1:2000.05.07-4.2 -> 1:2000.05.07-4.3]  
 netpbm [2:9.20-8 -> 2:9.20-8.2] samba [2.2.3a-12 -> 2.2.3a-12.3]  
 samba-common [2.2.3a-12 -> 2.2.3a-12.3]  
 smbclient [2.2.3a-12 -> 2.2.3a-12.3] smbfs [2.2.3a-12 -> 2.2.3a-12.3]  
 tcpdump [3.6.2-2.2 -> 3.6.2-2.4]

The following NEW packages will be installed:

bbmail [0.8.2-4] blackbox [0.62.1-1]

0 packages upgraded, 2 newly installed, 0 to remove and 13 not upgraded.

Need to get 268kB of archives. After unpacking 995kB will be used.

Do you want to continue? [Y/n/e/d/v/action/?]

aptitude benutzt *apt* als Backend zum Herunterladen von Paketen. Es wird also über */etc/apt* konfiguriert.

## 5.2 apt

*apt* ist das Fetch-System für Debian-Packages. Es wird über */etc/apt* konfiguriert. Dort befinden sich zwei Files, die von essentieller Bedeutung sind:

### 5.2.1 */etc/apt/sources.list*

```
deb http://debian.ethz.ch/mirror/debian woody main
deb-src http://debian.ethz.ch/mirror/debian woody main
deb http://non-us.debian.org/debian-non-US woody/non-US main
deb-src http://non-us.debian.org/debian-non-US woody/non-US main
deb http://security.debian.org/ stable/updates main
```

Hier sind die Quellen angegeben, von denen *apt* Pakete heruntergeladen wird. Hier wird noch eine weitere Zeile eingefügt werden, für das H+M spezifische Package-Repository.

### 5.2.2 */etc/apt/apt.conf*

```
APT
{
Acquire
{
    http
    {
        Proxy "http://proxy.v-1.ch:80";
    };
};
```

Dies ist das Konfigurationsfile von *apt*. Es enthält diverse Einstellungen, die *apt* betreffen. Meistens muss hier nur der korrekte Proxy-Server eingestellt werden.

### 5.2.3 *apt-get*

*apt-get* ist eines der Tools, die Debian ziemlich 'berühmt' gemacht haben. *apt-get* ist zwar sehr leicht zu bedienen, gehört aber absolut nicht in die Hände von Anfängern. Für diese empfiehlt sich die Verwendung von *aptitude*. (Ich persönlich verwende ebenfalls *aptitude*)

### 5.2.4 *apt-cache*

*apt-cache* kann verwendet werden, um den Paket-Cache von *apt* nach Stichwörtern zu durchsuchen.

### 5.3 dpkg

dpkg ist das Backend der Debian-Paketverwaltung. Sämtliche Pakete werden mittels dpkg installiert. Auch wer mit aptitude arbeitet, bekommt seine Pakete letztendlich von dpkg installiert. Nur geschieht dies im Hintergrund. Falls man ein einzelnes *.deb* File hat, kommt man zur Installation nicht um dpkg herum.

```
lb@nene:archives % sudo dpkg -i wget_1.8.2-5_i386.deb
Selecting previously deselected package wget.
(Reading database ... 23091 files and directories currently installed.)
Unpacking wget (from wget_1.8.2-5_i386.deb) ...
Setting up wget (1.8.2-5) ...
```

```
lb@nene:archives % sudo dpkg -r wget
(Reading database ... 23137 files and directories currently installed.)
Removing wget ...
```

```
lb@nene:archives % dpkg -l | egrep '(wget|zsh)'
```

rc	wget	1.8.2-5	retrieves files from the web
ii	zsh	4.0.6-13	A shell with lots of features.

dpkg -l listet alle Pakete auf, die die Paketverwaltung kennt. Dazu gehören auch Pakete, welche deinstalliert wurden. Beim deinstallieren eines Paketes werden die Konfigurationsdateien nämlich nicht gelöscht. Will man das dies geschieht, so muss man nachhelfen:

```
lb@nene:archives % sudo dpkg --purge wget
(Reading database ... 23091 files and directories currently installed.)
Removing wget ...
Purging configuration files for wget ...
```

Mit dpkg lassen sich Fehler im Paketmanagement reparieren, wenn aptitude versagt. Die Manpages und die Dokumentation sind sehr ausführlich, und deren Studium ist ausdrücklich angeraten.

## 6 Userverwaltung

Die Userverwaltung eines Unix-Systems ist sehr einfach gehalten. User sind in */etc/passwd* definiert, Gruppen in */etc/group* und User-Passwörter in */etc/shadow*, */etc/gshadow*. Mit Ausnahme von */etc/shadow*, */etc/gshadow* sind alle Files world-readable. Der Seperator hier ist immer ein Doppelpunkt.

### 6.1 */etc/passwd*

```
lb:x:1000:1000:Lukas Beeler:/home/lb:/bin/zsh
```

lb	x	1000	1000	Lukas Beeler	/home/lb	/bin/zsh
Username	Passwort	UID	GID	GECOS (Name)	Home-Directory	Loginshell

Das 'x' im Passwort-Feld zeigt an, dass das Passwort in der */etc/shadow* gespeichert ist. Ein '\*' würde angeben, dass der User kein Passwort hat, und sich demnach nicht einloggen kann.

### 6.2 */etc/group*

```
lb:x:1000:
audio:x:900:lb,root
```

lb	x	1000	
audio	x	900	lb,root
Group	Passwort	GID	Mitglieder

Das 'x' im Passwort-Feld zeigt an, dass das Passwort in der */etc/gshadow* gespeichert ist.

### 6.3 */etc/shadow*

```
lb:$1$SHi/Z7dE$iKqocK4ygdjfs9NU/7JLY0:12089::::::
```

lb	\$1\$SHi/Z7dE\$iKqocK4ygdjfs9NU/7JLY0	12089 ..
User	Echtes Passwort (MD5 Hash)	Expiry-Informationen

### 6.4 Bearbeitung

Diese Files werden normalerweise von Hand (*vi*/*vim*) editiert, da sie sehr leicht zu bearbeiten sind. Für die Batchverarbeitung empfehlen sich jedoch die Tools *useradd* und *groupadd*. Passwörter können mittels *passwd* geändert werden.

## 7 Verwaltung von Netzwerkkinterfaces

Netzwerkfunktionalität ist schon seit ewig Bestandteil diverser \*nix Betriebssysteme. Traditionell werden zur Konfiguration von Interfaces die Befehle `ifconfig` und `route` verwendet. Seit Linux 2.4 sind diese deprecated<sup>8</sup>, und wurden durch das Paket `iproute2` ersetzt. In `iproute2` befinden sich die Befehle `ip` und `tc`. Die Funktionalität von `tc` findet sich lediglich in `iproute2`, und nicht in `netkit-base`.

### 7.1 ip

`ip` ist ein umfassendes Tool zur Netzwerkkonfiguration. Es unterstützt IPv6 und IPv4. Routing, IP-Adressen, Netzwerke etc. lassen sich alle mittels `ip` konfigurieren.

#### 7.1.1 lo konfigurieren

```
lb@naru:lb $ sudo ip addr add 127.0.0.1/8 dev lo
lb@naru:lb $ sudo ip link set lo up
```

Dies gibt dem `lo`<sup>9</sup> Interface die IP Adresse 127.0.0.1 mit einer CIDR-Netmask von /8. Damit ist die Konfiguration von `lo` bereits abgeschlossen.

Achtung: Diese Einstellungen werden nirgendwo gespeichert. Solche zur Runtime gemachten Einstellungen werden nach einem Reboot wieder verworfen.

#### 7.1.2 Interfaces anzeigen

```
lb@naru:lb $ sudo ip addr sh
1: lo: <LOOPBACK,UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
2: sit0@NONE: <NOARP> mtu 1480 qdisc noop
    link/sit 0.0.0.0 brd 0.0.0.0
3: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 100
    link/ether 00:30:4f:10:cc:0f brd ff:ff:ff:ff:ff:ff
    inet 10.10.2.1/16 scope global eth0
    inet6 3ffe:202c:ffff:30:230:4fff:fe10:cc0f/64 scope global
    inet6 fe80::230:4fff:fe10:cc0f/64 scope link
4: tap0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 100
    link/ether 00:ff:de:19:d2:40 brd ff:ff:ff:ff:ff:ff
    inet 10.10.2.11/32 brd 10.10.2.11 scope global tap0
    inet6 fe80::2ff:deff:fe19:d240/64 scope link
```

---

<sup>8</sup>veraltet, obsolet

<sup>9</sup>Loopback

### 7.1.3 Routen erstellen

```
lb@naru:lb $ sudo ip ro add 10.10.6.0/24 via 10.10.0.1
lb@naru:lb $ sudo ip ro add 10.10.1.0/24 via 10.10.0.1
```

Dies erstellt zwei Routen über den Gateway 10.10.0.1 für die Netzwerke 10.10.6.0/24 und 10.10.1.0/24.

### 7.1.4 Routen anzeigen

```
lb@naru:lb % ip ro sh
10.10.2.10 dev tap0 scope link
10.10.6.0/24 via 10.10.0.1 dev eth0
10.10.1.0/24 via 10.10.0.1 dev eth0
10.10.0.0/16 dev eth0 proto kernel scope link src 10.10.2.1
default via 10.10.0.1 dev eth0
```

### 7.1.5 Hilfe anzeigen

```
lb@teletha:lb % ip ro help
Usage: ip route { list | flush } SELECTOR
       ip route get ADDRESS [ from ADDRESS iif STRING ]
                          [ oif STRING ] [ tos TOS ]
       ip route { add | del | change | append | replace | monitor } ROUTE
SELECTOR := [ root PREFIX ] [ match PREFIX ] [ exact PREFIX ]
           [ table TABLE_ID ] [ proto RTPROTO ]
           [ type TYPE ] [ scope SCOPE ]
ROUTE := NODE_SPEC [ INFO_SPEC ]
NODE_SPEC := [ TYPE ] PREFIX [ tos TOS ]
            [ table TABLE_ID ] [ proto RTPROTO ]
            [ scope SCOPE ] [ metric METRIC ]
INFO_SPEC := NH OPTIONS FLAGS [ nexthop NH ]...
NH := [ via ADDRESS ] [ dev STRING ] [ weight NUMBER ] NHFLAGS
OPTIONS := FLAGS [ mtu NUMBER ] [ advmss NUMBER ]
           [ rtt NUMBER ] [ rttvar NUMBER ]
           [ window NUMBER ] [ cwnd NUMBER ] [ ssthresh REALM ]
           [ realms REALM ]
TYPE := [ unicast | local | broadcast | multicast | throw |
         unreachable | prohibit | blackhole | nat ]
TABLE_ID := [ local | main | default | all | NUMBER ]
SCOPE := [ host | link | global | NUMBER ]
FLAGS := [ equalize ]
NHFLAGS := [ onlink | pervasive ]
RTPROTO := [ kernel | boot | static | NUMBER ]
```

## 7.2 tc

Mittels `tc` lässt sich Traffic Shaping/Policing konfigurieren. Eine detaillierte Beschreibung dieses Tools würde den Rahmen dieses Textes sprengen, deshalb sei hier auf <http://lartc.org> verwiesen.

### 7.3 /etc/network/interfaces

Dieses Debian-spezifische Konfigurationsfile enthält sämtliche Einstellungen von Netzwerkinterfaces, welche beim booten aktiviert werden. Ein Beispiel für dieses File:

```
# The loopback interface
auto lo
iface lo inet loopback
# The first network card
auto eth0
iface eth0 inet static
address 160.1.3.240
netmask 255.255.0.0
gateway 160.1.2.10
```

Ziemlich selbsterklärend, oder? Anstelle von 'static' kann man auch 'dhcp' angeben. eth0<sup>10</sup> ist das Ethernet-Netzwerkinterface dessen Treiber als erstes geladen wurde, und nicht dies, welches im ersten Slot steckt.

#### 7.3.1 ifup

`ifup` ist ein Debian-spezifischer Befehl, mit welchem ein Interface nach der Beschreibung in */etc/network/interfaces* aktiviert werden kann. Ein Beispiel:

```
ifup eth0
```

#### 7.3.2 ifdown

`ifdown` ist ein Debian-spezifischer Befehl, mit welchem ein Interface deaktiviert werden kann. Hierbei kommen wieder die Einstellungen in */etc/network/interfaces* zum tragen. Ein Beispiel:

```
ifdown eth0
```

---

<sup>10</sup>Ethernet Adapter #0

## 7.4 iptables

iptables ist das Userland-Tool um den im Kernel integrierten Paketfilter zu konfigurieren. Der Syntax von iptables ist ziemlich eingängig, und leicht zu erlernen. Grundlegende Ahnung von IP, TCP und UDP wird jedoch vorausgesetzt. Ein Hands-On Approach ist iptables dürfte ein minimales Einstellungsgerüst sein.

```
# default policy auf DROP (verwerfen)
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

# traffic auf lo erlauben
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

# versenden von paketen immer gestatten
iptables -A OUTPUT -o eth0 -j ACCEPT

# Connection Tracking zum erlauben von eingehenden Verbindungen
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

# Eingehende verbindungen fuer http/smtp erlauben
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -A INPUT -p tcp --dport 25 -j ACCEPT

# Eingehenden Verbindung auf ssh nur vom LAN her erlauben
iptables -A INPUT -i eth1 -p tcp --dport 22 -j ACCEPT
```

Simple, oder? Um den Kernel während dem Bootvorgang korrekt zu konfigurieren wird meist ein Shellscript geschrieben, das während jedem Bootvorgang aufgerufen wird.

iptables ist eine ziemlich komplexe Materie, und erlaubt weitaus mehr Möglichkeiten als die hier demonstrierten.

Übung: Erarbeiten sie zuerst einmal theoretisch ein Paketfilter-Ruleset.

Setzen sie dies nachher mittels iptables um. Am besten auf einem Blatt Papier, oder in einem Editor.

## 8 Verwaltung von Software Raids

Software-Raids werden mittels des Tools mdadm verwaltet.

### 8.1 Einrichten der Partitionen

Erstellen eines einfache Raids, bestehend aus zwei Disks: Hierzu werden die benötigten Partitionen erstellt, und den Partitionen der korrekte Typ zugewiesen, damit die RAIDs beim starten des Kernels automatisch erkannt werden können.

```
lb@naru:lb % cat /proc/mdstat
Personalities : [raid1]
read_ahead not set
unused devices: <none>
lb@naru:lb % sudo fdisk /dev/hda
```

The number of cylinders for this disk is set to 3736.  
There is nothing wrong with that, but this is larger than 1024,  
and could in certain setups cause problems with:

- 1) software that runs at boot time (e.g., old versions of LILO)
- 2) booting and partitioning software from other OSs  
(e.g., DOS FDISK, OS/2 FDISK)

Command (m for help): p

```
Disk /dev/hda: 30.7 GB, 30735581184 bytes
255 heads, 63 sectors/track, 3736 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
```

Device	Boot	Start	End	Blocks	Id	System
/dev/hda1		1	13	104391	83	Linux
/dev/hda2		14	144	1052257+	82	Linux swap
/dev/hda3		145	1968	14651280	5	Extended
/dev/hda4		1969	3736	14201460	83	Linux
/dev/hda5		145	1056	7325608+	83	Linux
/dev/hda6		1057	1968	7325608+	83	Linux

```
Command (m for help): t
Partition number (1-6): 5
Hex code (type L to list codes): FD
Changed system type of partition 5 to fd (Linux raid autodetect)
Command (m for help): t
Partition number (1-6): 6
Hex code (type L to list codes): FD
Changed system type of partition 6 to fd (Linux raid autodetect)
Command (m for help): w
The partition table has been altered!
Calling ioctl() to re-read partition table.
Syncing disks.
```

## 8.2 Erstellen des RAIDs

Wir haben nun zwei Partitionen auf den typ 'Linux raid autodetect' umgestellt, damit der Kernel beim booten das Raidset wieder erkennen kann. Hier befinden sich beide Teile des RAID1 auf der selben Festplatte. Das ist natürlich sehr schlecht, und wird in der Praxis nicht so gemacht.

```
lb@naru:lb % sudo mdadm --create /dev/md/0 -l 1 -n 2 /dev/hda5 /dev/hda6
mdadm: /dev/hda5 appears to contain an ext2fs file system
       size=7325608K  mtime=Sun May 18 13:16:04 2003
Continue creating array? y
mdadm: array /dev/md/0 started.
lb@naru:lb % cat /proc/mdstat
Personalities : [raid1]
read_ahead 1024 sectors
md0 : active raid1 ide/host0/bus0/target0/lun0/part6[1]
       ide/host0/bus0/target0/lun0/part5[0]
       7325504 blocks [2/2] [UU]
       [>.....] resync = 0.4% (36288/7325504)
       finish=50.2min speed=2419K/sec
unused devices: <none>
```

## 8.3 Erstellen des Dateisystemen

Das war es eigentlich schon, wir verfügen nun über ein RAID1 bestehend aus zwei Partitionen. Als erstes müssen wir auf unserem RAID ein Filesystem erstellen.

```
lb@naru:lb % sudo mkfs.ext3 /dev/md/0
mke2fs 1.33 (21-Apr-2003)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
915712 inodes, 1831376 blocks
91568 blocks (5.00%) reserved for the super user
First data block=0
56 block groups
32768 blocks per group, 32768 fragments per group
16352 inodes per group
Superblock backups stored on blocks:
       32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done
```

This filesystem will be automatically checked every 32 mounts or 180 days, whichever comes first. Use tune2fs -c or -i to override.

## 8.4 Deaktivieren einer Disk

Koppeln wir mal eine der Disks aus dem RAID1 aus, indem wir sie als defekt markieren.

```
lb@naru:lb % sudo mdadm -f /dev/md/0 /dev/hda6
mdadm: set /dev/hda6 faulty in /dev/md/0
lb@naru:lb % cat /proc/mdstat
Personalities : [raid1]
read_ahead 1024 sectors
md0 : active raid1 ide/host0/bus0/target0/lun0/part6[1](F)
      ide/host0/bus0/target0/lun0/part5[0]
      7325504 blocks [2/1] [U_]

unused devices: <none>
```

Natürlich spuckt der Kernel in diesem Fall eine Warnmeldung aus, welche wir mittels `dmesg` anschauen können.

```
raid1: Disk failure on ide/host0/bus0/target0/lun0/part6, disabling device.
      Operation continuing on 1 devices
md: recovery thread got woken up ...
md: updating md0 RAID superblock on device
md: (skipping faulty ide/host0/bus0/target0/lun0/part6 )
md: ide/host0/bus0/target0/lun0/part5 [events: 00000002]<6>(write)
      ide/host0/bus0/target0/lun0/part5's sb offset: 7325504
md: md_do_sync() got signal ... exiting
md0: no spare disk to reconstruct array! -- continuing in degraded mode
md: recovery thread finished ...
```

Für weiterführende Informationen steht man `mdadm` zur Verfügung.

## 9 Verwaltung von Diensten

Auf sämtlichen Maschinen wird das gleiche Konzept verwendet:

One-Shot Dienste mittels `sysvinit`

Long-Running Dienste mittels `runit` (H+M spezifisch)

### 9.1 sysvinit

Script in `/etc/init.d/` ablegen, mit `update-rc.d` Symlinks nach `/etc/rc2.d/` anlegen. Nach dem Einrichten des Symlinks muss der Dienst noch von Hand gestartet werden, beim Reboot wird er automatisch gestartet. Stirbt der Dienst, so wird er nicht wieder gestartet. (Für die Dienste die immer laufen sollen, unbedingt `runit` verwenden). Doku: `/etc/init.d/README`

#### 9.1.1 Starten

```
/etc/init.d/progname start
```

#### 9.1.2 Stoppen

```
/etc/init.d/progname stop
```

#### 9.1.3 Neustarten

```
/etc/init.d/progname restart
```

### 9.2 runit

`runit` ist nicht Teil einer normalen Debian-Installation. `runit` befindet sich in einem H+M spezifischen Paket.

Dienstverzeichnis mit `runfile` und `logger` in `/etc/service/progname/` anlegen. Dienst nach `/service` symlinken, dann werden sie automatisch gestartet, ebenfalls beim booten. Stirbt der Dienst, so wird er automatisch neu gestartet.

#### 9.2.1 Starten

Wird beim anlegen des Symlinks automatisch gestartet. Falls der Dienst beendet wurde, kann er mit:

```
svc -u /service/progname
```

gestartet werden:

#### 9.2.2 Stoppen

```
svc -d /service/progname
```

#### 9.2.3 Neustarten

```
svc -t /service/progname
```

Initscripts in `/etc/init.d/` sind üblicherweise in Shell geschrieben, Runfiles für `runit` sind normalerweise in `exectline` geschrieben.

## 10 Signale

Signale sind eine Form von IPC<sup>11</sup>. Sie dienen dazu, einem Prozess rudimentäre Befehle mitzuteilen, wie z.B. 'Beende dich' oder 'Halte an'.

Mittels `kill` können Signale versendet werden. Normalerweise verschickt `kill` das Signal `SIGTERM`. `TERM` veranlasst einen Prozess dazu, sich zu beenden. Es kann vorkommen, dass ein Prozess nicht auf `TERM` reagiert (Freeze). In diesem Fall kann ein `SIGKILL` geschickt werden (`kill -9, kill -KILL`), welches den Kernel dazu veranlasst, den Prozess zu entfernen. Ein anderes Signal, `SIGHUP`, (Hangup) wird bei Daemons dazu verwendet, ihre Konfiguration neu einzulesen. Bei den meisten Programmen hat `SIGHUP` jedoch die gleichen Auswirkungen wie ein `SIGTERM`.

`kill` verschickt Signale an PIDs. Er kann keine Signale an Prozessnamen senden. Mittels `ps` lassen sich die PIDs einzelner Prozesse herausfinden. Will man alle Prozesse mit gleichem Namen beenden, so wird `pkill` verwendet. `pkill` hat den gleichen Syntax wie `kill`, nur will es Name anstelle von PIDs. Das Linux `killall` hat den selben Syntax wie `pkill`, jedoch ist `killall` auf anderen Unices wie z.B. Solaris etwas anderes. Aus diesem Grund sollte immer `kill/pkill` verwendet werden, und nie `killall`.

Für Dienste die mittels `sysvinit` gestartet wurden, existieren manchmal Pidfiles in `/var/run`. Dies lässt sich in Scripts verwenden.

Für Dienste die mittels `runit` gestartet werden, lässt sich immer ein Signal an den Dienst schicken, indem man `svc` benutzt.

```
svc -t /service/progname SIGTERM
svc -k /service/progname SIGKILL
svc -h /service/progname SIGHUP
```

Es existieren noch zahlreiche andere Signale. Nachzulesen in `man 7 signal`.

---

<sup>11</sup>Inter Process Communication

## 11 Logfiles

Das verwendete Loggingsystem ist H+M spezifisch, und ist in einer normalen Debian-Installation nicht so.

Logfiles liegen unterhalb von `/var/log`. Es existieren zwei Logging-Mechanismen: `multilog` und `socklog`. `multilog` ist der neuere Logging-Mechanismus, und wird im Normalfall verwendet. Jedoch existieren noch viele Programme, welche lediglich den `syslog` Mechanismus verwenden können, diese Programme werden mittels `socklog` geloggt. Zusätzlich gibt es auch noch die Logs, welche vom Kernel stammen. Der Kernel schreibt seine Logfiles in einen Ringbuffer, welcher mit `dmesg` ausgelesen werden kann. Zusätzlich kümmert sich `socklog` nach darum, das der Inhalt dieses Ringbuffers auf die Platte geschrieben wird.

<code>/var/log/socklog/*</code>	Logfiles, die mittels des <code>syslog</code> Protokolles geschrieben wurden
<code>/var/log/klog/*</code>	Logfiles des Kernels
<code>/var/log/category/progname</code>	Logfiles von <code>/etc/service/category/progname</code>
<code>/var/log/progname</code>	Logfiles von <code>/etc/service/progname</code>

Logfiles sind die Freunde jedes Admins, da sie im allgemeinen helfen, Probleme zu debuggen. In Logfiles kann man Dinge erfahren, die man sonst nie zu Gesicht bekäme. Das kann unter Umständen auch Geschäftsgeheimnisse betreffen. Deswegen muss ein Admin Dinge, die er in Logfiles erfahren hat als Berufsgeheimnis betrachten.

### 11.1 Dateinamen

Sämtliche Logfiles sind im sogenannten `multilog` Format, auch solche die via `socklog` geschrieben wurden. Dies schaut dann so aus:

```
lb@naru:socklog % pwd
/var/log/socklog
lb@naru:socklog % sudo ls main
Password:
@400000003e3ea5b4250397f4.u @400000003eb295fa109c4fcc.u lock
@400000003e3ed87c05380194.u @400000003edbb46b0e9880ec.u state
@400000003e7395bb338a9a5c.u current
```

`current` ist das Logfile, welches momentan geschrieben wird. Die `@*` Files sind Logfiles, welche bereits rotiert wurden. Das `@*` ist ein sogenannter TAI64 Timestamp, welcher mittels des Tools `tai64nlocal` in ein menschenlesbares Format umgewandelt werden kann.

```
lb@naru:socklog % echo @400000003e3ea5b4250397f4.u |tai64nlocal
2003-02-03 18:23:54.620992500.u
```

Der Timestamp gibt an, wann das Logfile rotiert wurde.

## 11.2 Timestamps

Timestamps in Logfiles sind ebenfalls im TAI64 Format.

```
lb@naru:klog % tail -2 current
@400000003edc8ca111435d6c kern.info: raid1: mirror resync was not fully
finished, restarting next time.
@400000003edc8ca111436154 kern.info: raid1: mirror resync was not fully
finished, restarting next time.
lb@naru:klog % tail -2 current | tai64nlocal
2003-06-03 13:55:03.289627500 kern.info: raid1: mirror resync was not fully
finished, restarting next time.
2003-06-03 13:55:03.289628500 kern.info: raid1: mirror resync was not fully
finished, restarting next time.
```

## 11.3 Konfiguration, Rotation

multilog rotiert Logs nach Grösse, und nicht nach Zeit, wie es viele andere Logger tun. Auf den ersten Blick mag das schlecht erscheinen, jedoch schützt nur ein rotieren nach Grösse vor dem überfüllen des Dateisystemes durch einen böswilligen Angreifer. Schauen wir uns mal ein typisches multilog Konfigurationsfile an:

```
lb@naru:klog % cat /etc/service/openssh/log/run
#!/command/execlineb
# sshd logscript

setuidgid sshdl
multilog t /var/log/openssh
```

Wie man dieses Konfigurationsfile deutet, ist dem Schüler als Übung überlassen, nachdem er <http://cr.yip.to/daemontools/multilog.html> studiert hat.

.....

.....

.....

.....



## 13 Netiquette/Problemlösung

Auf den ersten Blick mag es seltsam erscheinen, das hier die Netiquette erklärt wird. Zwischenmenschliche Kommunikation ist vorallem im \*nix-Bereich sehr weit verbreitet, und meist die erste Anlaufstelle für Support. Doch bevor man im Usenet, IRC oder auf Mailinglisten nach Hilfe sucht, sollte man einige Punkte der Netiquette beachten.

Reihenfolge zur Problemlösung:

- Dokumentation lesen (RTFM)
- FAQ lesen
- Mit <http://www.google.com> und <http://groups.google.com> suchen
- Menschen fragen (Mailinglisten, Usenet, IRC)

Für Mailinglisten und Usenet gelten Regeln, deren Einhaltung die Akzeptanz wesentlich erhöht.

- Kompletter Realname im Header (Kein Firmenname oder Roleaccount)
- Keine HTML Mails
- Korrekte Rechtschreibung, Grammatik, Satzzeichen, Gross/Kleinschreibung
- Korrektes quoting (<http://learn.to/quote>)
- Korrekte Signatur (<http://learn.to/sign>)
- Gute, akkurate Fehlerbeschreibung, mit Copy&Paste aller Fehlermeldungen
- Korrekte Deklaration von Umlauten
- Anständiger Ton
- Kein 'Security through Obscurity', das verheimlichen von Informationen erschwert die Fehlersuche enorm. Ist sichtbar, das Informationen verfälscht wurden, versuchen die Leute meist nichtmal zu helfen

Eine gute, allgemeine FAQ zum erstellen von Usenet-Postings und E-Mails findet sich hier:

<http://www.lugs.ch/lugs/maillingliste/faq.phtml>.

Um diesen Regeln zu entsprechen, muss man einen tauglichen MUA<sup>12</sup>/NUA<sup>13</sup> haben, und dessen Konfiguration beherrschen. Für den Einstieg ins Usenet empfiehlt es sich, zuerst einmal de.newusers.\* zu lesen. Erste Postings sollte man in de.test oder ch.test absetzen, und diese peinlichst genau auf Korrektheit überprüfen.

Auf Mailinglisten und im Usenet hat man es im allgemeinen mit Hochbezahlten Experten zu tun, die Freiwillig Support leisten. Verschwendet man die Zeit solcher Leute mit falsch formatierten Mails, ungenauen Fragen, etc. ist die Wahrscheinlichkeit, das sie einem nicht helfen werden, sehr hoch.

In ein Mailinglisten oder Usenet Posting sollte man mindestens eine halbe Stunde Zeit investieren. Meistens kommt man innerhalb dieser Zeit schon selbst auf die Lösung. Falls nicht, sollte man sein Posting vor dem Absenden nochmals genau überprüfen. Postings an Mailinglisten oder im Usenet werden auf Lebenszeit archiviert. Falls man später einmal einen Job in einem professionellen Umfeld sucht (z.B. ISP) wird der Arbeitgeber erstmal anhand von Google die Kompetenz eines Anwerbers beurteilen.

---

<sup>12</sup>Mail User Agent

<sup>13</sup>News User Agent

## 14 Software bei Huber+Monsch

H+M setzt nur speziell getestete und evaluierte Software ein, die auch strengsten Sicherheitsanforderungen genügt. Um H+M eigene Software von 'normaler' Debian-Software unterscheiden zu können wird diese Software in einem anderen Prefix installiert. Dieser Prefix ist `/sw`.

Ein modifiziertes Service-Management ermöglicht eine höhere Verfügbarkeit, als dieses welches bei Debian standardmässig dabei ist. Jedoch erhöht die Existenz zweier Service-Management-Systeme die Komplexität. Ein kleiner Preis, wenn man den Gewinn an Effizienz betrachtet.

### 14.1 Service-Management

Das Service-Management setzt sich aus den vier Komponenten `ash-hm`, `daemontools`, `runit` und `etc-service` zusammen. Eine genaue Dokumentation dazu findet sich hier:

<http://cr.yip.to/daemontools.html>

<http://smarden.org/runit/>

### 14.2 HTTP/FTP Proxy

Als HTTP/FTP Proxy kommt die Software `squid` zum Einsatz. Sie verfügt über eine anständige Sicherheitshistorie, und viele intelligente Features. `squid` lässt sich im Bedarfsfalle leicht erweitern.

Dokumentation:

<http://www.squid-cache.org>

### 14.3 SMTP Proxy

Als SMTP Proxy kommt der MTA `qmail` zum Einsatz. Eine beeindruckende Sicherheitshistorie (keine Sicherheitsprobleme seit 1. Release vor 7 Jahren). Zudem sehr flexibel, leicht erweiterbar, und performant.

Dokumentation:

<http://www.lifewithqmail.org>

<http://www.qmail.org>

<http://cr.yip.to/qmail.html>

### 14.4 Paketfilter

Als Paketfilter kommt `iptables` zum Einsatz. `iptables` besteht aus mehreren Netfilter Modulen, und einem Userspace Tool, mit welchem diese Kernelmodule konfiguriert werden können.

Dokumentation:

<http://www.netfilter.org>

### 14.5 IP-IP Tunneling

Zum Tunneln von IP Verbindungen kommen zwei verschiedene Software-Pakete zur Anwendung, welche komplett voneinander unabhängig sind. `OpenVPN` wird zur Verbindung zwischen den einzelnen PF's eingesetzt. Zur Anbindung der `Roadwarrior's` kommt `FreeS/WAN` zum Einsatz.

Dokumentation:

<http://www.freeswan.org>

<http://openvpn.sf.net>

## A Tipps

- **mc**  
mc, midnight commander, ist ein Clone des altbekannten Norton Commanders. Für Leute, die schonmal mit dem nc gearbeitet haben, dürfte sich die Nutzung von mc empfehlen.
- **links**  
links ist ein textbasierter Browser. Er empfiehlt sich für schnelle Recherchen mit Google, da man in diesem Fall kein Fenster wechseln muss.
- **screen**  
screen ist ein Terminal-Manager. Er kann mehrere virtuelle Terminals in einem laufen lassen, und diese virtuellen Terminals auch detach<sup>14</sup> und wieder attach<sup>15</sup>.
- **samba**  
samba ist ein Softwarepaket, das die Protokolle SMB, NMB beherrscht (Die jedem Windows-Admin bekannt sein dürften). Mit samba kann man ebenfalls NT4 PDC spielen. Mehr zu diesem Stück Software in einem speziellen Modul.
- **bmon**  
bmon zeigt die Auslastung von Netzwerkinterfaces graphisch an. Dies kann sehr praktisch sein, um die Auslastung eines Links zu begutachten, ohne gross unsichere Software zu installieren (snmp, mrtg).
- **tcpdump**  
tcpdump ist ein packet sniffer. Er wird lediglich über Optionen gesteuert, und ist somit non-*interaktiv*. Er empfiehlt sich um Netzwerkproblem zu debuggen (Kenntnisse der darüber laufenden Protokolle vorausgesetzt).
- **ethereal**  
ethereal ist ebenfalls ein packet sniffer. Er verwendet das gleiche Backend wie tcpdump, bietet aber in Gegensatz zu diesem ein volles GUI. Auch bereitet er Protokolle graphisch einiges besser auf. Für eine umfassende Netzwerkanalyse eignet sich ethereal demnach besser. Unglücklicherweise kann man ethereal nicht auf Servern installieren, da X11 verlangt wird. Zum Glück lassen sich mit tcpdump -w Dumps schreiben, und diese dann von ethereal einlesen.

---

<sup>14</sup>vom aktiven Terminal entkoppeln

<sup>15</sup>an ein aktives Terminal ankoppeln

## B Credits

- Reto Schüttel  
Korrekturlesen, Verbesserungen
- Thomas Bader  
Korrekturlesen, Verbesserungen
- Gabi Beeler  
Korrekturlesen
- Daniel Hottinger  
Korrekturlesen, Verbesserungen, L<sup>A</sup>T<sub>E</sub>XTipps
- Attila Kinali  
Korrekturlesen, Verbesserungen
- Tobias Wyssling  
Korrekturlesen
- +kaosu

## C Legal Stuff

Dieses Dokument ist Public Domain, das heisst es kann frei verändert und verbreitet werden. Es unterliegt keinerlei Einschränkungen.