

mitp

2. Auflage

Eric Amberg

Linux-Server mit Debian GNU/Linux

Das umfassende Praxis-Handbuch

Aktuell für die Versionen Debian 5.0 (Lenny)
und Debian 4.0 (Etch)

Praxis-Szenarien: Backoffice-Server, Root-
Server, Linux als Gateway, Server-Security

Zahlreiche Workshops mit
Schritt-für-Schritt-Anleitungen

Das Debian-System – Grundlagen

Willkommen auf unserem ersten Rundgang durch Ihr neues Debian-GNU/Linux-System. In diesem Kapitel lernen Sie Folgendes:

- Grundlagen der Konsole
- Herunterfahren und Neustarten des Systems
- Basisbefehle zur Navigation durch das Dateisystem
- Aufbau des Dateisystems – Wo befindet sich was?
- Manipulation von Dateien und Verzeichnissen
- Hilfe zur Selbsthilfe – die Man-Pages und Infoseiten

Der Inhalt dieses Kapitels ist eigentlich nicht Debian-spezifisch, sondern kann fast eins zu eins auf viele andere Linux-Distributionen angewendet werden. Es handelt sich um allgemeine Linux-Grundlagen, die ihre Gültigkeit größtenteils sogar bei Unix-Systemen haben.

4.1 Die Konsole

Nach der Installation des Basissystems bietet Ihnen Debian zunächst einmal eine Konsole an. Sie müssen einen Benutzernamen und sein Passwort eingeben, um sich einzuloggen.

```
Debian GNU/Linux 5.0 debian tty1
debian login: root
Password: _
```

Während der Eingabe Ihres Passwortes sehen Sie keine Zeichen auf dem Bildschirm – das hat seine (sicherheitsbedingte) Richtigkeit. Anschließend sehen Sie den *Prompt*. Dieser zeigt Ihnen in der Voreinstellung den Rechnernamen, das aktuelle Verzeichnis und Ihren User-Status an:

```
debian: ~#
```

Dieses System hat den Rechnernamen `debian`, das aktuelle Verzeichnis ist das Home-Verzeichnis des angemeldeten Benutzers. Dies wird durch die Tilde (~) angezeigt. Nach der Anmeldung ist dieses Verzeichnis grundsätzlich Ihr Startpunkt. Das Doppelkreuz (#) zeigt an, dass Sie Superuser-Rechte haben, also als `root` angemeldet sind. Bei normalen Benutzern steht hier in der Voreinstellung das Dollarzeichen (\$).

Linux ist ein echtes Multiuser-Betriebssystem. Das bedeutet, es können mehrere Benutzer zur gleichen Zeit auf dem System arbeiten. Sie können aber auch so tun, als seien Sie selbst

zu mehreren. Das heißt, Sie können sich mehr als einmal anmelden – als derselbe Benutzer oder als ein beliebiger anderer dem System bekannter Benutzer.

Dies hat durchaus praktischen Nutzen – so kann es vorkommen, dass Sie mehrere Vorgänge parallel anstoßen möchten oder einen Vorgang in Echtzeit beobachten oder überwachen wollen.

Dazu gibt es die virtuellen Konsolen. Sie wechseln zwischen den Konsolen durch die Tastenkombinationen `[Alt]+[F1]` (für die erste Konsole) bis `[Alt]+[F6]` (für die letzte Konsole). Mit `[Alt]+[F7]` gelangen Sie auf eine grafische Oberfläche – vorausgesetzt, diese ist installiert. Von dort kommen Sie mit der Tastenkombination `[Strg]+[Alt]+[F1]-[F6]` wieder auf eine der Textkonsolen, da die Kombination `[Alt]+[F1]-[F6]` bereits durch die GUI belegt ist.

Testen Sie es aus – am besten jetzt. Melden Sie sich an mehreren Konsolen an, mal als `root` und mal als ein normaler Benutzer – einen müssten Sie ja wenigstens bei der Installation erstellt haben.

Auf welcher Konsole Sie sich befinden, bekommen Sie übrigens heraus, indem Sie `tty` eingeben. Die Ausgabe ist die Gerätedatei (zum Beispiel `/dev/tty5` für die fünfte Konsole), die die aktuelle Konsole verwaltet. Haben Sie sich noch nicht angemeldet, wird die Konsole oben rechts über dem Login-Prompt angezeigt.

Geben Sie `logout` ein, um sich an einer Konsole abzumelden. Sie können auch `exit` eingeben – damit schließen Sie die aktuelle Shell (Ihre Benutzerumgebung). Handelt es sich um Ihre Login-Shell (was normalerweise der Fall ist) werden Sie ebenfalls abgemeldet. Näheres hierzu in Kapitel 9 *Einführung in die Bash*.

4.2 Herunterfahren und Neustarten des Systems

Ein ketzerisches Thema! Linux muss nicht neu gestartet werden, alles geht im laufenden Betrieb ... Das ist größtenteils richtig, dennoch kommt es vor, dass Sie Ihren Server herunterfahren oder neu starten möchten, zum Beispiel weil Sie Hardware austauschen müssen, oder weil Sie es als die einfachste Lösung für ein größeres Problem im Zusammenhang mit Programmen oder Diensten sehen, die sich »irgendwie verhakt« haben – sprich: abgestürzt sind und nicht mehr sauber zu beenden und neu zu starten sind.

Vielleicht möchten Sie das System auch ausschalten, um es an einen anderen Ort zu transportieren, oder weil Sie es im Moment nicht mehr benötigen – im Falle einer Workstation kommt Letzteres praktisch jeden Tag (bzw. Abend) vor.

Es sollte selbstverständlich sein, dennoch eine kurze Begründung: Um ein Linux-System sauber beenden zu können, müssen etliche Prozesse ausgeführt bzw. beendet werden. Auf keinen Fall sollten Sie die Hardware einfach ausschalten! Das gefährdet die Konsistenz Ihrer Daten und des Systems bis hin zu möglichen Hardware-Schäden, insbesondere an der Festplatte.

Es gibt mehrere Möglichkeiten, ein Linux-System herunterzufahren oder zu »rebooten«. Für jede dieser Möglichkeiten benötigen Sie `root`-Rechte.

Sowohl `halt` als auch `reboot` greifen auf den Befehl `shutdown` mit entsprechenden Parametern zurück, wie Sie sich selbst durch Aufruf der entsprechenden Online-Hilfeseiten (Man-Pages) überzeugen können. Geben Sie `man 8 halt` bzw. `man 8 reboot` ein. Daher beschränke ich mich auf die Darstellung dieses einen Befehls:

```
# shutdown -h now
```

Hiermit fahren Sie das System herunter (-h) und zwar auf der Stelle (now). Allerdings wird keine Power-Off-Funktion aufgerufen. Dies können Sie mit der Option -P tun. Möchten Sie eine Zeit für diese Aktion festlegen, nutzen Sie die Option -t:

```
# shutdown -P -t <Sekunden>
```

Hiermit wird der Shutdown inklusive Power-Off verzögert, um die angegebenen Sekunden, ausgeführt. Möchten Sie das System rebooten, geben Sie folgenden Befehl ein:

```
# shutdown -r now
```

Damit wird das System sofort heruntergefahren und neu gestartet – sprich: Ein Reboot wird durchgeführt.

Außerdem haben Sie die Möglichkeit, über die Angabe des Runlevels (siehe Kapitel 6 *Der Linux-Systemstart*) das System entweder zu stoppen (Runlevel 0) oder zu rebooten (Runlevel 6). Hierzu geben Sie den Befehl `init` und den gewünschten Runlevel an, um in diesen zu wechseln. Folgender Befehl führt zu einem Neustart:

```
# init 6
```

Mit diesen wenigen Befehlen sollten Sie auskommen – ich habe bisher noch nie einen anderen Befehl benötigt.

4.3 Basisbefehle zur Navigation

Lassen Sie uns die Grundbefehle in gebührender Kürze durchgehen, nur um sicherzustellen, dass Ihnen später nicht das Handwerkszeug für grundlegende Arbeiten fehlt.

4.3.1 Aktuelles Verzeichnis anzeigen lassen

Wie bereits erwähnt, befinden Sie sich nach dem Login in Ihrem Home-Verzeichnis. Welches dies ist, bekommen Sie mit folgendem Befehl heraus:

```
# pwd
```

Dies steht für *print working directory* und zeigt Ihnen das Verzeichnis an, in dem Sie sich aktuell befinden. Normalerweise ist das nicht notwendig, da der Prompt ebenfalls darüber Aufschluss gibt – bei der Tilde (~) ist das allerdings nicht so direkt abzulesen. Wie Sie sehen, befinden Sie sich in `/root` (wenn Sie sich als `root` angemeldet haben) oder in `/home/<Ihr-Benutzername>`, wenn Sie als normaler User angemeldet sind.

4.3.2 Inhalt eines Verzeichnisses anzeigen lassen

Folgender Befehl zeigt Ihnen den Inhalt des aktuellen Verzeichnisses an:

```
# ls
```

Er steht für `list`. Führen Sie `ls` in Ihrem Home-Verzeichnis aus, wird Ihnen womöglich nichts angezeigt, weil noch keine normalen Dateien und keine Verzeichnisse existieren. Geben Sie `ls -a` ein, erhalten Sie zusätzlich die versteckten Objekte angezeigt. Deren Namen beginnen mit einem Punkt (`.`). Möchten Sie detaillierte Informationen, geben Sie `ls -la` ein. Hier ein Beispiel:

```
# ls
# ls -a
.  ..  .aptitude  .bashrc  .profile
# ls -la
insgesamt 5
drwxr-xr-x  3 root root 1024 2007-01-21 19:05 .
drwxr-xr-x 23 root root 1024 2007-01-21 19:07 ..
drwx-----  2 root root 1024 2007-01-21 19:05 .aptitude
-rw-r--r--  1 root root  412 2004-12-15 23:53 .bashrc
-rw-r--r--  1 root root  110 2004-11-10 17:10 .profile
```

In der ersten Spalte der Ausgabe finden Sie den Objekttyp. Verzeichnisse werden durch `d` (für *directory*) gekennzeichnet. Normale Dateien haben hier ein Minus (`-`). Während `.aptitude` ein verstecktes Verzeichnis ist, handelt es sich bei `.bash_history`, `.bashrc` und `.profile` um Dateien.

Hinter dem Objekttyp stehen die Rechte, anschließend die Anzahl der Verlinkungen im Dateisystem, der Eigner des Objekts, die Gruppe, der das Objekt zugeordnet ist, die Größe in Byte, das Änderungsdatum einschließlich der Uhrzeit und schließlich der Name des Objekts.

Für »Objekt« können Sie übrigens auch getrost »Datei« einsetzen, da unter Linux einfach alles – einschließlich der Verzeichnisse und sogar der Hardware – als Datei gehandhabt wird (wie bereits in Kapitel 2 *Debian installieren* erwähnt).

Die versteckten Dateien und Verzeichnisse Ihres Home-Verzeichnisses können Konfigurationsparameter für verschiedene Programme (zum Beispiel `aptitude`) enthalten, die nur für Ihren Benutzer gelten. Es handelt sich also um Ihre Profildateien.

4.3.3 In ein anderes Verzeichnis wechseln

Die ersten beiden Einträge in der Auflistung stehen übrigens für das aktuelle Verzeichnis (`.`) und das übergeordnete Verzeichnis (`..`). Der Befehl `cd` (*change directory*) dient dazu, in bestimmte Verzeichnisse zu wechseln. Als Parameter geben Sie den Pfad zum Verzeichnis an. Ein paar einfache Beispiele:

`cd` (Pfadangabe) wechselt in das Home-Verzeichnis des Benutzers:

`cd /` wechselt in das `/`-Verzeichnis (Wurzelverzeichnis).

`cd ..` wechselt ein Verzeichnis nach oben.

4.3.4 Pfadangaben

Pfade können absolut oder relativ angegeben werden. Das Dateisystem ist hierarchisch untergliedert (siehe nächster Abschnitt). Die Ebenen werden durch / voneinander getrennt. Der oberste Punkt des Dateisystems ist / (*root*, die Wurzel). *Absolute Pfade* werden vom obersten Punkt – *root* – angegeben, zum Beispiel:

/ – das *root*-Verzeichnis (**Achtung:** nicht das Home-Verzeichnis des Benutzers *root*!)

/usr – ein Verzeichnis auf der obersten Ebene

/usr/sbin – ein Unterverzeichnis von /usr

Relative Pfade gehen vom aktuellen Verzeichnis aus und geben relativ hierzu den Weg zum Ziel an. Sie haben kein führendes /.

../usr – eine Ebene nach oben, dort in das Verzeichnis *usr* auf dieser Ebene

../../usr/sbin – drei Ebenen aufwärts, dann in *usr*, darunter in *sbin*.

local/bin – aus aktuellem Verzeichnis in das Unterverzeichnis *local*, darunter *bin*

Damit diese Informationen nun nicht einfach im Raum stehen bleiben, sehen wir uns nun das Dateisystem Ihres Debian GNU/Linux-Systems an.

4.4 Die Struktur des Dateisystems

Der Begriff »Dateisystem« wird in zwei Bedeutungen benutzt:

- Das System, mit dem eine Partition formatiert ist – Hier ist definiert, wie die Bits und Bytes organisiert sind, welche Features unterstützt werden (Journaling, Quotas, Benutzerrechte etc.). Dazu gehören *ext2*, *ext3*, *xf*s, *jfs* usw.
- Das System, das die Verzeichnisstruktur definiert – also die Hierarchie der Verzeichnisse

In diesem Abschnitt geht es um die Verzeichnisstruktur. Es existiert ein internationaler Unix-Standard, an den sich Debian weitgehend hält – FHS (Filesystem Hierarchy Standard), nachzulesen unter <http://www.pathname.com/fhs/>.

Abbildung 4.1 stellt die Grundzüge dieser Verzeichnisstruktur dar. Im Anschluss daran werden wir die einzelnen Verzeichnisse kurz durchsprechen.

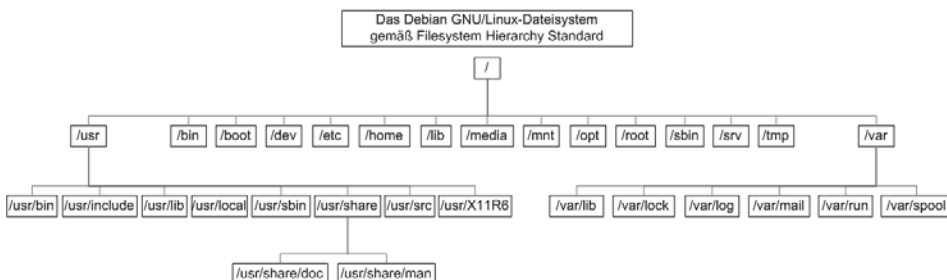


Abb. 4.1: Debian hält sich mit seiner Verzeichnishierarchie weitgehend an das FHS.

Die Abbildung enthält die wichtigsten Verzeichnisse eines Debian-GNU/Linux-Dateisystems entsprechend des FHS.

Sollten Sie noch nicht vertraut sein mit dieser Struktur, empfehle ich Ihnen, parallel zu der folgenden Übersicht in die entsprechenden Verzeichnisse zu wechseln und sich deren Inhalt anzusehen.

Die folgende Aufzählung ist unter Umständen etwas trocken, aber Sie sollten sich trotzdem mit der Verzeichnishierarchie auseinandersetzen. Das fördert das Verständnis für Ihr Debian-GNU/Linux-System und wird später an vielen Stellen sehr nützlich sein.

/

Die Wurzel, root genannt. Dies ist vergleichbar mit C:\, allerdings noch eine Ebene höher, da es unter Windows-Dateisystemen keinen absoluten obersten Punkt gibt – jedes Laufwerk hat einen eigenen Buchstaben. Bei Linux werden den Partitionen und externen Speichermedien Mountpoints in Form von beliebigen Verzeichnissen unterhalb von / zugewiesen, unter denen diese in das Dateisystem eingehängt (gemountet) werden (siehe nächstes Kapitel).

/bin

Das Verzeichnis für Befehle, die auch von normalen Benutzern ausgeführt werden können. Insbesondere Navigationsbefehle (**ls**, **cd**, **pwd**) und Dateimanipulationsbefehle (**cp**, **mv**, **mkdir** etc.) sind hier zu finden. Außerdem wird hier ein Link von `/bin/sh` auf die aktuelle Shell benötigt. Unter Linux ist das standardmäßig `/bin/bash`. Dies ist notwendig, weil Shellskripte allgemein auf `/bin/sh` verweisen.

/boot

Hier befinden sich die Dateien, die für den Bootprozess notwendig sind. Normalerweise befindet sich hier der Kernel.

/dev

Das Geräteverzeichnis – alle ansprechbaren Peripheriegeräte (Maus, Tastatur, Grafikkarte, Festplatten, CD-ROM-Laufwerke etc.) werden hier in Form einer Gerätedatei geführt. Enthielt dieses Verzeichnis früher noch teilweise Hunderte von Einträgen für alle möglichen Geräte, finden Sie in aktuellen Linux-Distributionen neuerdings nur noch die relevanten Gerätedateien, da diese zu einem Teil von einem Dienst namens `udev` entsprechend der Hardware-Erkennung dynamisch erstellt werden. Im nächsten Kapitel werde ich auf dieses Thema noch näher eingehen.

/etc

Enthält die (Text-)Konfigurationsdateien der Programme. Einstellungen in diesen Dateien gelten für das gesamte System. Existieren optionale Konfigurationsdateien in den Home-Verzeichnissen der Benutzer, überschreiben diese die allgemeinen Einstellungen. Unter `/etc` können weitere Verzeichnisse liegen, zum Beispiel `/etc/postfix`, `/etc/apache`, `/etc/X11` usw.

/home

Hier liegen die Home-Verzeichnisse der Benutzer. Sie haben normalerweise den Anmeldenamen des Benutzers. Eine Ausnahme stellt `root` dar. Dessen Home-Verzeichnis liegt unter `/root`. Dies hat sicherheitstechnische Gründe, da auf dieses Verzeichnis gesonderte Rechte gesetzt werden können – ein normaler User kommt damit nicht in das Verzeichnis hinein.

/lib

Enthält die für den Systemstart und die elementaren Prozesse notwendigen dynamischen Bibliotheken, *shared libraries* genannt.

/media

Mountpoint für externe transportable Datenspeicher wie zum Beispiel USB-Sticks und -Festplatten, Floppy-Disk, CD/DVD-ROM usw.

Dieses Konzept wird verstärkt verfolgt und löst die Sitte ab, solche Datenspeicher unter `/mnt/<medium>` zu mounten.

/mnt

Wird oft noch mit Unterverzeichnissen als Mountpoint für externe Speichermedien genutzt (zum Beispiel `/mnt/cdrom`), ist aber eigentlich als direkter Einhängpunkt für temporäre gemountete Dateisysteme gedacht.

/opt

Ist reserviert für die Installation zusätzlicher Software-Pakete. Dazu wird häufig auch `/usr/local` verwendet.

/root

Home-Verzeichnis des Benutzers `root`.

/sbin

Hier liegen die Befehle für die Systemadministration (*root-only commands*). Normale Benutzer haben keinen Zugriff auf diese Kommandos. Weitere Speicherorte für diese Art von Befehlen sind `/sbin`, `/usr/sbin` und `/usr/local/sbin`.

/srv

Dieses Verzeichnis enthält in Unterverzeichnissen spezielle Daten für Serverdienste, zum Beispiel FTP, rsync usw.

/tmp

Das `/tmp`-Verzeichnis dient Programmen zur temporären Ablage von Dateien. Es wird in der Voreinstellung bei jedem Systemstart gelöscht.

`/usr`

Unter dieser Verzeichnishierarchie befinden sich in der Regel nur lesbare Dateien, die mit anderen Hosts geteilt werden können. Das Verzeichnis enthält zahlreiche Unterverzeichnisse.

`/usr/bin`

Dies enthält die meisten User-Befehle, die für alle Benutzer zur Verfügung stehen.

`/usr/include`

Hier befinden sich die Standard-Include-Dateien für die Programmiersprachen C und C++.

`/usr/lib`

Enthält Objektdateien, Bibliotheken und interne Binärdateien für Programme und Pakete. Diese Dateien sind nicht für den direkten Gebrauch durch Benutzer oder Shellskripte vorgesehen.

`/usr/local`

Dieses Verzeichnis wird vom Administrator zur Installation von lokaler Software genutzt, ähnlich wie `/opt`. Es enthält oft noch etliche Unterverzeichnisse wie `/usr/local/bin`, `/usr/local/etc` usw.

`/usr/share`

Enthält architekturunabhängige Daten. Hier befinden sich in Unterverzeichnissen auch die Man-Pages und Dokumentation zu den einzelnen Programmen und Tools.

`/usr/share/doc`

Hier befinden sich die Dokumentationsdateien der Programme und Pakete.

`/usr/share/man`

Hauptverzeichnis für die Man-Pages. Nicht alle Pakete halten sich an diese Struktur, aber in der Regel finden Sie die Online-Hilfen für die einzelnen Programme, Konfigurationsdateien und Strukturen unterhalb dieses Verzeichnisses.

`/usr/src`

Enthält den Sourcecode von Programmen und dem Kernel – dies ist optional. In Kapitel 15 *Den Kernel anpassen* komme ich darauf zurück.

`/var`

In diesem Verzeichnis liegen variable Daten. Dazu gehören die Logfiles, Drucker-Spool-Verzeichnis, Mailboxen etc.

`/var/lib`

Enthält Statusinformationen über bestimmte Applikationen. Diese Informationen können von den jeweiligen Programmen zur Laufzeit verändert werden.

/var/lock

In diesem Verzeichnis befinden sich so genannte Lockfiles (engl. *to lock* = sperren). Mit diesen Dateien werden bestimmte Ressourcen (zum Beispiel ein serieller Anschluss) für andere Anwendungen geblockt, so dass nur ein einziger Prozess darauf Zugriff hat. Dies dient der Vermeidung von Inkonsistenzen.

/var/log

Davon zu unterscheiden sind die Logdateien, die sich in dieser Verzeichnisstruktur befinden. Auf sie komme ich in Kapitel 14 *Protokollierung* zurück.

/var/mail

Hier befinden sich die Mailboxen der Benutzer. Die Spool-Dateien haben den Login-Namen.

/var/run

Dies sind so genannte *variable Runtime*-Daten. Hier sind Informationen über den aktuellen Zustand des Systems abgelegt. Diese werden zur Laufzeit (*runtime*) erstellt. Hier befinden sich auch die Prozess-ID-Dateien der Form:

```
<Programm-Name>.pid.
```

/var/spool

Dieses Verzeichnis enthält Daten, die in irgendeiner Art noch weiterverarbeitet werden, zum Beispiel Maildaten. Es existiert ein symbolischer Link (siehe Abschnitt 4.5.8) von `/var/spool/mail` auf `/var/mail`.

4.5 Dateioperationen

Als Systemadministrator werden Sie sehr oft Dateien manipulieren wollen oder müssen. Hier schauen wir uns an, wie Sie Dateien und Verzeichnisse

- erstellen,
- bearbeiten,
- kopieren,
- verschieben,
- umbenennen,
- löschen und
- verknüpfen

können. Damit Sie ein wenig Routine darin bekommen, werde ich Ihnen im Anschluss daran eine kleine Übung anbieten, mit der Sie Ihre Kenntnisse in die Praxis umsetzen können.

4.5.1 Dateien und Verzeichnisse erstellen

Sie können eine neue Datei mit folgendem Kommando erstellen:

```
# touch <Dateiname>
```

Es wird eine leere Textdatei im aktuellen Verzeichnis erstellt. Existiert die angegebene Datei bereits, wird die Zeitangabe der Datei aktualisiert. Ein neues Verzeichnis erstellen Sie mit folgendem Befehl:

```
# mkdir <Verzeichnis>
```

Im aktuellen Verzeichnis wird ein Unterverzeichnis mit dem angegebenen Namen erstellt. Mit der Option `-p` können Sie mehrere Verzeichnisse untereinander gleichzeitig anlegen:

```
# mkdir -p <Verzeichnis1>/<Verzeichnis2>
```

Ohne diese Option muss das jeweils darüber liegende Verzeichnis (`Verzeichnis1`) bereits existieren. Ein Beispiel soll das verdeutlichen:

```
# mkdir -p /oben/darunter
```

erstellt auf der obersten Ebene (unter `/`) ein Verzeichnis `oben` und im gleichen Zug ein Unterverzeichnis namens `darunter`.

Beachten Sie: Der oben angegebene `mkdir`-Befehl kann von überall aufgerufen werden! Ihr aktuelles Verzeichnis muss sich nicht an der Stelle befinden, wo Sie eine Dateimanipulation vornehmen möchten – Sie können bei fast jedem Befehl einen Pfad angeben, um die Stelle im Dateisystem anzugeben, an der etwas geschehen soll (zum Beispiel das Erstellen eines Verzeichnisses). Nutzen Sie absolute Pfade, ist es völlig egal, wo Sie sich im Moment befinden.

Beachten Sie weiterhin, dass Linux zwischen Groß- und Kleinschreibung unterscheidet: `verzeichnis1` ist nicht gleich `Verzeichnis1`!

4.5.2 Textdateien bearbeiten mit nano

Eine leere Datei nutzt Ihnen in den seltensten Fällen etwas – in der Regel möchten Sie den Inhalt bearbeiten. Unter Linux stehen Ihnen Dutzende von Editoren zur Verfügung.

Zwar ist `vi` bzw. `vim` (`vi improved`) nach wie vor der kleinste gemeinsame Nenner auf fast allen Unix-artigen Systemen, aber er ist in der Bedienung alles andere als intuitiv! Er stammt aus der Unix-Urzeit und war auf Effizienz und nicht auf Bedienbarkeit ausgelegt. Trotz seiner geringen Größe (ein gutes Megabyte) ist der Editor unglaublich leistungsfähig – Suchen und Ersetzen, Makros usw.

Auf der anderen Seite gibt es das Nonplusultra: `emacs` – ein Alleskönner, der vollkommen anders konzipiert ist als `vi` (es gibt Gerüchte, dass dieser Editor auch Kaffee kochen kann ...). Emacs wird allerdings aufgrund seines Umfangs nicht auf jedem Basissystem installiert.

Es gibt eigentlich nur zwei Lager: Die einen lieben `vi` (und hassen `emacs`), die anderen halten es genau umgekehrt und schwören auf `emacs` – das wird zelebriert, das ist Kult! Ein Mittelding gibt es nicht, darf es auch nicht geben.

Abgesehen davon sollten Sie sich in jedem Fall ein wenig Grundlagenwissen zu **vi** bzw. **vim** aneignen – ob Sie ihn mögen oder nicht. Irgendwie stolpert man im Laufe der Zeit doch immer wieder über diesen kleinen, etwas zickigen, aber irgendwie urigen Editor. Er repräsentiert im Grunde das gesamte Wesen von Unix und Linux – ungeheuer mächtig, aber nicht immer leicht zugänglich. Etwas später werde ich Ihnen noch einen kleinen Workshop zu **vim** anbieten, falls Sie meinem Rat folgen möchten und sich für alle (Linux-)Eventualitäten wappnen möchten. Doch zunächst habe ich noch eine gute Nachricht für Sie:

Auf den meisten Linux-Versionen ist auch nach einer Basisinstallation meist **nano** vorhanden – auf unserem Debian-System ist er sogar der Standardeditor!

Bei **nano** handelt es sich um einen kleinen GNU-Editor, der für unsere Zwecke vollkommen ausreicht. Sie öffnen ihn entweder durch Eingabe von **nano** **<Dateiname>** oder – da es der Standardeditor ist – durch **editor** **<Dateiname>**.

Sie rufen den Standardeditor immer über **editor** auf. Dieser Eintrag befindet sich unter `/usr/bin/`. Es handelt sich hierbei um einen Softlink auf `/etc/alternatives/editor`. Hierbei handelt es sich ebenfalls um einen Softlink, der auf das echte Editorprogramm (in diesem Fall **nano**) in `/bin` zeigt. Den Standardeditor können Sie ändern, indem Sie den Softlink für `/etc/alternatives/editor` ändern (siehe Abschnitt 4.5.8) und auf einen anderen Editor zeigen lassen, zum Beispiel auf **ed**. Bemerkung: Ist **vim** installiert, wird er als Standardeditor eingetragen. Bei Debian *Etch* und *Lenny* ist das aber standardmäßig nicht der Fall.

Haben Sie **nano** gestartet, können Sie Ihre Eingaben wie in jedem anderen »normalen« Editor machen – **vi** (und **vim**) gehört *nicht* in die Kategorie »normal«.

```

GNU nano 1.2.4           Datei: /etc/fstab
# /etc/fstab: static file system information.
#
# <file system> <mount point> <type> <options> <dump> <pass>
proc /proc proc defaults 0 0
/dev/hda1 / ext3 defaults,errors=remount-ro 0 1
/dev/hda9 /home ext3 defaults 0 2
/dev/hda8 /tmp ext3 defaults 0 2
/dev/hda5 /usr ext3 defaults 0 2
/dev/hda6 /var ext3 defaults 0 2
/dev/hda7 none swap sw 0 0
/dev/hdc /media/cdrom0 iso9660 ro,user,noauto 0 0
/dev/fd0 /media/ floppy0 auto rw,user,noauto 0 0

[ 12 Zeilen gelesen ]
^G Hilfe ^O Speichern ^R Datei öffn ^Y Seite zurück ^K Ausschneid ^C Cursor
^X Beenden ^J Ausrichten ^W Wo ist ^U Seite vor ^U Einfügen ^T Rechtschr.

```

Abb. 4.2: Der Debian-Standardeditor nano

In diesem Fall haben wir die Datei `/etc/fstab` geöffnet (siehe nächstes Kapitel). Haben Sie Ihre Änderungen vorgenommen, können Sie die Datei speichern, indem Sie `[Strg]+[o]` drücken. Bei Bedarf können Sie an dieser Stelle den Namen der Datei ändern. Mit `[Strg]+[r]` öffnen Sie eine Datei. Geben Sie hier entweder den relativen Pfad ausgehend vom aktuellen Verzeichnis an oder besser den absoluten Pfad zur gewünschten Datei.

Mit `[Strg]+[c]` brechen Sie die aktuelle Aktion (Speichern, Laden etc.) ab. `[Strg]+[x]` beendet den Editor. Haben Sie Änderungen vorgenommen, müssen Sie die Beendigung noch bestätigen. Letztlich steht alles im unteren Bereich.

Mit `[Strg]+[g]` erhalten Sie eine kurze Hilfe, die aber alle wichtigen Tastenkürzel enthält. Beachten Sie: In der Hilfe steht die Zeichenkombination `^<Taste>` für `[Strg]+<Taste>`. Die Darstellung sieht auf den ersten Blick zwar merkwürdig aus, ist aber allgemeine Konvention. So wird also `[Strg]+[G]` durch `^G` dargestellt.

4.5.3 vim – ein Crashkurs

Da **vim** bzw. im Härtefall **vi** der kleinste gemeinsame Nenner auf einem Linux/Unix-System ist, möchte ich Ihnen an dieser Stelle einen Mini-Crashkurs anbieten, um sich mit diesem durchaus *nicht* intuitiv zu bedienenden Ur-Editor anzufreunden. Die meisten **vi**-Kommandos funktionieren auch beim **vim** – Letzterer ist allerdings deutlich erweitert worden, um die Funktionalität und Bedienbarkeit zu verbessern.

vim arbeitet in verschiedenen Modi:

1. *Kommandomodus* (command mode): Dies ist der Modus, in dem **vim** sich nach dem Start befindet. Hier können Sie Kommandos eingeben, die in der Regel einer bestimmten Taste entsprechen. In diesem Modus werden Eingaben nicht als Text im Dokument angezeigt. So kommen Sie zum Beispiel über die Taste `i` in den *Einfüge-Modus*.
2. *Einfüge-Modus* (insert mode): In diesem Modus können Sie den Inhalt der Textdatei bearbeiten. Vom *Kommandomodus* gelangen Sie in den Einfüge-Modus durch die Tasten `i` (*insert*), `a` (*append*, Zeichen werden hinter die Stelle des Cursors angefügt), `o` (eine neue Zeile wird unter dem Cursor eröffnet) und `O` (eine neue Zeile wird über dem Cursor eröffnet). Über `[ESC]` gelangen Sie zurück in den *Kommandomodus*.
3. *Visueller Modus* (visual mode): Dieser dient zum Markieren und Bearbeiten von Textpassagen, ähnlich, wie Sie es von Programmen wie MS Word kennen. Die markierten Textpassagen können gelöscht, kopiert oder verschoben werden.
4. *Kommandozeilen-Modus* (command-line mode): In diesem Modus können Sie einzeilige Kommandos absetzen. Er wird durch Eingabe von Doppelpunkt (`:`) gestartet. Anschließend können Sie bestimmte Kommandos, wie zum Beispiel `:syntax on` eingeben.

Es gibt weitere Modi, die wir jedoch nicht weiter betrachten wollen.

Im Kommandomodus können Sie zwar den Text nicht direkt bearbeiten, jedoch den Cursor an einer beliebigen Stelle positionieren. Hierzu können Sie die Cursor-Tasten und die üblichen Sondertasten wie `[Pos1]`, `[Ende]`, `[Bild↑]` und `[Bild↓]` nutzen. Anschließend gehen Sie in den Einfüge- oder in den visuellen Modus.

Workshop: Einführung in vim

Gehen wir ein kurzes Beispiel durch:

Zunächst öffnen Sie **vim** und geben ihm als Dateinamen `test.txt` an:

```
# vim test.txt
```

Der Editor öffnet sich mit leeren Zeilen. Geben Sie `i` ein, um in den *Einfüge-Modus* zu gelangen. Anschließend geben Sie einige Zeilen ein, um mit diesen experimentieren zu können. Verlassen Sie den *Einfüge-Modus* anschließend wieder mit `[ESC]`.

Nun wollen wir eine Textpassage kopieren. Hierzu bewegen Sie den Cursor an die erste Stelle der ersten zu kopierenden Zeile und geben im *Kommandomodus* `v` ein, um in den *visuellen Modus* zu wechseln. Bewegen Sie den Cursor mit den Pfeiltasten nach rechts, links, oben oder unten, um den gewünschten Bereich auszuwählen (siehe Abbildung 4.3).

```
Dies ist ein Testtext, der keine weitere Bedeutung hat.  
Er besteht aus vielen, vielen sinnlosen Zeilen.  
Ein Wort jagt das nächste, und es ist kein Ende in Sicht.  
Manchmal muss man sich schon fragen, wie ein einzelner Autor auf soviel Unsinn kommen kann.  
Andererseits stecken in manchen Texten tiefe Wahrheiten, die nur erkannt werden wollen.  
Nun ja, in diesem Text sicherlich nicht ...
```

Abb. 4.3: Im visuellen Modus können Textpassagen markiert werden.

Ist die gewünschte Textpassage markiert, drücken Sie `y`, um den Text in die Zwischenablage (den Puffer) zu kopieren. Sie gelangen anschließend automatisch wieder in den *Kommandomodus*. Bewegen Sie jetzt den Cursor an die erste Stelle des Textes und drücken Sie `O`, um vom *Kommandomodus* in den *Einfüge-Modus* zu gelangen und eine Zeile über der aktuellen Cursorposition zu öffnen. Der Cursor blinkt nun in der neuen Zeile. Nun gehen Sie aus dem *Einfüge-Modus* über `[ESC]` zurück in den *Kommandomodus* und fügen über Eingabe von `p` die Textpassage in der neuen Zeile ein. Das Ergebnis stellt sich in unserem Beispiel so dar, wie in Abbildung 4.4 gezeigt.

```
Manchmal muss man sich schon fragen, wie ein einzelner Autor auf soviel Unsinn kommen kann.  
Andererseits stecken in manchen Texten tiefe Wahrheiten, die nur erkannt werden wollen.  
Nun ja, in diesem Text sicherlich nicht ...  
Dies ist ein Testtext, der keine weitere Bedeutung hat.  
Er besteht aus vielen, vielen sinnlosen Zeilen.  
Ein Wort jagt das nächste, und es ist kein Ende in Sicht.  
Manchmal muss man sich schon fragen, wie ein einzelner Autor auf soviel Unsinn kommen kann.  
Andererseits stecken in manchen Texten tiefe Wahrheiten, die nur erkannt werden wollen.  
Nun ja, in diesem Text sicherlich nicht ...
```

Abb. 4.4: Nach dem Einfügen der Textpassage ist der Text zweimal vorhanden.

Spielen wir weiter. Als Nächstes löschen wir die zweite Zeile. Dies bewerkstelligen Sie, indem Sie den Cursor im *Kommandomodus* in die entsprechende Zeile bewegen und `dd` (Doppel-D) eingeben. Die zweite Zeile ist verschwunden.

```
Manchmal muss man sich schon fragen, wie ein einzelner Autor auf soviel Unsinn kommen kann.  
Nun ja, in diesem Text sicherlich nicht ...  
Dies ist ein Testtext, der keine weitere Bedeutung hat.
```

Abb. 4.5: Die zweite Zeile war einmal ...

Das war nun aber gar nicht das, was Sie wollten – stattdessen war Zeile Nummer 3 überflüssig! Also müssen wir die eben vorgenommene Änderung rückgängig machen. Dies

geschieht im *Kommandomodus* durch Drücken von u (für *undo*). Schwupps, die Zeile ist wieder da. Nehmen Sie stattdessen nun wie beschrieben Zeile 3 heraus.

```
Manchmal muss man sich schon fragen, wie ein einzelner Autor auf soviel Unsinn kommen kann.
Andererseits stecken in manchen Texten tiefe Wahrheiten, die nur erkannt werden wollen.
Dies ist ein Testtext, der keine weitere Bedeutung hat.
```

Abb. 4.6: Zeile 2 ist wieder da, stattdessen ist Zeile 3 nun weg.

Sie können übrigens mehrere Zeilen auf einmal löschen, indem Sie vorher die gewünschte Anzahl der Zeilen angeben, zum Beispiel 10dd. Die Angabe von Zahlen vor dem Befehl funktioniert bei vielen *vim*-Kommandos.

Natürlich ist es mit *vim* auch möglich, nach einer Zeichenkette zu suchen. Hierzu geben Sie im *Kommandomodus* /<Suchbegriff> ein. Wird der Suchbegriff gefunden, blinkt der Cursor unter dem Begriff. Mit n können Sie sich die nächste Fundstelle anzeigen lassen.

Haben Sie *vim* mit Dateinamen aufgerufen und die Textdatei erstellt bzw. eine Änderung an einer bestehenden Datei vorgenommen, möchten Sie diese sicherlich speichern. Hierzu wechseln Sie über : in den *Kommandozeilenmodus* und geben w ein. Haben Sie *vim* ohne Dateinamen aufgerufen oder möchten Sie die Datei unter einem anderen Namen abspeichern, geben Sie :w <Dateiname> an. <Dateiname> kann auch einen relativen oder absoluten Pfad enthalten, also zum Beispiel /etc/test.cfg.

Möchten Sie *vim* verlassen, geben Sie im *Kommandomodus* :q ein. Haben Sie eine Änderung vorgenommen, die noch nicht gespeichert wurde, verweigert *vim* dies.

```
E37: No write since last change (add ! to override)                2,1          all
```

Abb. 4.7: Regulär verweigert *vim* die Beendigung, wenn Änderungen nicht gespeichert wurden.

Ergänzen Sie den Befehl durch Ausrufezeichen (!), können Sie die Beendigung erzwingen. Der Befehl zum Beenden ohne Speichern lautet also :q!.

Fast alle Befehle werden im *Kommandomodus* eingegeben. Geben Sie im Zweifel immer ein oder zweimal [ESC] ein, wenn Sie nicht sicher sind, wo Sie sich gerade befinden.

Im Folgenden finden Sie noch einmal die wichtigsten *vim*-Befehle:

Befehl	Bedeutung
i, a, o, O	Vom Kommandomodus in den Einfüge-Modus wechseln: i – Zeichen können an der aktuellen Cursorposition eingefügt werden. a – Zeichen können nach der aktuellen Cursorposition eingefügt werden. o – neue Zeile unterhalb der aktuellen Cursorposition O – neue Zeile oberhalb der aktuellen Cursorposition

Befehl	Bedeutung
:w [<Dateiname>]	Datei speichern (ggf. unter dem angegebenen Namen)
:q[!]	Editor beenden, mit ! Beendigung erzwingen
[<Anzahl>dd	Zeile löschen, ggf. ab Cursorposition die angegebene Anzahl der Zeilen
v	Visueller Modus
y	Markierten Text im visuellen Modus in den Puffer schreiben und in den Kommandomodus zurückspringen
p	Text im Puffer an der aktuellen Cursorposition einfügen (im Kommandomodus)
/<Suchbegriff>	Suche nach Suchbegriff
n	Nächste Fundstelle im Text
w	Im Kommandomodus: Wort vorwärts
b	Im Kommandomodus: Wort rückwärts
x	Im Kommandomodus: Zeichen unter dem Cursor löschen (funktioniert auch bei vi, bei vim auch <code>Entf</code> möglich)
:set number	Zeigt die Zeilennummern an
:syntax on	Aktiviert das Syntax-Highlighting, vim unterstützt eine Vielzahl von Programmiersprachen

Dies ist lediglich eine kurze Einführung, um sich eigenständig im **vim** bewegen zu können. Sie werden etwas Geduld benötigen, bis Sie sich mit dem Editor angefreundet haben. Andererseits sollten Sie sich meiner Erfahrung nach in jedem Fall etwas Basiswissen über **vim** aneignen, da Sie immer wieder über dieses Urgestein stolpern werden, wenn Sie auch mit anderen Linux-Systemen außer Ihrem eigenen zu tun haben.

4.5.4 Textdateien betrachten

Möchten Sie eine vorhandene Textdatei lediglich betrachten, ohne sie zu verändern, bieten sich gleich fünf Programme an:

```
# more <Datei>
```

Der alte – und auf fast allen Systemen vorhandene – Pager zeigt Ihnen eine Datei seitenweise an. Mit `Leertaste` gehen Sie weiter, zurück können Sie nicht. Die Taste `q` beendet den Betrachter.

```
# less <Datei>
```

Der Standard-Pager unter Linux ist **less**. Unter Debian GNU/Linux müssen Sie ihn allerdings unter Umständen erst nachinstallieren, wie Sie im letzten Kapitel gesehen haben, da er in der Basisinstallation nicht enthalten ist.

Der Aufruf funktioniert genauso wie **more**. Sie können mit den Cursortasten zeilenweise hoch- und runterscrollen und mit `/<Suchbegriff>` sogar nach Begriffen suchen. Mit `n` kommen Sie zur nächsten Fundstelle, `Shift+n` führt zur vorigen Fundstelle. Wiederum

beendet `q` das Programm. Erwähnenswert: Durch die Taste `v` wird der Standardeditor gestartet. Verlassen Sie den Editor anschließend, landen Sie wieder im Pager **less**.

```
# cat <Datei>
```

Dieser Befehl ist eigentlich für andere Dinge ausgelegt. Der Name kommt von *concatenate* (zusammenfügen). Genau das kann das Programm dann auch: Textdateien auf bestimmte Arten zusammenfügen. Als »Nebenprodukt« wird eine Textdatei von Anfang bis Ende ohne Unterbrechung auf dem Bildschirm angezeigt, wenn die obige Syntax verwendet wird. Dies ist insbesondere nützlich, wenn man den Inhalt einer Datei durch einen Filter schicken möchte. Näheres hierzu in Kapitel 10 *Wichtige Befehle zur Systemadministration*.

```
# head -n<Zeilen> <Datei>
```

Hiermit werden die ersten Zeilen einer Datei angezeigt. `-n40` zeigt die ersten 40 Zeilen an. Wird keine Option angegeben, werden die ersten zehn Zeilen angezeigt.

```
# tail -n<Zeilen> <Datei>
```

Dieses Kommando funktioniert wie **head**, nur umgekehrt: Sie lassen sich die letzten Zeilen einer Datei anzeigen. Per Default zehn Zeilen, was aber durch `-n<Zeilen>` angepasst werden kann. Dieser Befehl bietet noch einen sehr interessanten Modus mit der Option `-f`. Hiermit wird die Anzeige ständig aktualisiert. Das ist besonders nützlich, wenn man »live« Logfile-Einträge beobachten möchte.

4.5.5 Kopieren von Dateien und Verzeichnissen

Der Befehl zum Kopieren von Dateien und Verzeichnissen lautet folgendermaßen:

```
# cp <Optionen> <Quelle> <Ziel>
```

Möchten Sie zum Beispiel einen in das Home-Verzeichnis von `root` (`/root`) heruntergeladenen Tarball namens `gulugulu.tar.gz` zur Installation nach `/usr/local` kopieren, so lautet der Befehl folgendermaßen:

```
# cp /root/gulugulu.tar.gz /usr/local
```

Zum Kopieren von Verzeichnissen benötigen Sie die Option `-r` (rekursiv), um auch den Inhalt des Verzeichnisses einschließlich der Unterverzeichnisse zu erfassen. Beispiel:

```
# cp -r /home/hans/ordner1 /tmp
```

Dieser Befehl kopiert den Ordner `Ordner1` aus dem Home-Verzeichnis des Benutzers `hans` einschließlich aller Unterverzeichnisse nach `/tmp`.

4.5.6 Verschieben und Umbenennen

Beim Verschieben und Umbenennen handelt es sich um ein und denselben Befehl: **mv** (move). Je nach Kontext wird eine Datei oder ein Verzeichnis umbenannt oder verschoben. Die Syntax ist so ziemlich dieselbe wie beim Kopierbefehl:

```
# mv <Quelle> <Ziel>
```

Zur Veranschaulichung schauen wir uns wieder ein Beispiel an:

```
# mv text1 Verzeichnis1
```

Damit verschieben Sie eine Datei `text1` aus dem aktuellen Verzeichnis in ein Unterverzeichnis `Verzeichnis1`. **Jetzt aufgepasst:** Wenn das angegebene Verzeichnis `Verzeichnis1` nicht existiert, wird die Datei `text1` einfach in `Verzeichnis1` umbenannt – der Befehl ist also kontextsensitiv.

Das Verschieben und Umbenennen von Verzeichnissen funktioniert exakt genauso:

```
# mv Verzeichnis1 Verzeichnis2
```

Je nach Kontext wird `Verzeichnis1` in ein Unterverzeichnis `Verzeichnis2` verschoben oder einfach in `Verzeichnis2` umbenannt – wenn Letzteres nicht existiert.

Das Verschieben von Verzeichnissen funktioniert – im Gegensatz zum Kopierbefehl – ohne Zusatzoptionen.

4.5.7 Löschen von Dateien und Verzeichnissen

Das mit dem Löschen ist so eine Sache unter Linux! Im Gegensatz zu Windows werden Sie als Linux-Administrator nämlich nicht vor sich selber geschützt – zu Deutsch: Weg ist weg! Seien Sie also sehr vorsichtig mit den Löschbefehlen.

Zum Löschen einfacher Dateien nutzen Sie folgenden Befehl:

```
# rm <Datei>
```

Sie können bei Bedarf mehrere Dateien durch Leerzeichen voneinander getrennt angeben. Möchten Sie (leere) Verzeichnisse löschen, nutzen Sie folgenden Befehl:

```
# rmdir <Verzeichnis>
```

Wie erwähnt: Das angegebene Verzeichnis muss leer sein, sonst folgt eine Fehlermeldung. Möchten Sie ein nicht-leeres Verzeichnis löschen – womöglich noch mit Unterverzeichnissen –, nutzen Sie den Befehl `rm` mit der Option `-r` (rekursiv):

```
# rm -r <Oberstes Verzeichnis>
```

Dieser Befehl löscht die gesamte Verzeichnisstruktur ab dem angegebenen obersten Verzeichnis.

Denken Sie daran: Sie müssen sich *nicht* in den Verzeichnissen befinden, in denen sich etwas abspielen soll (kopieren, verschieben, löschen etc.). Sie können jederzeit und für jeden Parameter (Datei oder Verzeichnis) einen Pfad angeben, wenn sich das Objekt woanders befindet oder danach woanders befinden soll.

4.5.8 Eine Verknüpfung erstellen

Sie können einen Verzeichniseintrag auf eine Datei oder ein Verzeichnis zeigen lassen, das sich an einem anderen Ort befindet. Dies nennt sich *link* (engl. *to link* = verknüpfen). Eine Verknüpfung ist zum Beispiel sinnvoll, wenn Sie – sagen wir als Benutzer *hans* – bequem von Ihrem Home-Verzeichnis auf ein Projektverzeichnis `/home/Projekte/2009/Linux-Team/Operating/Migration_Debian_5.0` zugreifen möchten. Sie können jedes Mal den normalen Weg über den Verzeichniswechsel gehen:

```
# cd /home/Projekte/2009/Linux-Team/Operating/Migration_Debian_5.0
```

Das ist natürlich relativ aufwändig. Der elegantere Weg führt über einen *symbolischen Link*, auch *Symlink* oder *Softlink* genannt:

```
# ln -s /home/Projekte/2009/Linux-Team/Operating/Migration_Debian_5.0 ~/Lenny
```

Damit wird ein Eintrag in Ihrem Home-Verzeichnis (`/home/hans`) namens *Lenny* erstellt. Dieser Eintrag verweist auf das angegebene Verzeichnis `/home/Projekte/2009/Linux-Team/Operating/Migration_Debian_5.0`. Der Eintrag stellt sich anschließend folgendermaßen dar:

```
hans@etch:~$ ls -l
insgesamt 4
lrwxrwxrwx 1 hans hans 62 2007-01-21 20:22 Lenny -> /home/Projekte/2009/Linux-Team/
Operating/Migration_Debian_5.0/
```

Beachten Sie, dass als Dateityp ganz links ein `l` steht, um den symbolischen bzw. Softlink (sorry, aber es existieren wirklich beide Bezeichnungen!) zu kennzeichnen. Im Übrigen sehen Sie auch hinter dem Pfeil nach dem Namen, auf welchen Ort der Link zeigt.

Möchten Sie nun aus Ihrem Home-Verzeichnis in das o.a. Verzeichnis wechseln, geben Sie einfach Folgendes ein:

```
hans@etch:~$ cd Lenny
```

Kurz, knackig und erheblich weniger Tipperei. Ich gehe einmal selbstbewusst davon aus, dass Sie dieses Beispiel bereits überzeugt hat; aber es gibt noch andere Gründe, einen Link zu erstellen. Insbesondere kann somit ein fester Eintrag bei Bedarf auf unterschiedliche Ziele zeigen. Nehmen wir den Standardeditor. Wie Sie wissen, rufen Sie ihn mit **editor** `<Dateiname>` auf. Hinter den Kulissen passiert Folgendes:

Der Befehl **editor** befindet sich als Softlink in `/usr/bin/`. Überzeugen Sie sich selbst:

```
# ls -l /usr/bin/editor
lrwxrwxrwx 1 root root 24 2007-01-21 18:43 /usr/bin/editor -> /etc/alternatives/
editor
```

Der Link zeigt auf einen Eintrag namens *editor* in `/etc/alternatives/`. Dieses Verzeichnis wird unter Debian dafür verwendet, um virtuelle Pakete anzusprechen – sprich, um bestimmte Funktionalitäten (*editor*, *www-browser* usw.) abzubilden. Damit kann das Standardprogramm für diese Funktionalität leicht ausgetauscht werden. Warum? Weil wir in `/etc/alternatives` wiederum nur einen Link – diesmal auf das echte Programm – finden:

```
# ls -l /etc/alternatives/editor
lrwxrwxrwx 1 root root 9 2007-01-21 18:43 /etc/alternatives/editor -> /bin/nano
```

Dieses zeigt hier auf `/bin/nano`. Möchten Sie hier einen anderen Editor angeben, löschen Sie zunächst den alten Link und erstellen wie oben gezeigt einen neuen, der auf den anderen Editor zeigt. Und schon haben Sie einen anderen Standardeditor.

Diese komplizierte Struktur mag zunächst recht sinnfrei erscheinen – aber wenn Sie es im Gesamtzusammenhang betrachten, sorgen solche Strukturen für ein klares Konzept, das man nur erst einmal »gefressen« haben muss ... das wiederum dauert natürlich eine kleine Weile.

Warum spreche ich eigentlich die ganze Zeit von Softlinks? Gibt es auch Hardlinks? Allerdings, die gibt es! Sie werden viel seltener genutzt und sind weniger flexibel. Einen Hardlink erstellen Sie wie einen Softlink mit `ln`, allerdings lassen Sie die Option `-s` weg. Das Ergebnis ist ein Verzeichniseintrag, der sich unter `ls -l` in keiner Weise von anderen Dateien unterscheidet – kein `l` in der Spalte `Dateityp` ganz links und kein Pfeil mit dem Verweis auf das Ziel hinter dem Dateinamen. Dies funktioniert nur innerhalb einer Partition, während Softlinks auch partitionsübergreifend eingesetzt werden können.

Hintergrund: Während ein *Softlink* eine eigene (zweite) Datei als Link erstellt, ist ein *Hardlink* ein Eintrag mit anderem Namen aber gleichem *Inode* wie das Ziel. Der *Inode* identifiziert eine Datei in einem Dateisystem – daher stellt ein *Hardlink* lediglich einen zweiten Namen für ein und dieselbe Datei dar. Sie können sich die Inodes mit `ls -li` anzeigen lassen, natürlich auch kombiniert mit anderen Optionen des Befehls, zum Beispiel `ls -lia`.

Und noch ein wenig Background: Vielleicht haben Sie sich schon gefragt, welche Arten von Dateien Linux noch so auf Lager hat? Hier eine Übersicht:

Zeichen	Dateityp
-	Normale Datei (Text, Binär, Archiv usw.)
d	Directory (Verzeichnis)
l	Symbolische Links (s.o.)
b	Blockorientiertes Gerät – ermöglicht einen wahlfreien Zugriff, zum Beispiel Festplatte oder CD-ROM
c	Character (Zeichen-)orientiertes Gerät – liest sequenziell, also zum Beispiel Streamer oder serielle Schnittstelle
p	Named Pipes – eine benannte Umleitung der Ausgabe eines Prozesses für einen anderen Prozess (siehe Kapitel 9 <i>Einführung in die Bash</i>)
s	Socket-Datei – ähnlich wie Named Pipes, aber im Netzwerk

4.5.9 Eine Übung zum Vertiefen

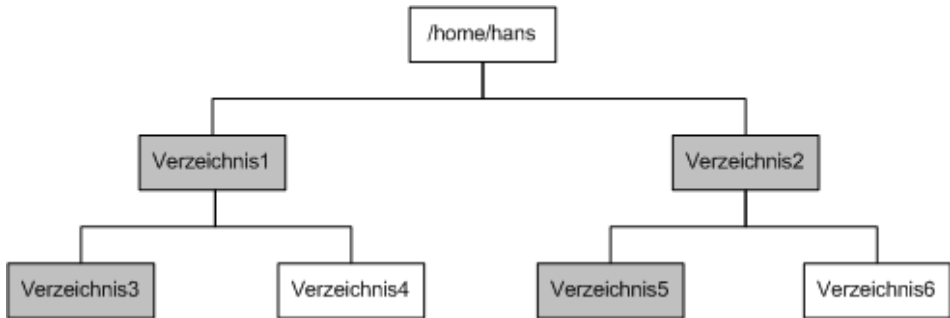
Vielleicht ist dieser ganze Abschnitt für Sie nur alter Kaffee gewesen. Erfahrungsgemäß ist es jedoch für viele Administratoren und Poweruser recht schwer, sich mit den Befehlen zur Dateimanipulation anzufreunden, wenn dies ohne Mausschuberei vor sich gehen soll (nicht persönlich nehmen – mir ging es am Anfang nicht anders!). Daher biete ich Ihnen

an dieser Stelle eine kleine Übung an, die Ihnen ein wenig Routine verschaffen wird. Erwarten Sie bitte keinen tieferen Sinn hinter den Übungsschritten, außer, dass Sie im Laufe dieses Lehrgangs immer wieder mit diesen Prozeduren zu tun haben werden.

Noch ein Tipp, bevor Sie anfangen: Ihre besten Freunde für die Übersicht auf der Konsole sind **ls**, **cd** und **pwd**. Nutzen Sie sie so oft wie möglich.

Ich unterstelle hier einen normalen Benutzer hans – ersetzen Sie hans durch ihren Benutzer, den Sie bei der Installation angelegt haben, oder erstellen Sie sich eben einen hans.

1. Melden Sie sich als hans (bzw. Ihr Benutzername) an. Erstellen Sie unter Ihrem Home-Verzeichnis `/home/hans` folgende Verzeichnisstruktur:



Nutzen Sie für die grau unterlegten Verzeichnisse auf der Konsole den Befehl **mkdir -p**, um die Verzeichnisse mit einer einzigen Befehlszeile anzulegen.

2. Erstellen Sie unter `Verzeichnis4` eine leere Textdatei namens `text1.txt`.
3. Füllen Sie diese Datei nun mit einem kurzen Text. Nutzen Sie dazu den Standardeditor.
4. Erstellen Sie nun noch eine Datei unter `Verzeichnis6` mit dem Namen `text2.txt`.
5. Geben Sie auch hier einige Zeilen ein.
6. Erstellen Sie einen Softlink in `Verzeichnis6` zu `text1.txt`. Nutzen Sie absolute Pfadangaben.
7. Kopieren Sie `text1.txt` nach `Verzeichnis6`.
8. Kopieren Sie `Verzeichnis1` mit der gesamten Verzeichnisstruktur darunter nach `Verzeichnis5`.
9. Erstellen Sie unter `Verzeichnis3` ein weiteres Verzeichnis namens `Verzeichnis7`.
10. Haben Sie noch einen Schimmer, was gerade läuft? Schauen Sie sich mal mit den oben genannten Befehlen `cd`, `ls` und `pwd` Ihre neuen Verzeichnisse und deren Inhalt an und orientieren Sie sich. ;-)
11. Verschieben Sie nun `Verzeichnis1` samt Unterverzeichnissen nach `Verzeichnis6`.
12. Überzeugen Sie sich, dass alles dort ist, wo es hingehört. Ganz schön verwirrend, was?
13. Löschen Sie anschließend die gesamte Struktur wieder. Nutzen Sie elegant den Befehl **rm -r**, aber passen Sie auf, dass Sie das richtige Verzeichnis angeben (mit **rm -r /** hatten Sie mal ein Linux-System...).

Haben Sie sich bis hierher durchgebissen, können Sie sich auf die Schulter klopfen. Sie haben es sicherlich bemerkt: Zunächst lesen sich die Befehle ganz einfach. Aber in der Realität muss man höllisch aufpassen, um die Übersicht zu behalten.

4.6 Man-Pages – Hilfe zur Selbsthilfe

Die Optionsliste mancher Befehle liest sich wie das Alphabet. Nehmen Sie nur den völlig alltäglichen Befehl **ls**. Geben Sie doch einfach mal Folgendes ein, um sich die Kurzhilfe anzeigen zu lassen:

```
# ls --help
```

Es existiert nicht nur fast jeder Buchstabe des Alphabets als Option, darüber gibt es einzelne Buchstaben auch noch in Groß- und Kleinschreibung – natürlich mit teilweise völlig unterschiedlicher Bedeutung! So, und nun lernen Sie die Liste auswendig, auf der nächsten Seite erwartet Sie ein kleines Quiz!

Bevor Sie das Buch jetzt entrüstet zur Seite legen: Das war natürlich nur ein Scherz! Aber Sie werden mir recht geben, wenn ich behaupte, dass auch echte Hardcore-Linux-Administratoren nicht alle Optionen zu allen Befehlen im Kopf haben können.

4.6.1 Die Man-Pages nutzen

Hilfe bieten hier die Man-Pages (**man** wie *manual*). Es handelt sich um (oftmals recht ausführliche) Online-Hilfeseiten zu Programmen, Programmprozeduren und Konzepten. Man-Pages sind sehr nützliche Helfer, um sich einen Überblick über ein Programm zu verschaffen oder um sich eine Option für eine bestimmte Funktion in Erinnerung zu rufen. Andererseits sind Man-Pages meines Erachtens oftmals nicht dafür geeignet, sich erstmals in einen Befehl oder ein Programm einzuarbeiten – dafür sind die Erläuterungen meistens zu abstrakt. Außerdem gibt es in vielen Man-Pages keine Beispiele, die den Einsatz eines Befehls erläutern würden.

Leider sind Man-Pages oft nur auf Englisch vorhanden, so dass Sie häufig auf Ihre Englischkenntnisse angewiesen sind. Viele Man-Pages sind aber auch schon lokalisiert, also in unserem Fall auf Deutsch, verfügbar. Auch wenn die Übersetzung vorangeht, ist dies wirklich verbesserungsfähig. Hoffentlich können Sie etwas Englisch ... ;-)

Sie rufen die Man-Page eines Befehls folgendermaßen auf:

```
# man <Befehl>
```

Versuchen Sie das zum Beispiel mit dem Kommando **mount**. Geben Sie ein:

```
# man mount
```

Die Man-Page wird durch einen Pager angezeigt, standardmäßig ist das das Programm **less**. Ist **less** nicht installiert, wird **more** genutzt. Sie können in der Man-Page mit den Cursortasten hoch- und runterscrollen oder seitenweise mit Bild auf und Bild ab – wenn Sie **less** nutzen können. Ansonsten bleibt Ihnen mit **more** nur das zeilen- und seitenweise Vorwärtsgen. Mit **q** beenden Sie die Anzeige.

```

MOUNT(8)                               Linux Programmer's Manual                               MOUNT(8)
NAME
    mount - mount a file system
SYNOPSIS
    mount [-lhU]

    mount -a [-fFnrsvw] [-t vfstype] [-O optlist]
    mount [-fnrsvw] [-o options [,...]] device | dir
    mount [-fnrsvw] [-t vfstype] [-o options] device dir
DESCRIPTION
    All files accessible in a Unix system are arranged in one big tree, the
    file hierarchy, rooted at /. These files can be spread out over sev-
    eral devices. The mount command serves to attach the file system found
    on some device to the big file tree. Conversely, the umount(8) command
    will detach it again.

    The standard form of the mount command, is
        mount -t type device dir
    This tells the kernel to attach the file system found on device (which
    is of type type) at the directory dir. The previous contents (if any)
    and owner and mode of dir become invisible, and as long as this file
Manual page mount(8) line 1

```

Abb. 4.8: Die Man-Page des Befehls mount

Sie können im Pager nach einem Begriff suchen, indem Sie `<Suchbegriff>` eingeben. Ein gefundener Begriff wird markiert. Mit `n` wird zur nächsten Fundstelle gesprungen.

Ganz oben links fällt Ihnen vielleicht auf, dass dort `MOUNT(8)` steht. Aber was bedeutet die Ziffer in Klammern hinter dem Befehl? Antwort: die Man-Pages sind nach Kategorien sortiert:

Kategorie	Sektion
1	Benutzerkommandos
2	Systemaufrufe (Systemcalls)
3	Bibliotheksaufufe
4	Spezielle Dateien (zum Beispiel Gerädateien)
5	Dateiformate der Konfigurationsdateien und Konventionen
6	Spiele
7	Makropakete und Konventionen, Diverses
8	Kommandos für die Systemadministration
9	Kernel-Routinen

Für jeden Abschnitt finden Sie ein Unterverzeichnis unter `/usr/share/man`. Die Unterverzeichnisse lauten `man1` bis `man9`. Die lokalisierten Fassungen finden sich unter einem entsprechenden Unterverzeichnis, zum Beispiel `/usr/share/man/de`.

Es kann vorkommen, dass ein Eintrag mit gleichem Namen in mehreren Sektionen auftaucht, zum Beispiel `passwd` oder `mount`. Daher wird hinter dem Namen in Klammern die jeweilige Sektion angegeben. Demnach bedeutet `mount(8)`, dass es sich um ein Kommando für die Systemadministration handelt.

Möchten Sie sich einen Eintrag innerhalb einer bestimmten Sektion anzeigen lassen, geben Sie die Sektion mit an:

```
# man 8 mount
```

Mit der Option `-a` werden Ihnen nacheinander alle Einträge angezeigt. Der folgende Befehl zeigt Ihnen (unter *Etch*) zunächst `mount(8)`, anschließend (nach Drücken von `Enter`) `mount(2)`:

```
# man -a mount
```

Warum wird Ihnen zunächst der Eintrag in Sektion 8 und danach in Sektion 2 gezeigt? Daran ist die interne Priorität schuld – Kommandos werden gegenüber System- und Bibliotheksaufrufen bevorzugt. Geben Sie also ohne Ziffer `man mount` an, wird Ihnen der Eintrag `mount(8)` angezeigt, nicht `mount(2)`.

Eventuell können Sie dies nach der Basisinstallation nicht unmittelbar nachvollziehen, da hier der Manual-Page-Eintrag für `mount(2)` fehlt. Diese ist im Paket `manpages-dev` bzw. `manpages-de-dev` (für die deutsche Version) enthalten, das Sie jederzeit nachinstallieren können.

4.6.2 whatis und apropos

Benötigen Sie lediglich eine kurze Orientierungshilfe, stehen Ihnen die beiden Befehle **whatis** und **apropos** zur Seite. Das Programm **whatis** sucht in der Man-Page-Datenbank nach dem gesuchten Begriff. Dieser muss in der eigentlichen Bezeichnung der Man-Page auftauchen. Hier ein Beispiel:

```
# whatis apt
apt (8)          - Advanced Package Tool
```



Dagegen können Sie mit **apropos** auch die Kurzbeschreibung durchsuchen lassen:


```
# apropos apt
apt (8)          - Advanced Package Tool
apt-cache (8)    - APT package handling utility - - cache manipulator
apt-cdrom (8)    - APT CDROM management utility
apt-config (8)   - APT Configuration Query program
apt-extracttemplates (1) - Utility to extract DebConf config and templates from
Debian packages
apt-ftpparchive (1) - Utility to generate index files
apt-get (8)      - APT package handling utility - - command-line interface
apt-key (8)      - APT key management utility
apt-secure (8)   - Archive authentication support for APT
apt-sortpkgs (1) - Utility to sort package index files
apt.conf (5)     - Configuration file for APT
apt_preferences (5) - Preference control file for APT
aptitude (8)     - high-level interface to the package manager
captaininfo (1)  - convert a termcap description into a terminfo description
debconf-apt-progress (1) - install packages using debconf to display a progress bar
sources.list (5) - Package resource list for APT
```

Dieser Befehl findet in der Regel mehrere Einträge, da sein Suchradius größer ist. Schauen Sie sich den letzten Eintrag an – hier befindet sich der Begriff nur in der Beschreibung, nicht im Namen der Man-Page. Andererseits dürfte `captinfo` nichts mit den APT-Tools zu tun haben.

4.6.3 info – die neuen Man-Pages

Handelt es sich um GNU-Software, existieren zusätzlich zu den traditionellen Man-Pages oder statt derer so genannte Infoseiten. Das Infosystem basiert auf den Konzepten des Hypertexts.

Zu einem Befehl, Konzept oder Systemaufruf existiert ein Hauptknoten und eventuelle Unterknoten. Zwischen diesen können Sie mit `n` (*next* – weiter) und `p` (*previous* – zurück) wechseln. Außerdem gibt es häufig Hyperlinks – zu diesen können Sie über  springen,  ruft die verlinkte Seite auf.

Außerdem gibt es manchmal Menüs, durch die Sie ebenfalls mit  wechseln können, während `m` den Menüeintrag aufruft. Mit `p` kommen Sie wieder zurück.

Mit `?` erhalten Sie eine Hilfe zur Benutzung des Infosystems. Sind Sie die Man-Pages gewöhnt, bedarf es etwas Eingewöhnungszeit, aber die Infoseiten sind in der Regel ausführlicher und daher für eine umfassende Einarbeitung nützlicher, zumal Sie mit den Hyperlinks zwischen verschiedenen Bereichen hin- und herspringen können. Ich empfehle Ihnen, an dieser Stelle ein wenig mit den Infoseiten zu üben. Mit folgendem Befehl können Sie das Angenehme mit dem Nützlichen verbinden:

```
# info info
```

Das ruft die Infoseite über das Infosystem auf – Bildung und Praxis in einem ... was will man mehr ;-).

4.7 Zusammenfassung und Weiterführendes

In diesem Kapitel haben Sie einige wichtige Grundlagen zur Nutzung Ihres Linux-Systems erlernt und können sich nun selbstständig durch das Dateisystem bewegen. Sie kennen die Dateisystemstruktur und wissen, wie Sie Textdateien manipulieren können. Da fast jede Konfigurationsdatei unter Linux als Textdatei daherkommt, ist dieses Wissen essenziell.

Sie haben darüber hinaus gelernt, wie Sie Dateien und Verzeichnisse manipulieren, sprich erstellen, löschen, verschieben, kopieren etc. können. Auch dies werden Sie immer wieder benötigen.

Schließlich haben Sie herausgefunden, wie Sie mit Hilfe der Man-Pages und Infoseiten Informationen zu Dateien, Diensten und Strukturen erhalten können.

Sie werden in diesem Buch noch sehr viele weitere Befehle kennen lernen. Jedoch sollten Sie die hier vorgestellten Kommandos beherrschen – sonst kommen Sie früher oder später in die Situation, dass Sie zwar wissen, was Sie tun müssen, aber leider nicht, wie ...

Nützliche Weblinks

Im Folgenden möchte ich Ihnen noch ein paar nützliche Websites vorstellen, die Informationen (How-tos, Man-Pages, Tutorials und Einführungen) zu verschiedenen Aspekten von Linux haben. Schauen Sie einmal auf die genannten Seiten, wenn Sie sie noch nicht kennen, und erstellen Sie einen Bookmark für sie:

<http://www.tldp.org/> - Die Seite des *Linux Documentation Project*. Hier finden Sie einen Haufen interessanter Informationen rund um Linux. Leider nur in Englisch und anderen (nicht deutschen) Sprachen.

<http://www.linuxfiel.de/> – eine Website, die eine sehr gute Einführung in viele Themen der Linux-Systemadministration bietet

<http://www.linux.org/> – Die »offizielle« Linux-Website. Auch hier finden Sie jede Menge Links zu Dokumentationen jeder Art – ebenfalls auf Englisch.

<http://www.linuxhaven.de/> – Hier verbergen sich jede Menge How-tos in Deutsch.

<http://debiananwenderhandbuch.de> – ein Onlinehandbuch, das ebenfalls zu sehr vielen Themen ausführliche Anleitungen enthält

<http://www.redhat.de/documentation/> – Das Red-Hat-Handbuch ist zwar eine Dokumentation, die auf Red Hat ausgerichtet ist, kann aber in vielen Bereichen (genau wie dieses Buch) distributionsübergreifend genutzt werden.