# Linux—Anfängerkurs

Oliver Zendel
zendel@unix-ag.uni-kl.de

Version 1.0

15. Mai 1997

**Zusammenfassung**

Dieser Kurs ist eine kurze Einführung in das Betriebssystem Linux. Da speziell Anfänger große Probleme bei ihren ersten Schritten mit Linux haben, wird hier versucht eine kleine Starthilfe zu geben. Dabei wird kurz auf die Konzepte des Betriebssystems Linux eingegangen, um dann die wichtigsten Befehle vorzustellen. Dieser Kurs entstand als Ausarbeitung für einen Vortrag zum 2. LinuxTag'97.

Ich danke Christel Marquardt für unzählige Tips, Anmerkungen sowie Korrektur- und Verbesserungsvorschläge für diesen Linux Anfängerkurs.

# Inhaltsverzeichnis


**1 Was ist Linux? 2**

**2 Grundlegende Konzepte von Linux 2**

2.1 GNU-Programme . . . 2
2.2 Multitasking . . . 3
2.3 Speicherschutz . . . 3
2.4 Multiuserkonzept . . . 3

**3 Anmelden und Abmelden 4**

3.1 Das Anmelden . . . 5
3.2 Das Abmelden . . . 6

**4 Von Verzeichnissen und Dateien 6**

4.1 Das Heimverzeichnis . . . 7
4.2 Das tmp-Verzeichnis . . . 7
4.3 Namen für Dateien . . . 7

**5 Eingabe von Befehlen 7**

5.1 Ändern des Passwortes . . . 8
5.2 Wechseln des Verzeichnisses . . . 8
5.3 Anzeigen des Verzeichnisinhalts . . . 8
5.4 Anlegen eines neuen Verzeichnisses . . . 10
5.5 Löschen eines leeren Verzeichnisses . . . 10
5.6 Kopieren von Dateien . . . 11
5.7 Verschieben von Dateien . . . 11
5.8 Löschen von Dateien . . . 12
5.9 Editieren und Betrachten von Dateien . . . 12

**6 Wo bekomme ich Hilfe? 14**

6.1 Die man pages . . . 14
6.2 Die HOWTOs . . . 15
6.3 Bücher über Linux . . . 20

# 1 Was ist Linux?

Linux ist ein Computerbetriebssystem und steht somit in direkter Konkurrenz zu MS Windows, DOS, OS/2 und anderen Betriebssystemen. Linux wird von vielen Freiwilligen auf der ganzen Welt entwickelt und vorangetrieben.

Anfangs belächelt und nicht ernst genommen, hat es sich zu einer ernstzunehmenden Alternative zu den „großen Betriebssystemen“ entwickelt. Immer mehr Softwareentwickler erkennen das Potential, das in Linux steckt und entwickeln ihre Produkte auch für Linux. So kann Linux mittlerweile auf ein beträchtliches Repertoire von Officepaketen blicken, und muß sich auch in dieser „Höhle des Löwens“ nicht mehr hinter Windows verstecken.

Ursprünglich nur für Computer mit Intel-Prozessor entwickelt, existieren heute schon ausgereifte Linuxversionen für alternative Computerplattformen, z. B.Alpha-Computer, PowerPCs und Sun-Computer. Und es wird sich zeigen, was die Zukunft bringt !

# 2 Grundlegende Konzepte von Linux

Fragt man Linuxbenutzer warum sie Linux allen anderen Betriebssystemen vorziehen, so hört man immer wieder folgende Schlagworte[1]:

- Multitasking
- Multiuser
- Speicherschutz
- GNU-Programme
- Netzwerkanbindung

Leider vergessen schwärmende Anwender viel zu oft dem meist „unbedarften Anfänger“ zu erklären, wovon sie eigentlich reden. Während **Netzwerkanbindung** noch selbsterklärend ist, es handelt sich hierbei um die Möglichkeit einen Computer über ein Netzwerk mit anderen Computern zu verbinden, sind die Begriffe **Multiuser**, **Multitasting** bzw. **Speicherschutz** nicht ohne weiteres verständlich.

## 2.1 GNU-Programme

GNU-Programme bilden das Herzstück von Linux. Sie sind unter der sogenannten „GNU public licence“ erschienen. Diese Lizenz regelt die Weitergabe und Nutzung der Programme. Alle GNU-Programme dürfen kostenlos kopiert und benutzt werden. Außerdem ist für jedes GNU-Programm der Quellcode verfügbar. Somit besteht die Möglichkeit, diese Programme zu verändern und an eigene Bedürfnisse anzupassen.

GNU-Programme haben meist ein erstaunlich hohes Niveau, hier gilt also nicht die Devise „was nichts kostet, taugt nichts“. Angespornt von den vielen guten und kostenlosen Programmen sind große Softwareentwicker dazu übergegangen, ihre Produkte für den privaten, nicht kommerziellen Gebrauch ebenfalls kostenfrei anzubieten.

[1] diese Liste könnte wahrscheinlich unendlich lang weitergeführt werden

## 2.2 Multitasking

Unter **Multitasking** versteht man die Fähigkeit des Betriebssystems, mehrere Programme „gleichzeitig“ ablaufen zu lassen. Damit hat der Benutzer die Möglichkeit, flexibler zu arbeiten. Während die Datenverarbeitung noch Datensätze sortiert, kann der Anwender z. B. Briefe mit der Textverarbeitung schreiben, und im Hintergrund behält ein Programm pausenlos die Uhr im Auge, damit der Anwender seine Verabredung nicht verpaßt.

## 2.3 Speicherschutz

Da kein Programm fehlerfrei ist, muß ein Betriebssystem eine gewisse Robustheit gegenüber Programmfehlern bzw. Programmabstürzen besitzen. Hier greift u. a. das **Speicherschutzkonzept**. Bei diesem Konzept läuft jedes Programm in seiner eigenen „Miniwelt“, aus der es nicht ausbrechen kann, um z. B. Daten von anderen Programmen mutwillig oder unfreiwillig zu zerstören.

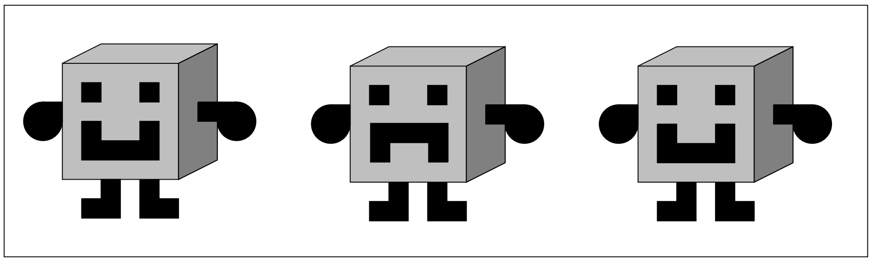

Abbildung 1: Alle Programme teilen sich den kompletten Speicher

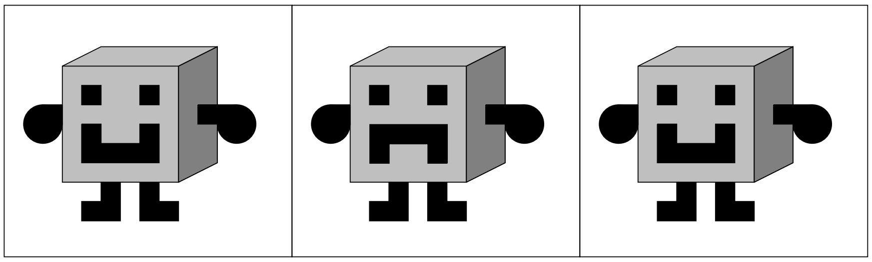

Abbildung 2: Jedes Programm befindet sich in einer „Miniwelt“

## 2.4 Multiuserkonzept

Das **Multiuserkonzept** ermöglicht es, mehrere Benutzer unabhänig voneinander zu verwalten. Dies kann von großem Nutzen sein, da jeder Anwender z. B. die Möglichkeit hat seine Arbeitsumgebung individuell zu gestalten. Auch können mehrere Benutzer „gleichzeitig“ auf einem Rechner arbeiten, wenn dieser z. B. in ein Netzwerk integriert ist.

Wenn mehrere Benutzer auf einem Rechner arbeiten, müssen Schutzmechanismen für persönliche Daten und Programme vorhanden sein. Dies ist unter Linux durch Dateirechte realisiert. Jeder Datei [2] wird ein Besitzer und eine Gruppe zugeordnet. Die komplette Welt aller Benutzer zerfällt somit in folgende drei Teile:

- Besitzer der Datei

[2] Dateien sind u. a.Programme, Daten und Verzeichnisse

- Mitglieder aus der Gruppe der Datei
- Rest der Welt

Der Besitzer einer Datei, ist derjenige dem die Daten gehören. Neu angelegte Dateien gehören demjenigen, der sie angelegt hat. Wenn Frau Testbenutzer z. B. einen Brief schreibt, so gehört ihr diese Datei. Mehrere Benutzer können zu einer Gruppe zusammengefaßt sein. In einer Firma kann man z. B. die Benutzer nach Abteilungen zusammenfassen, um bestimmte Dokumente nur für eine spezielle Abteilung lesbar zu machen, d. h. alle Mitglieder dieser Abteilung (Gruppe) dürfen die Datei lesen. Es ist auch möglich, daß ein Benutzer in mehreren Gruppen aufgeführt ist.

Alle anderen Benutzer werden dem Rest zugeordnet. Jedem dieser drei Teile (Besitzer, Gruppe, Rest) können bestimmte Dateirechte zugestanden werden:

- Lesen der Datei
- Schreiben der Datei
- Ausführen der Datei

In der Praxis werden diese Rechte meist durch einen Buchstaben abgekürzt. Dabei steht `r` (read) für Lesen der Datei, `w` (write) für Schreiben der Datei und `x` (execute) für Ausführen der Datei. Mit diesen Rechten ist es möglich seine persönlichen Daten, vor dem Zugriff anderer zu schützen. Bleibt nur die Frage zu klären, woher das Betriebssystem weiß, wer gerade die Datei lesen möchte. Diese Information ist sehr wichtig, da das Betriebssystem sonst nicht entscheiden kann, wer der Besitzer oder wer Mitglied der Gruppe der Datei ist.

# 3 Anmelden und Abmelden

Um eindeutig feststellen zu können, welcher Benutzer gerade am Computer arbeitet, muß sich jeder Benutzer vor seiner Arbeit beim Betriebssystem anmelden. Diesen Vorgang nennt man auch `login`. Jeder Benutzer eines Linuxsystems ist in einer Datenbank registriert. Das System kennt von jedem Benutzer folgende Daten:

- Benutzerkennung
- Gruppenzugehörigkeit
- Name des Benutzers

Die **Benutzerkennung** (auch oft als **Accountname, Loginname** bezeichnet) ist eine, meist auf 8 Zeichen beschränkte, Kennung des Benutzers. Diese wird bei der Einrichtung des Benutzers angegeben und in der Regel nie mehr geändert. Im Gegensatz zu der Benuzterkennung kann der **Name des Benutzers** (auch oft als **real name** bezeichnet) meistens frei vom Besitzer gewählt und jederzeit geändert werden. Dabei fällt die Beschränkung auf 8 Zeichen weg. Der Vorgang des Anmeldens wird im folgenden Beispiel erläutert:

Frau Martina Mustermann beantragt einen Zugang an einem Unixrechner.
Ihr wird folgender Zugang eingerichtet:

| | | |
|---|---|---|
| Name des Benutzers | : | Martina Mustermann |
| Kennung | : | musterma |
| Passwort | : | ab435bbn |

Es fällt auf, daß Martina Mustermann ein Passwort bekommen hat. Dies ist sehr wichtig für den Anmeldevorgang, da sonst jeder, der die Kennung von Martina Mustermann kennt, sich als sie ausgeben kann und somit ihre Daten verändern oder sonstigen Unfug in ihren Namen treiben kann. Das vom Systemadministrator[3] vorgegebene Passwort sollte unmittelbar nach dem ersten Anmelden geändert werden. Ein Passwort muß so gewählt sein, daß es niemand durch systematisches Ausprobieren erraten kann. Für die Wahl des Passwortes sollten folgende Sicherheitskriterien erfüllt sein:

- Das Passwort sollte kein Wort aus einem Wörterbuch enthalten.
- Das Passwort sollte kein Bezug zur Person zulassen.
- Das Passwort sollte Groß- und Kleinbuchstaben enthalten.

Und der wichtigste Punkt:

- Das Passwort sollte geheimgehalten werden.

Nachdem wir nun die Notwendigkeit des Anmeldens mit einem Passwort geklärt haben, können wir uns den Vorgang des Anmeldens genauer ansehen.

## 3.1 Das Anmelden

Das Anmelden läuft im Prinzip immer gleich ab. Es macht keinen Unterschied, ob man sich an einem grafischen Arbeitsplatz oder z. B über eine Netzverbindung rein textbasiert anmeldet.

Auch wenn hier nur die rein textbasierte Anmeldung besprochen wird, dürfte es keine Probleme bereiten sich grafisch anzumelden, da bis auf ein anderes Design die wesentlichen Schritte identisch sind.

Der Anmeldevorgang besteht im Prinzip nur aus der Abfrage der Kennung und des Passwortes, wobei das Passwort auf seine Richtigkeit überprüft wird.

Nachdem Martina Mustermann ihren Zugang bekommen hat, möchte sie sich natürlich sofort am Rechner anmelden. Schauen wir uns dazu mal die erforderlichen Schritte an:

```
Red Hat Linux release 4.1 (Vanderbilt)
Kernel 2.0.30 on an i686

login: _
```

Dies ist die Aufforderung die Benutzerkennung einzugeben. In unserem Beispiel wäre die Kennung musterma.

```
Red Hat Linux release 4.1 (Vanderbilt)
Kernel 2.0.30 on an i686

login: musterma
Password: _
```

Nun muß das Passwort eingegeben werden. Bei der Eingabe des Passwortes ist

[3] Systemadministratoren sind die Rechnerverwalter und –betreuer. Sie besitzen als einzige das Recht, neue User auf dem Rechner einzurichten und haben Zugang zu allen Dateien und Programmen.

meistens das eingegebene Passwort nicht auf dem Bildschirm lesbar. Dies ist aus Sicherheitsgründen wichtig, da sonst jeder Anwesende leicht das Passwort mitbekommen könnte.

Wenn das Passwort korrekt eingegeben ist, wird die Arbeitsoberfläche gestartet. Hat man sich grafisch angemeldet, so erhält man eine mehr oder weniger mit MS Windows verwandte Benutzeroberfläche. Hat man sich jedoch an einem Rechner ohne grafische Oberfläche (textbasierter Zugang) angemeldet, sieht die Arbeitsfläche eher nach einer DOS-Oberfläche aus.

### 3.2 Das Abmelden

Ebenso wichtig wie das Anmelden ist das Abmelden (oft auch logout genannt) bei Beendung der Rechnerarbeit. Vergißt man sich abzumelden, können andere den eigenen Zugang mißbrauchen, während dieser unbeaufsichtigt ist.

## 4 Von Verzeichnissen und Dateien

Linux ist dateiorientiert. Das bedeutet, daß alle Daten als Dateien abgelegt werden. Legt man viele Dateien an, so kann der Benutzer sehr schnell den Überblick verliereren. Vergleicht man die Organisation von Dateien mit Akten, so wird eine Sekretärin wohl kaum alle Akten in einen einzigen Ordner abheften. Ab einer bestimmten Anzahl von Akten wird sie verschiedene Aktenordner benutzen und die Akten sortiert darin einordnen.

Nach diesem Vorbild existieren unter Linux spezielle Dateien, sogenannte **Verzeichnisse** (Directories). Mit Hilfe dieser Verzeichnisse kann man Dateien sortiert in einer Verzeichnisstruktur ablegen.

Oft wird diese Verzeichnisstruktur auch als Verzeichnisbaum bezeichnet. Das oberste Verzeichnis nennt man Wurzelverzeichnis oder root-Verzeichnis. Dieses Verzeichnis wird mit / abgekürzt.

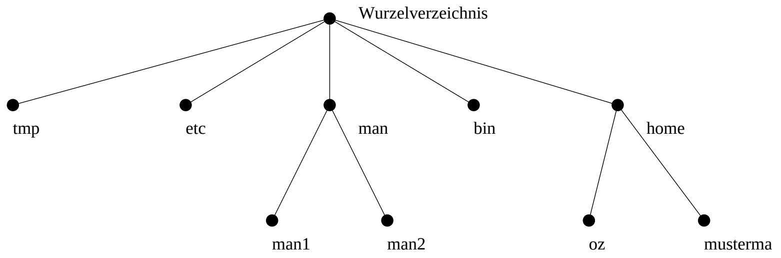


Abbildung 3: Ausschnitt eines Linuxverzeichnisbaums

Als **absoluten Pfad** bezeichnet man die komplette Angabe des Weges ausgehend vom Wurzelverzeichnis zu einem Verzeichnis. Dabei steht der oben schon vorgestellte / nicht nur für das Wurzelverzeichnis, sondern fungiert auch als Trenner zwischen den einzelnen Verzeichnissen. Der absolute Pfad für das Verzeichnis musterma wäre demnach **/home/musterma**. Neben der „absoluten Pfadangabe“ existiert noch eine „relative Pfadangabe“. Dabei wird nicht vom Wurzelverzeichnis ausgegangen, sondern vom aktuellen Verzeichnis. Dazu bezeichnet **./** das aktuelle Verzeichnis und **../** das übergeordnete Verzeichnis (ein Schritt zurück in Richtung Wurzelverzeichnis). Schauen wir uns folgende Beispiele an:

Wir befinden uns im Verzeichnis **/home/musterma** und wollen den „absoluten Pfad“ **/tmp** durch einen „relativen Pfad“ ausdrücken. Dieser wäre dann **../../tmp**.

Befindet wir uns nun im Verzeichnis **/home**, so kann der „absoluten Pfad“ **/home/musterma** auch durch den Relativen **./musterma** ausgedrücken werden.

### 4.1 Das Heimverzeichnis

Das Heimverzeichnis (home-Verzeichnis) ist das persönliche Verzeichnis eines Users. Es ist das primäre Arbeitsverzeichnis des Benutzers, in das er alle seine Daten speichert. Jeder Benutzer hat ein eigenes Heimverzeichnis, das im Verzeichnisbaum unterhalb von **/home** angeordnet ist.

### 4.2 Das tmp-Verzeichnis

Als **tmp-Verzeichnis** bezeichnet man das Verzeichnis für temporäre Daten. Der Pfad des tmp-Verzeichnis lautet meistens **/tmp**. In dieses Temporärverzeichnis können Daten abgelegt werden, für die z. B. der Platz im Heimverzeichnis nicht ausreicht. Allerdings sollte darauf geachtet werden, diese Daten wirklich nur kurzzeitig dort auszulagern, da das tmp-Verzeichnis normalerweise regelmäßig gelöscht wird.

### 4.3 Namen für Dateien

Da wir nun geklärt haben, wo die Dateien abgelegt werden können, bleibt noch die Frage offen, welche Namen die Dateien haben dürfen. Veraltete Betriebsysteme haben bei Dateinamen oft Beschränkungen:

- ☹ Beschränkung auf 12 Zeichen
- ☹ Keine Sonderzeichen erlaubt
- ☹ Keine Unterscheidung zwischen Groß- und Kleinschreibung

Unter Linux gelten diese Beschränkungen nicht, denn es gilt:

- ☺ lange Dateinamen sind erlaubt (keine Beschränkung auf 12 Zeichen)
- ☺ Sonderzeichen sind in den Dateinamen erlaubt
- ☺ Unterscheidung zwischen Groß- und Kleinschreibung

Man sollte diese Möglichkeiten nutzen und sinnvolle, lange Dateinamen verwenden. Die Zeiten der kryptischen, gekürzten Namen sind somit endlich vorbei.

## 5 Eingabe von Befehlen

Befehle werden meistens in einer **Kommandozeile** (Eingabezeile im Arbeitsfenster, shell) eingegeben. Die Kommandozeile bietet dem Anwender die Möglichkeit, Befehle über die Tastatur dem Betriebssystem mitzuteilen. Dabei reagiert die Kommandozeile interaktiv auf die Benutzereingabe. Eingaben werden immer mit der Enter-Taste abgeschlossen. Die Kommandozeile überprüft jede Eingabe auf ihren Inhalt und reagiert dementsprechent (z. B. werden Programme gestartet oder Fehlermeldungen ausgegeben).

Eine Shell ist die Standardarbeitsumgebung auf textbasierten Zugängen. Sollten Sie einen grafischen Zugang benutzen, so muß eine Shell in einem Fenster gestartet werden, um die Befehle einzugeben. Oft erhält man diese „Emulation eines Textzuganges“ durch einen Menueeintrag namens `xterm` oder `shell`.

## 5.1 Ändern des Passwortes

Das Ändern des Passwortes geschieht mit dem Befehl `passwd`. Dabei muß zuerst das aktuelle und danach das neue Passwort eingegeben werden. Sicherheitshalber muß das neue iPasswort zweimal eingegeben werden — so werden Tippfehler vermieden.

## 5.2 Wechseln des Verzeichnisses

Das **Arbeitsverzeichnis** ist das Verzeichnis, indem sich der Benutzer aktuell befindet. Nach dem Anmelden befinden sich alle User normalerweise in ihrem Heimverzeichnis. Dabei vergessen speziell Anfänger, daß jede Shell, ja sogar jedes gestartete Programm, unabhänig von allen anderem Programmen, ein eigenes Arbeitsverzeichnis besitzt.

In der Kommandozeile existiert der Befehl `cd`, um das aktuelle Arbeitsverzeichnis der Shell zu ändern. Dabei ist `cd` eine Abkürzung für den englischen Ausdruck „change directory“. Dem Befehl `cd` kann der Zielpfad als Parameter übergeben werden. Der Zielpfad kann sowohl „absolut“ als auch „relativ“ angegeben werden. Wird der Parameter weggelassen, so wird in das Heimverzeichnis des Benutzers gewechselt.

Beispiel:

```
$ pwd
/home/musterma
$ cd /tmp
$ pwd
/tmp
$ cd ..
$ pwd
/
$ cd
$ pwd
/home/musterma
$ _
```

In diesem Beispiel wurde ausgehend von dem Heimverzeichnis von Martina Mustermann in das Verzeichnis /tmp, / (das Wurzelverzeichnis) und danach wieder in das Heimverzeichnis gewechselt. Dabei wurde der Befehl `pwd` (print working directory) benutzt. Dieser Befehl gibt das aktuelle Arbeitsverzeichnis als absolute Pfadangabe aus.

## 5.3 Anzeigen des Verzeichnisinhalts

Der Inhalt des aktuellen Verzeichnisses kann mit Hilfe des Befehls `ls` angezeigt werden. Dabei ist `ls` eine Abkürzung des englischen Wortes „list“. Der einfache Aufruf von `ls` ohne Parameter zeigt Dateien des aktuellen Arbeitsverzeichnis. Es werden jedoch u. U. nicht alle Dateien angezeigt. Sogenannte Punktfiles (Dotfiles) werden standardmässig nicht angezeigt. Diese Punktfiles beginnen im Dateinamen mit einem Punkt und sind meistens Konfigurationsdateien. Wie diese „versteckten“ Konfigurationsfiles trotzdem sichtbar gemacht werden können, wird später beschrieben.

Beispiel:

```
$ ls
 Daten1        Daten2        Ordner1        Ordner2
$ _
```

In diesem Beispiel wurde mit Hilfe des Befehls `ls` der Inhalt des aktuellen Arbeitsverzeichnis ausgegeben. Demnach befinden sich im Verzeichnis vier Dateien. Über die Art der Verzeichnisse (Daten oder weitere Verzeichnisse), die Dateigröße oder die Dateirechte wird nichts ausgesagt. Dies ändert sich, wenn der Befehl `ls` mit der Option -l aufgerufen wird. Die Option bewirkt eine ausführliche, lange Ausgabe.

Beispiel:

```
$ ls -l /
 drwxr-xr-x   2   root  root  1024   Apr  13  21:00  bin/
 drwxr-xr-x   2   root  root  1024   Apr  26  16:13  boot/
 drwxrwxr-x   2   root  root  16384  May  4   22:03  dev/
 drwxr-xr-x   18  root  root  2048   May  4   22:03  etc/
 drwxr-xr-x   14  root  root  1024   Apr  10  01:09  home/
 drwxr-xr-x   4   root  root  1024   Apr  9   17:24  lib/
 drwxr-xr-x   2   root  root  12288  Apr  9   17:16  lost+found/
 drwxr-xr-x   6   root  root  1024   Apr  10  01:08  mnt/
 dr-xr-xr-x   5   root  root  0      May  4   22:03  proc/
 drwxr-xr-x   5   root  root  1024   May  2   09:46  root/
 drwxr-xr-x   3   root  root  2048   Apr  13  21:14  sbin/
 drwxrwxrwt   11  root  root  3072   May  4   22:03  tmp/
 drwxr-xr-x   20  root  root  1024   Apr  9   17:28  usr/
 drwxr-xr-x   13  root  root  1024   Apr  9   17:23  var/
$ _
```

Doch diese Informationen nutzen nichts, wenn man nicht weiß, was sie bedeuten. Betrachten wir uns dazu die erste Zeile:

```
drwxr-xr-x   2   root   root   1024   Apr   13   21:00   bin/
```

Aus der ersten Spalte kann man Informationen über die Art der Datei und die Rechte der Datei gewinnen.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| d | r | w | x | r | - | x | r | - | x |

Dabei steht an erster Stelle die Art der Datei. In unserem Beispiel zeigt das **d**, daß es sich bei der Datei um ein Verzeichnis handelt. Wäre die Datei eine Datendatei, so würde anstelle eines **d** ein **-** stehen. Außer **d** oder **-** gibt es noch andere Dateitypen, auf die hier allerdings nicht näher eingegangen wird.

Die neun folgenden Stellen geben Aufschluß über die Zugriffsrechte (vgl. Multiuserkonzept) der Datei. Dabei werden die neun Stellen zu je drei Stellen zusammengefaßt. Die ersten drei Stellen stehen für die Zugriffsrechte des Besitzers, die zweiten für die Gruppe und die dritten Ziffern für den „Rest der Welt“.

Die nächste Spalte gibt spezielle Informationen über die Datei. Der Anfänger kann getrost über diese Spalte hinwegsehen, da sie wenig Informationen für ihn bereitstellt.

Aus den folgenden zwei Spalten kann der Besitzer und die Gruppe der Datei ermittelt werden. Dabei bedeutet

| | |
|---|---|
| root | root |

daß diese Datei dem Benutzer „root“ und der Gruppe „root“ gehört.

Nun haben wir eine Liste von allen Dateien bis auf die „versteckten Dateien“. Diese sogenannten „Dotfiles“ können mit der Option -a angezeigt werden. Wird die Option -a verwendet, so wird bei der Ausgabe des Verzeichnisinhalts auch die Verzeichnisse **.** und **..** angezeigt (diese beiden Verzeichnisse existieren in jedem Verzeichnis). Die Optionen -a und -l können auch kombiniert angegeben werden.

Wird dem Befehl `ls` als Option ein Pfad (absolut oder relativ) angegeben, so wird nicht der Inhalt des aktuellen Arbeitsverzeichnisses, sondern der des Pfades angezeigt. Gibt man auf der Kommandozeile z. B. den Befehl `ls -la /` eingegeben, so wird unabhänig von dem aktuellen Arbeitsverzeichnis, der Inhalt des Wurzelverzeichnis ausgegeben (siehe Beispiel).

## 5.4 Anlegen eines neuen Verzeichnisses

Neue Verzeichnisse können mit dem Befehl `mkdir` (make directory) angelegt werden. Als Option wird der Pfad (relativ oder absolut) des neuen Verzeichnis angegeben. Dies kann jedoch nur geschehen, wenn man Schreibrechtigung für das Verzeichnis hat, indem das neue Verzeichnis angelegt werden soll. Das Verzeichnis **/bin/test** kann nur dann angelegt werden, wenn man Schreibberechtigung für das Verzeichnis **/bin** besitzt ( Schreibberechtigung für das Verzeichnis **/bin** besitzt nur der Benutzer root und die Gruppe root).

Beispiel:

```
$ pwd/
/home/musterma
$ ls
  Daten1        Daten2        Ordner1
$ mkdir ./Ordner2
$ ls
  Daten1        Daten2        Ordner1        Ordner2
$ _
```

## 5.5 Löschen eines leeren Verzeichnisses

Leere Verzeichnisse werden mit dem Befehl `rmdir` gelöscht werden. Dieser Befehl löscht ein Verzeichnis aus dem Verzeichnisbaum. Dies kann nur geschehen, wenn das Verzeichnis leer ist, das bedeutet, in dem zu löschenden Verzeichnis dürfen sich nur die Dateien **.** und **..** befinden.

```
Beispiel:

$ pwd
/home/musterma
$ ls
  Daten1          Daten2          Ordner1          Ordner 2
$ ls ./Ordner2
$ rmdir ./Ordner2
$ ls
  Daten1          Daten2          Ordner1
$ _
```

Wie nichtleere Verzeichnisse gelöscht werden können, wird später vorgestellt.

## 5.6 Kopieren von Dateien

Das Kopieren von Dateien geschieht mit Hilfe des Befehls `cp` (copy). Dabei wird die zu kopierende Datei vervielfacht, d. h. sie existiert nach dem Kopiervorgang mehrfach. Um den Kopiervorgang durchführen zu können, müssen die zu kopierenden Dateien lesbar sein und das Verzeichnis, in das kopiert wird, muß schreibar sein.

```
Beispiel:

$ pwd
/home/musterma
$ ls
  Daten1          Ordner1          Ordner 2
$ cp ./Daten1 ./Daten2
$ ls
  Daten1          Daten2          Ordner1          Ordner 2
$ cp ./Daten1 ./Daten2 ./Ordner1
$ ls ./Ordner1   Daten1          Daten2
$ _
```

Werden wie im Beispiel mehrere Dateien kopiert, muß als Zieldatei ein Verzeichnis angegeben werden.

## 5.7 Verschieben von Dateien

Sollen Dateien nicht kopiert, sondern nur an eine andere Stelle des Verzeichnisbaumes verschoben werden, so geschieht dies mit dem Befehl `mv` (move). Der Befehl `mv` verhält sich ähnlich wie der Befehl `cp`. Im Vergleich dazu, muß jedoch nicht nur für die Zielverzeichnis Schreibberechtigung vorliegen, sondern auch für das Verzeichnis, indem die zu verschiebenden Dateien liegen. Wird als Zieldatei ein Pfad zu einem Verzeichnis angegeben, so wird die Datei in das neue Verzeichnis „verschoben“.

```
$ mv ./datei1 /tmp
```

Der Dateiname **datei1** wird beibehalten, da nur der Pfad zu einem Verzeichnis als Zieldatei angegeben worden ist. Soll die Datei **datei1** aus dem aktuellen Arbeitsverzeichnis nach **/tmp** verschoben werden, allerdings dort unter dem Namen **daten_unwichtig** abgelegt werden, so kann das folgendermaßen geschehen:

```
$ mv ./datei1 /tmp/daten_unwichtig
```

Werden mehrere Dateien verschoben, so muß (vergl. Befehl `cp`) als Zielverzeichnis ein Verzeichnis angegeben werden. Dabei können die Dateien nur verschoben werden und behalten ihren Dateinamen bei.

```
$ mv ./datei1 ./datei2 /tmp
```

## 5.8 Löschen von Dateien

Das Löschen von Dateien geschieht mit Hilfe des Befehls `rm` (remove). Dem Befehl `rm` können eine oder mehrere Dateien übergeben werden, die dann physikalisch gelöscht werden. Man beachte, daß die Dateien theoretisch unwiederbringbar gelöscht sind. Praktisch ist nicht mit einem totalen Datenverlust zu rechnen, da bei vielen Computersystemen in regelmäßigen Abständen Sicherheitskopien des Datenbestandes gemacht wird. Im schlimmsten Fall fehlen also nur die Daten bis zum letzten Zeitpunkt der Datensicherung. Ob Sicherheitskopien angelegt werden und wie man sie zurücklesen kann, kann nur der zuständige Systemverwalter (root) beantworten.

```
$ rm ./datei1 ./datei2
```

Bei diesem Beispiel werden die Dateien **datei1** und **datei2** im aktuellen Arbeitsverzeichnis gelöscht. Der Löschvorgang kann nur durchgeführt werden, wenn man Schreibberechtigung für die Dateien besitzt.

Um komplette Verzeichnisse mitsamt ihrem Inhalt löschen zu können, kann man dem Befehl `rm` die Option -r verwenden. Die Option bewirkt, daß das Verzeichnis rekursiv gelöscht wird, also erst alle Dateien des Verzeichnisses und dann das Verzeichnis selbst.

```
$ rm -r ./Ordner1
```

Bei diesem Beispiel wird im aktuellen Arbeitsverzeichnis das Verzeichnis `Ordner1` mitsamt seinem Inhalts gelöscht. Dabei sollte man sich im Klaren sein, welche Dateien sich im Verzeichnis **Ordner1** befinden, sonst können schnell Daten gelöscht werden, die man eigentlich noch behalten wollte.

## 5.9 Editieren und Betrachten von Dateien

Texteditoren für Linux gibt es unzählig viele. Für jeden ist ein passender darunter, man muß sie nur alle mal ausprobieren. Sie alle eignen sich mehr oder weniger gut zum Erstellen von Textdateien. Textdateien sind Dateien, deren Inhalt ein Text ist. Die gebräuchlichsten Editoren sind:

- emacs

- joe
- pico
- jed
- vi
- axe

Wobei `axe` nur bei einem grafischen Zugang, nicht bei einem rein textbasierten Zugang genutzt werden kann.

Zum Anzeigen des Inhaltes einer Textdatei muß nicht unbedingt ein Editor aufgerufen werden. Unter Linux stehen eine Reihe von Befehlen zur Verfügung, die den Inhalt einer Datei anzeigen, nicht aber das Editieren der Datei erlauben. Diese Trennung ist sinnvoll, da bei diesen „Dateibetrachtern" nicht (aus Versehen) der Inhalt einer Datei geändert werden kann. Ein so veränderter Inhalt muß dann mühsam wieder restauriert werden.

Mit dem Befehl `cat` wird der Inhalt der Datei auf dem Bildschirm ausgegeben. Es werden keinerlei Benutzerinteraktion zugelassen, d. h. der Benutzer hat keine Möglichkeit aktiv einzugreifen. Umfaßt die Datei 1000 Zeilen Text, so werden diese in einem Stück ausgegeben, ohne daß der Benutzer die Möglichkeit hat diese zu lesen. Wozu braucht man dann überhaupt diesen Befehl? Eine sinnvolle Anwendung des Befehls `cat` ist z. B. das Konkatenieren (Hintereinangerhängen, Zusammenfügen) von mehreren Dateien. Es spielt dabei keine Rolle ob es sich um Textdateien oder sonstige Datendateien handelt.

```
$ cat ./Datei1 ./Datei2 ./Datei3 > ./Gesamtdatei
```

Dabei gibt `cat Datei1 Datei2 Datei3` den Inhalt der Dateien auf dem Bildschirm aus. Sollte es sich dabei nicht um reine Textdateien handeln, können wirre Zeichenkombinationen auf dem Bildschirm ausgegeben werden. Diesen Vorgang nennt man **Umleitung**. Eine Umleitung leitet die Ausgabe eines Befehles in eine Datei um. So wird in unserem Beispiel der Inhalt der Dateien nicht auf dem Bildschirm ausgegeben, sondern in die Datei mit dem Namen **Gesamtdatei** geschrieben. Dazu muß folgendes bedacht werden:

1. Die Datei muß für den Benutzer schreibbar sein bzw. das Verzeichnis in dem die Datei neu angelegt werden soll muß schreibbar sein.

2. Der aktuelle Inhalt der Datei **Gesamtdatei** geht verloren, da er durch die neuen Daten überschrieben wird (evt. alte Datei vorher kopieren).

Eine weitere häufige Anwendung des Befehls `cat` ist die Verwendung in einem sogenannten **Pipe-Konstrukt**:

```
$ Befehl1 | Befehl2 | Befehl3
```

Dadurch werden zwei oder mehrere Befehle zu einer mit | getrennter Kette zusammengefügt. Die Ausgaben des links des | stehenden Befehls werden als Eingaben des rechts stehenden Befehls ausgewertet. In unserem abstrakten Beispiel bedeutet dies folgendes: Der Befehl **Befehl1** wird ausgeführt und produziert eine Ausgabe. Diese wird als Eingabe für den Befehl **Befehl2** genommen. Mit dieser Eingabe produziert **Befehl2** eine Ausgabe, die als Eingabe des letzten Befehls in der Kette genommen wird. Die Ausgabe von `Befehl3` ist die Ausgabe, die dem Benutzer auf seinem Bildschirm dargestellt wird. Der Benutzer bekommt nur die Ausgabe des letzten Befehls der Kette, alle anderen Ausgaben werden nicht zusätzlich auf dem Bildschirm ausgegeben. Stellt man sich die Aufgabe, den Inhalt einer Textdatei so darzustellen, daß man seitenweise vor- und zurückblättern kann, so könnte das folgendermaßen aussehen.

```
$ cat ./Datei1 | less
```

Der Befehl `less` dient wie `cat` dem Darstellen des Dateiinhaltes, der Befehl ist jedoch wesentlich komfortabler. In unserem Beispiel bekommt der Befehl `less` als Eingabe die Ausgabe von `cat ./Datei1`. Der Befehl `less` kümmert sich nun um die Aufarbeitung des Textes. Dazu zählt u. a. das freie Bewegen im Text und das Suchen nach Schlüsselwörtern. Unsere obige Konstruktion ist etwas umständlich, demonstriert aber anschaulich die Verwendung des Befehls `cat` in Verbindung mit einer Pipe. Viel eleganter und kürzer erhält man dasselbe Ergebnis, wenn man dem Befehl `less` direkt die Datei übergibt.

```
$ less ./Datei1
```

Werden mehrere Dateien angegeben, so kann zwischen diesen Dateien gewechselt werden.

# 6 Wo bekomme ich Hilfe?

## 6.1 Die man pages

**man pages** bilden das Herzstück der Linux Dokumentation. Sie bieten u. a. Beschreibungen von Programmen, Konfigurationsdateien und C-Funktionen. Dabei sollte allerdings erwähnt werden, daß **man pages** sich gut eignen um Optionen und Funktion von Programmen nachzuschlagen, sich allerdings weniger gut eignen um Konzepte zu verstehen bzw. Programmiersprachen zu erlernen. Zugriff zu den **man pages** bietet das Programm `man`.

```
$ man man
```

Mit Hilfe dieses Beispiels wird die **man page** des Programms `man` angezeigt. Es kann auch zu jedem anderern hier vorgestellten Befehl die Hilfeseite angezeigt werden. So gibt

```
$ man ls
```

die Hilfeseite zum Befehl `ls` aus. Neben den hier vorgestellten Optionen für `ls`, erhält man auch eine Beschreibung für die vielen hier nicht vorgestellten Optionen.

## 6.2 Die HOWTOs

Die **HOWTOs** geben mehr Auskunft über spezielle Probleme unter Linux. Sie beschreiben detaliert die Vorgehensweise, um Programme zu konfigurieren und anzupassen, oder Erklären bestimmte Sachverhalte ausfürlich. Bisher existieren folgende HOWTOs:

- AX25-HOWTO
- Access-HOWTO
- BootPrompt-HOWTO
- Bootdisk-HOWTO
- Busmouse-HOWTO
- CDROM-HOWTO
- Commercial-HOWTO
- Consultants-HOWTO
- Cyrillic-HOWTO
- DNS-HOWTO
- DOSEMU-HOWTO
- Danish-HOWTO
- Distribution-HOWTO
- ELF-HOWTO
- Emacspeak-HOWTO
- Ethernet-HOWTO
- Finnish-HOWTO
- Firewall-HOWTO
- Ftape-HOWTO
- GCC-HOWTO
- German-HOWTO
- HAM-HOWTO
- HOWTO-INDEX
- Hardware-HOWTO
- Hebrew-HOWTO
- IPX-HOWTO
- ISP-Hookup-HOWTO
- Installation-HOWTO
- Italian-HOWTO

- Java-CGI-HOWTO
- Java-HOWTO
- Kernel-HOWTO
- Keyboard-HOWTO
- Linux-HOWTOs.tar
- MGR-HOWTO
- Mail-HOWTO
- Module-HOWTO
- NET-2-HOWTO
- NIS-HOWTO
- News-HOWTO
- PCI-HOWTO
- PCMCIA-HOWTO
- PPP-HOWTO
- Polish-HOWTO
- Portuguese-HOWTO
- Printing-HOWTO
- Printing-Usage-HOWTO
- SCSI-HOWTO
- SCSI-Programming-HOWTO
- SMB-HOWTO
- Serial-HOWTO
- Shadow-Password-HOWTO
- Slovenian-HOWTO
- Sound-HOWTO
- Sound-Playing-HOWTO
- Spanish-HOWTO
- Term-HOWTO
- Tips-HOWTO
- UMSDOS-HOWTO
- UPS-HOWTO
- UUCP-HOWTO
- WWW-HOWTO

- XFree86-HOWTO

und Mini-HOWTOs:

- 
- 3-Button-Mouse
- ADSM-Backup
- AI-Alife
- Advocacy
- Assembly
- Backup-With-MSDOS
- Boca
- BogoMips
- Bridge
- Bridge+Firewall
- CD-Writer
- Colour-ls
- Consoles
- Consoles-Many
- DHCP
- DOS2Linux
- Diald
- Dip+SLiRP+CSLIP
- Diskless
- Dynamic-IP-Hacks
- GTEK-BBS-550
- GUI-Development
- Graphics-Tools
- Gravis-UltraSound
- HTML-Validation
- HTTP+Netware
- INDEX
- INDEX.html
- IO-Port-Programming
- IP-Alias

- IP-Masquerade
- JE
- Java-WorkShop
- Jaz-Drive
- Kerneld
- Key-Setup
- Keystroke
- LBX
- LF1000
- LILO
- Large-Disk
- Linux+DOS+Win95
- Linux+DOS+Win95+OS2
- Linux+OS2+DOS
- Linux+Win95
- Linux+WinNT
- Linux-mini-HOWTOs.tar.gz
- Locales
- MIDI+SB
- Mail-Queue
- Mail2News
- Man-Page
- Modeline
- Multiple-Disks-Layout
- Multiple-Ethernet
- NFS-Root
- NFS-Root-Client
- Netscape+Proxy
- Online-Support
- PLIP
- PPP-over-ISDN
- Pager
- Partition

- Print2Win
- Process-Accounting
- Proxy-ARP
- Quota
- README
- Reading-List
- Remote-Boot
- SLIP+proxyARP
- Sendmail+UUCP
- Swap-Space
- TIA
- Term-Firewall
- Tiny-News
- Token-Ring
- Upgrade
- Virtual-Web
- Virtual-wu-ftpd
- Visual-Bell
- Win95+Win+Linux
- WordPerfect
- X-Big-Cursor
- X-Notebook
- XFree86-XInside
- Xterm-Title
- Xterminal
- ZIP-Drive
- ZIP-Install

## 6.3 Bücher über Linux

Bisher sind eine Reihe von Büchern über Linux erschienen. Hier ist eine kleine Auswahl von Bücher für Linux in deutscher Sprache.

- **LinuX Anwenderhandbuch**
  von Hetze, Sebastian; Hohndel, Dirk; Müller, Martin; Kirch Olaf u.a. LunetIX Softflair, ISBN: 3-929764-05-9
  http://www.lunetix.de/docs/Ver6.0/

- **Linux — Wegweiser zur Installation & Konfiguration**
  von Matt Welsh, Lar Kaufmann
  O'Reilly, ISBN: 3-930673-19-3
  http://www.lunetix.de/docs/IGS/