

# Einführung in UNIX

Computerlinguistik der Universität Erlangen  
Martin Lorenz / Jörg Kapfer / Matthias Bethke

25.4.2005

## Contents

<b>1</b>	<b>Das Unix-Dateisystem</b>	<b>2</b>
1.1	Dateien und Verzeichnisse . . . . .	2
1.2	Pfade . . . . .	2
1.2.1	Relative Pfadangabe . . . . .	2
1.2.2	Absolute Pfadangabe . . . . .	3
1.2.3	Beispiele . . . . .	4
1.3	Die Tilde (~) . . . . .	4
<b>2</b>	<b>Die Kommandozeile</b>	<b>5</b>
2.1	Grundlagen . . . . .	5
2.2	Optionen . . . . .	6
2.3	Wildcards . . . . .	6
2.4	Vorder- und Hintergrundprozesse . . . . .	7
2.5	Umlenken der Ein- und Ausgabe . . . . .	7
2.6	Verkettung von Kommandos . . . . .	9
2.7	Online-Dokumentation des Unix-Systems . . . . .	9
<b>3</b>	<b>Wichtige Unix-Befehle</b>	<b>9</b>
3.1	Arbeitsverzeichnis wechseln — cd . . . . .	9
3.2	Inhalt eines Verzeichnisses auflisten — ls . . . . .	10
3.3	Zugriffsrechte einer Datei ändern — chmod . . . . .	11
3.4	Textdatei seitenweise anzeigen — less . . . . .	12
3.5	Verzeichnisse anlegen und entfernen — mkdir, rmdir . . . . .	12
3.6	Kopieren von Dateien — cp . . . . .	13
3.7	Entfernen von Dateien — rm . . . . .	14
3.8	Umbenennen von Dateien und Verzeichnissen — mv . . . . .	15
3.9	Kennwort ändern — passwd . . . . .	15
3.10	Dateien ausdrucken — lp . . . . .	15

3.11	Dateien archivieren — tar . . . . .	16
3.12	Dateien komprimieren — gzip . . . . .	16
3.13	Verschicken von Dateien — mail . . . . .	17
3.14	Verketteten von Dateien — cat . . . . .	17
3.15	Umwandeln des Textformats — latex, dvips . . . . .	17

# 1 Das Unix-Dateisystem

## 1.1 Dateien und Verzeichnisse

Bevor man Unix-Befehle anwenden kann, muß man sich erst einmal klargeworden sein, wie die Daten, mit denen man arbeiten will, organisiert sind. Dazu ist es wichtig, Dateien und Verzeichnisse auseinanderzuhalten:

- **Dateien** enthalten die eigentlichen Informationen. Es gibt z.B. ausführbare Dateien (*Programme*), die Anweisungen für den Rechner enthalten, oder Textdateien, die für den Menschen lesbaren Text speichern.
- **Verzeichnisse** können beliebig viele Dateien oder Verzeichnisse enthalten, speichern aber selbst keine Informationen.

Bildlich gesprochen kann man eine Datei mit einem Buch vergleichen, das in Form seines Textes Information speichert. Im Gegensatz dazu ist ein Verzeichnis eher ein Schrank, der seinerseits Bücher (*Dateien*) oder Schubladen (*Unterverzeichnisse*) enthält.

Alle Dateien und Verzeichnisse eines Dateisystems stehen miteinander in Beziehung. Diese Tatsache läßt sich am ehesten durch einen Baum beschreiben, der auf dem Kopf steht. In Abb. 1 ist ein Teil eines Dateisystems dargestellt. Alle Kästchen stehen für Verzeichnisse, Dateien sind nicht eingerahmt. Ganz oben ist die sogenannte Wurzel (*root*), die einige Unterverzeichnisse enthält. Diese können natürlich auch wieder Dateien oder weitere Unterverzeichnisse enthalten. Daraus ergibt sich eine baumähnliche Gestalt. Dateien können natürlich nicht innerhalb eines Baumes, sondern nur an dessen Blättern (also an den unteren Enden) existieren, da eine Datei keine anderen Dateien oder Verzeichnisse enthalten kann.

## 1.2 Pfade

Ein Benutzer kann sich immer nur in *einem* Verzeichnis befinden. Dieses nennt man *Arbeitsverzeichnis*. Von dort aus kann er aber jede Datei und jedes Verzeichnis des Dateisystems eindeutig bezeichnen, indem er den *Zugriffspfad* mitangibt, auf dem man zum gewünschten Objekt kommt. Dabei gibt es zwei Möglichkeiten, die im folgenden dargestellt werden.

### 1.2.1 Relative Pfadangabe

Die relative Pfadangabe hat als **Ausgangspunkt immer das Arbeitsverzeichnis**. Um den Pfad richtig anzugeben, ist es sinnvoll, sich den Dateibaum vorzustellen und nach folgenden Regeln vorzugehen:

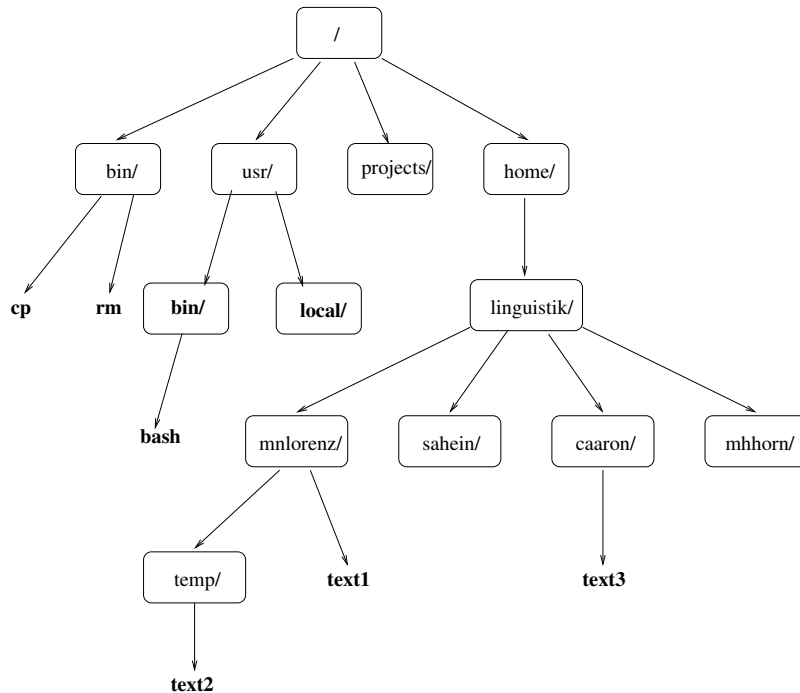


Figure 1: Ausschnitt aus einem Unix-Dateibaum

1. Zuerst sucht man im Dateibaum die Verbindung zwischen dem Arbeitsverzeichnis und dem gewünschten Objekt.
2. Danach wandert man vom Arbeitsverzeichnis aus auf dieser Verbindung immer ein Verzeichnis weiter. Dabei gibt es zwei Möglichkeiten:
  - (a) Das nächste Verzeichnis liegt weiter oben im Dateibaum: Dann muß die Pfadangabe um “..” ergänzt werden.  
**Hinweis:** Die zwei Punkte stehen für das übergeordnete Verzeichnis. Im Gegensatz dazu steht *ein* Punkt für das aktuelle Verzeichnis.
  - (b) Das nächste Verzeichnis liegt weiter unten im Dateibaum: Dann muß die Pfadangabe um den Namen des nächsten Verzeichnisses gefolgt von einem Schrägstrich (*Slash*) ergänzt werden, also z.B. “verzeichnis/”.
3. Ist das gewünschte Objekt eine Datei, hängt man deren Namen noch an den Pfad an.

Eine relative Pfadangabe ist immer dann vorteilhaft, wenn das Arbeitsverzeichnis und die gewünschte Datei bzw. das gewünschte Verzeichnis im Baum über einen kurzen Pfad zu erreichen sind.

### 1.2.2 Absolute Pfadangabe

Im Gegensatz zur relativen Pfadangabe ist bei der absoluten Adressierung der *Ausgangspunkt immer das Wurzelverzeichnis*, also ganz oben. Deshalb besteht nie die Notwendigkeit, durch “..” im Dateibaum nach oben zu wandern. Auch hier kann man schematisch nach folgenden Regeln vorgehen:

1. Zuerst sucht man im Dateibaum die Verbindung zwischen dem Wurzelverzeichnis und dem gewünschten Objekt.
2. Nun beginnt man die Pfadangabe mit einem “/”. Durch dieses Zeichen wird kenntlich gemacht, daß der Pfad als absolut interpretiert werden soll.
3. Danach wandert man von der Wurzel aus auf dieser Verbindung immer ein Verzeichnis weiter nach unten und ergänzt die Pfadangabe jeweils um den Namen des nächsten Verzeichnisses, gefolgt von einem Schrägstrich, also z.B. “verzeichnis/”.
4. Ist das gewünschte Objekt eine Datei, hängt man deren Namen noch an den Pfad an.

Die absolute Pfadangabe wird meist dann verwendet, wenn die Wurzel über einen kürzeren Pfad mit dem gewünschten Objekt verbunden ist als das Arbeitsverzeichnis, oder wenn ein Verzeichnis unabh"angig vom aktuellen Verzeichnis angesprochen werden soll (z.B. in einem *Shellskript*). Bei der Wahl zwischen absoluter und relativer Adressierung geht es also meistens darum, Tipparbeit zu sparen. Ansonsten sind beide Arten gleichwertig.

**Anmerkung:** Wenn das gewünschte Objekt ein Verzeichnis ist, kann der abschließende Schrägstrich auch weggelassen werden.

### 1.2.3 Beispiele

Die folgenden Beispiele beziehen sich alle auf Abb. 1.

AV steht für Arbeitsverzeichnis.

AV	Ziel	relativer Pfad	absoluter Pfad
bin	cp	cp	/bin/cp
projects	cp	../bin/cp	/bin/cp
linguistik	cp	../../bin/cp	/bin/cp
mnlorenz	text2	temp/text2	/home/linguistik/mnlorenz/temp/text2
sahein	text1	../mnlorenz/text1	/home/linguistik/mnlorenz/text1
mhhorn	bin/	../../bin/	/bin/

Die ersten drei Beispiele zeigen deutlich, daß absolute Pfadangaben unabhängig vom Arbeitsverzeichnis immer gleich sind. Dies ist leicht zu erklären, denn der Ausgangspunkt ist immer das Wurzelverzeichnis.

## 1.3 Die Tilde (~)

Eine besondere Bedeutung bei Pfadangaben hat die Tilde. Mit ihr läßt sich einige Schreibarbeit sparen, denn sie steht für das Homeverzeichnis des jeweiligen Benutzers. Es ist also möglich, die relativ lange Pfadangabe des Homeverzeichnisses durch ein einziges Symbol — nämlich ~ — zu ersetzen.

**Beispiel:** Angenommen, der Benutzer *mnlorenz* möchte auf das Verzeichnis *temp* in seinem Homeverzeichnis zugreifen. Dann sind folgende Ausdrücke gleichwertig:

```
/home/linguistik/mnlorenz/temp
~/temp
```

Auch auf die Homeverzeichnisse anderer Benutzer kann man mit der Tilde sehr einfach zugreifen, indem man den Namen des Homeverzeichnisses des jeweiligen Benutzers an die Tilde anhängt.

**Beispiel:** Wenn ein anderer Benutzer das Verzeichnis *temp* im Homeverzeichnis von *mnlorenz* ansprechen will, hat er folgende Möglichkeiten:

```
/home/linguistik/mnlorenz/temp  
~mnlorenz/temp
```

**Achtung:** Diese Beispiele waren noch keine Befehle, sondern lediglich Möglichkeiten, wie man Pfade angibt! Ein ausführbares Kommando dagegen startet immer mit einem Unix-Befehl, der gegebenenfalls Pfade als Argumente akzeptiert. So kann man z.B. mit

```
cd ~msbethke/temp
```

in das Verzeichnis *temp* des Benutzers *mnlorenz* springen.

## 2 Die Kommandozeile

### 2.1 Grundlagen

Die Anzeige

```
fdfeuers@clue49:~>
```

wird als *Prompt* oder *Eingabeaufforderung* bezeichnet. Durch den Prompt wird dem Benutzer/der Benutzerin angezeigt, daß der Rechner auf eine Eingabe wartet. Man kann nun Buchstaben, Zahlen und Sonderzeichen eingeben. Sie werden auf dem Bildschirm dargestellt, und der blinkende, buchstabengroße Block, die Eingabemarkierung (engl. *Cursor*), wandert nach rechts. Sonst aber tut der Computer nichts. Erst wenn Sie einen vollständigen Unix-Befehl eingegeben haben und die RETURN-Taste drücken, versucht der Computer, den Befehl zu verstehen und auszuführen. Anders als MS-DOS-Systeme unterscheidet Unix zwischen Groß- und Kleinschreibung; normalerweise werden Unix-Befehle klein geschrieben.

Ein Unix-Befehl fängt immer mit einem Befehlsnamen an. Das heißt, die Syntax der Shell erlaubt an der ersten Stelle nur zwei Arten von Wörtern: Entweder einen Befehl, den die Shell selbst bereitstellt, oder eine ausführbare Datei. Wenn irgendein anderes Wort am Zeilenanfang steht (z.B. eine nicht ausführbare Datei), erhält man eine Fehlermeldung. Später werden die (für den Anfang) wichtigsten Unix-Befehle noch genauer erklärt.

**Beispiel:**

```
$ pwd (RETURN)
```

gibt den absoluten Pfadnamen des Arbeitsverzeichnisses aus. Der Befehlsname ist eine Abkürzung für *Print Working Directory*, 'drucke Arbeitsverzeichnis'.

Hinter dem Befehlsnamen können weitere Wörter folgen, die sich in zwei Klassen einteilen lassen: Argumente und Optionen. Durch Drücken der Tab-Taste kann ein angefangenes Wort

automatisch vervollständigt werden, wenn die Shell die Fortsetzung „erraten“ kann. Bei der Eingabe des Befehls „crecord“, z.B., genügt es, „cdre“ einzugeben, und die Tab-Taste zu drücken. Da es (gegenwärtig) keinen anderen Unix-Befehl gibt, der mit „cdre“ beginnt, bleibt nur noch „cdrecord“ übrig. Die Vervollständigung ist von der Position des Cursors abhängig, d.h. das erste Wort in der Zeile wird immer zu einem Befehlsnamen vervollständigt, alle weiteren Wörter als Datei- oder Verzeichnisnamen.

Argumente geben z.B. an, auf welche Dateien oder Verzeichnisse sich der Befehl bezieht. Optionen modifizieren das Verhalten des Befehls. Argumente und Optionen müssen durch mindestens ein Leerzeichen vom Befehlsnamen abgetrennt werden. Der Befehl

```
$ cd .. (RETURN),
```

*Change Directory*, ‘wechsle Verzeichnis’, z.B. nimmt das Argument ‘.’ (das ist das Verzeichnis, das das Arbeitsverzeichnis enthält) und macht dieses Verzeichnis zum neuen Arbeitsverzeichnis.

## 2.2 Optionen

Hinter dem Befehlsnamen (und vor den Argumenten, falls vorhanden) dürfen bei vielen Befehlen noch sogenannte *Optionen* stehen, die die Arbeitsweise eines Befehls noch genauer steuern können. Eine Option beginnt mit ‘-’, hinter dem üblicherweise mindestens ein Buchstabe steht.

Beispiel: Der Befehl *ls* (für *list*, ‘auflisten’), gibt alle Dateien und Verzeichnisse aus, die im Arbeitsverzeichnis stehen. Geben Sie nur

```
$ ls
```

ein, dann werden nur die Namen der Dateien und Verzeichnisse ausgegeben. Geben Sie

```
$ ls -l
```

ein, dann gibt der Computer genauere Informationen zu jeder Datei aus.

Es gibt aber auch Optionen, die nicht nur aus einem Buchstaben bestehen, sondern aus einem Wort. Diese werden mit zwei Bindestrichen eingeleitet, z.B. werden bei *ls --all* auch die versteckten Dateien angezeigt, also die, die mit einem Punkt beginnen.

Zu den meisten Befehlen gibt es sehr viele Optionen, die man bei Bedarf benutzen kann. Von diesen muß man sich natürlich nur die wichtigsten merken; den Großteil schlägt man bei Bedarf in Büchern nach oder benutzt die Online-Hilfe **man**.

## 2.3 Wildcards

Viele Befehle können auf mehrere Dateien gleichzeitig angewendet werden. Hierbei helfen die sogenannten *Dateimuster* (engl. *wildcards*). Hier ein Beispiel: *Text?.tex* steht für jeden Dateinamen, der mit ‘Text’ anfängt, dann ein beliebiges Zeichen enthält, und mit ‘.tex’ aufhört. Die wichtigsten Sonderzeichen für Dateimuster sind:

- ‘?’ Steht für ein beliebiges Zeichen im Dateinamen.

- ‘[abde]’ Steht für eines der Zeichen a, b, d oder e.
- ‘\*’ Steht für eine beliebig lange Folge beliebiger Zeichen im Dateinamen.

### Beispiele:

- ‘N[aeo]nnen’ kann für ‘Nannen’, ‘Nennen’ und für ‘Nonnen’ stehen,
- ‘\*.tex’ steht für jede Datei, deren Namen mit ‘.tex’ aufhört,
- ‘A??en’ für Namen wie ‘Affen’, ‘Alpen’, ‘Anden’...

## 2.4 Vorder- und Hintergrundprozesse

Normalerweise wartet die Shell (das Fenster, in dem die Kommandos eingegeben werden) immer ab, bis ein aufgerufener Befehl vollständig abgearbeitet worden ist, und akzeptiert erst dann das nächste Kommando. Dies kann unter Umständen aber unerwünscht sein, z.B. wenn man einen Texteditor (z.B. Emacs) oder einen WWW-Browser (z.B. Netscape) startet. In diesen Fällen würde die Shell auf keine Eingabe mehr reagieren, solange der Editor bzw. der Browser nicht beendet wird.

Deshalb gibt es in Unix die Möglichkeit, eine Befehlsausführung als Hintergrundprozeß zu deklarieren. Dadurch kann man auch weiterhin die Kommandoeingabe benutzen, wenn ein Editor oder Browser läuft. Erreicht wird dies durch das Anhängen von **&** an das betroffene Kommando.

### Beispiel:

```
$ emacs brief &
```

Hier wird der Editor Emacs zur Bearbeitung der Datei *brief* aufgerufen, und der Prozeß startet im Hintergrund, d.h. die Shell reagiert weiterhin auf Eingaben.

## 2.5 Umlenken der Ein- und Ausgabe

In Unix gibt es drei Kanäle, die jeweils mit einem Gerät verbunden sind:

1. *stdin*: standard-input (Standardeingabe-Kanal), über den Kommandos ihre Eingaben lesen (Tastatur)
2. *stdout*: standard-output (Standardausgabe-Kanal), auf den Kommandos ihre Ausgaben schreiben (Bildschirm)
3. *stderr*: standard-error (Standardfehler-Kanal), auf dem Fehlermeldungen ausgegeben werden (Bildschirm)

Diese Vorbelegung kann mit den Umleitungen ‘>’, ‘<’ und ‘2>’ verändert werden (siehe Abb. 2).

Die Umleite-Zeichen <, > oder 2> müssen dazu — wie ein Argument — hinter dem Befehlsnamen stehen, bei dessen Ausführung sie wirksam sein sollen. Hinter jedem Umleite-Zeichen

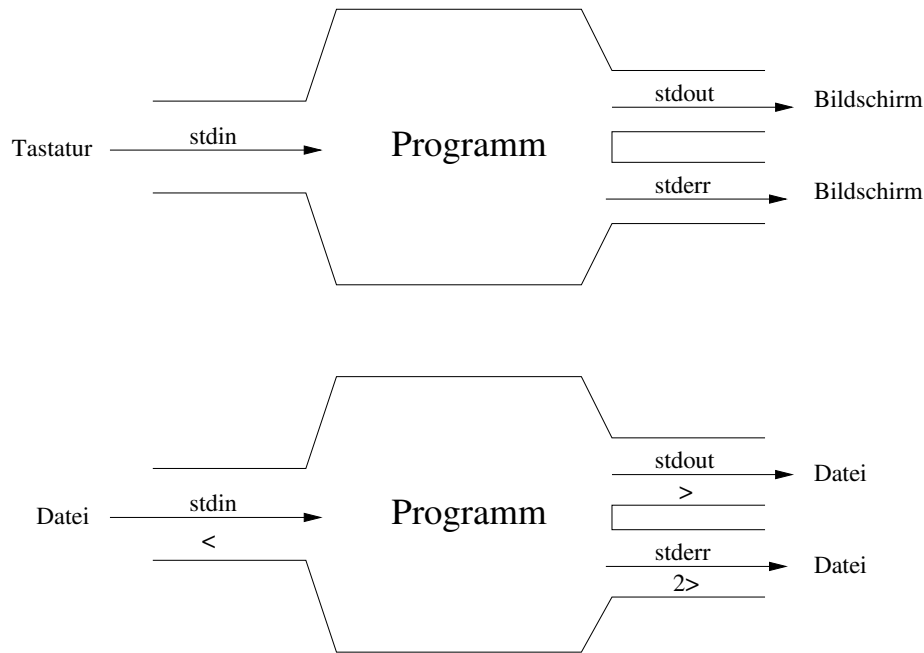


Figure 2: Die Standardkanäle des Unix-Systems und die umgelenkte Ein- und Ausgabe

muß der Name der Datei angegeben werden, aus der (bei <) statt vom Standard-Eingabekanal gelesen bzw. in die (bei > oder 2>) Ausgaben bzw. Fehlermeldungen geschrieben werden sollen.

### Beispiel:

```
[fdfeuers@clue18:/users/linguistik/fdfeuers/Test]
$ cd ..
[fdfeuers@clue18:/users/linguistik/fdfeuers]
$ ll .. > long-listing
[fdfeuers@clue18:/users/linguistik/fdfeuers]
$ less long-listing
total 396
drwxr-xr-x  7 aewald  users      2048 May 10 13:53 aewald
drwxr-xr-x 13 aagrue  users      3072 Feb 19 11:59 aagrue
drwxr-xr-x  5 aahauth users      1024 Jul 29  1992 aahauth
...
drwxr-xr-x  7 jgalber  users      1024 Feb 12 14:49 jgalber
drwxr-xr-x  6 jgdehn   users      1024 Feb  1 13:35 jgdehn
drwxr-xr-x  8 jgsieman users      2048 Apr  5 09:56 jgsieman
--More--(40%)
[fdfeuers@clue18:/users/linguistik/fdfeuers]
$ mail mnlorenz < long-listing
```

Normalerweise würde die Ausgabe des Befehls "ll .." auf dem Bildschirm erfolgen. Durch den Zusatz "> long-listing" wird die Ausgabe auf die Datei *long-listing* umgelenkt. Die letzte Beispielzeile bewirkt, daß dem Benutzer *mnlorenz* der Inhalt der Datei *long-listing* als Mail geschickt wird. Hier wird die Eingabe von der Tastatur auf die Datei *long-listing* umge-

lenkt.

## 2.6 Verkettung von Kommandos

Mit Hilfe sogenannter *Pipes* (Röhren) können mehrere Kommandos miteinander verbunden werden. Dabei nimmt das folgende Kommando die Ausgabe des vorherigen Kommandos direkt als Eingabe. Schreiben sie dazu beide Befehle in die gleiche Kommandozeile, und schreiben Sie das Zeichen '|' dazwischen.

### Beispiel:

```
[fdfeuers@clue18:/users/linguistik/fdfeuers]
$ ll /etc | less
total 23372
-r--r--r--  1 root      sys           321 May  2  1992 #clusterconf
-rw-r--r--  1 root      sys           885 Aug 11  1992 #hosts
...
-r-xr-xr-x  1 bin        bin          20480 Jun  6  1991 edquota
drwxr-xr-x  5 bin        bin           1024 Aug 11  1992 eisa
-r-xr--r--  1 bin        bin          204800 Jun  6  1991 eisa_config
--More--
```

Hier nimmt *less* als Argument die Eingabe des Kommandos '|ll'; der Inhalt des Verzeichnisses wird durch diesen (Doppel-) Befehl seitenweise angezeigt.

## 2.7 Online-Dokumentation des Unix-Systems

Mit dem Befehl

**man** *Befehl*

erhalten Sie Informationen über Eingabeformat und Funktion eines Befehls sowie über Fehlermeldungen und Verweise auf hiermit zusammenhängende Befehle. Beispiel:

```
$ man sort
```

Es kann manchmal eine Weile dauern, bis die Dokumentation auf dem Bildschirm steht. Der Text wird seitenweise angezeigt, die nächste Seite erscheint auf dem Bildschirm, wenn Sie die Leertaste drücken. Die Anzeige wird abgebrochen, wenn Sie 'q' (*quit*) eingeben.

Gerade für Anfänger(innen) sind die angezeigten Seiten wegen der großen Informationsdichte oft abschreckend. Allerdings reicht es oft aus, sich die Kurzbeschreibung und Syntax auf der ersten Seite anzuschauen, um das wesentliche eines Befehls zu verstehen.

## 3 Wichtige Unix-Befehle

### 3.1 Arbeitsverzeichnis wechseln — cd

Mit dem Befehl

## *cd* directory

wird das Arbeitsverzeichnis auf das Verzeichnis *directory* gesetzt. *directory* kann dabei ein absoluter oder relativer Pfadname eines Verzeichnisses sein. Wird die Bezeichnung des Verzeichnisses weggelassen, also nur *cd* angegeben, so wird in das Benutzerverzeichnis gewechselt.

### Beispiel:

```
[fdfeuers@clue18:/users/linguistik/fdfeuers]
$ cd /users/goodies
[fdfeuers@clue18:/users/goodies]
$ cd ../../etc
[fdfeuers@clue18:/etc]
$ cd .
[fdfeuers@clue18:/etc]
$ cd ..
[fdfeuers@clue18:/]
$ cd
[fdfeuers@clue18:/users/linguistik/fdfeuers]
$
```

## 3.2 Inhalt eines Verzeichnisses auflisten — *ls*

Der Befehl

### *ls* directory

gibt die Namen aller Dateien und Verzeichnisse, die sich im Verzeichnis *directory* befinden, auf dem Bildschirm aus. Wird *directory* weggelassen, dann werden die Dateien und Unterverzeichnisse des Arbeitsverzeichnis ausgegeben.

Der Befehl *ll* arbeitet ähnlich wie *ls*, aber er gibt noch genauere Informationen über die Dateien aus.

### Beispiel:

```
$ ll /users/tex/PS
total 330
-rw-rw-rw-  1 root   sys      540 Apr 29 11:56 BUGS
-rw-rw-rw-  1 root   sys     2001 Apr 29 11:56 Makefile
-rw-rw-rw-  1 root   sys    4018 Apr 29 11:56 README
-rw-rw-rw-  1 root   sys     951 Apr 29 11:56 fntchoice-b.tex
-rw-rw-rw-  1 root   sys     941 Apr 29 11:56 fntchoice-h.tex
-rw-rw-rw-  1 root   sys     967 Apr 29 11:56 fntchoice-n.tex
-rw-rw-rw-  1 root   sys     943 Apr 29 11:56 fntchoice-p.tex
-rw-rw-rw-  1 root   sys    1792 Apr 29 11:56 font-table
-rw-rw-rw-  1 root   sys    6793 Apr 29 11:56 fonts.tex
-rw-rw-rw-  1 root   sys   28059 Apr 29 11:56 lfonts.tex
-rw-rw-rw-  1 root   sys    3382 Apr 29 11:56 makefonts.ps
```

```

-rw-r--r--  1 root      sys          83677 Apr 29 11:56 pslatex.shar
-rw-rw-rw-  1 root      sys          10849 Apr 29 11:56 pslatex.tex
-rw-rw-rw-  1 root      sys          18616 Apr 29 11:56 pslplain.tex

-rwxr-xr--  1 fdfeuers  stone       21373 May 11  14:02 hammer

```

Datei- oder Verzeichnisname  
 Uhrzeit der letzten Veränderung  
 Datum der letzten Veränderung  
 Dateigröße in Zeichen/Bytes  
 Gruppe, zu der der Dateibesitzer gehört  
 Loginname des Dateibesitzers  
 Anzahl der Verweise auf diese Datei  
 Dateiart, Zugriffsrechte für Dateibesitzer (user), Gruppe (group)  
 und alle anderen Benutzer (others)

Ein ‘-’ in der ersten Spalte, der Dateiartspalte, kennzeichnet eine Datei, ein ‘d’ ein Verzeichnis. Für jede der drei Benutzerklassen — den Besitzer der Datei, die Mitglieder der Gruppe, in der auch der Dateibesitzer Mitglied ist, und alle anderen Benutzer — werden die *Zugriffsrechte* angegeben. Die Zugriffsrechte sind ‘r’ (lesen, *Read*), ‘w’ (schreiben, löschen und ändern, *Write*) und ‘x’ (als Programm ausführen, *eXecute*) für Dateien. Für Verzeichnisse haben die Zugriffsrechte eine etwas andere Bedeutung:

- *r* — Dateien in dem Verzeichnis können aufgelistet werden
- *w* — in dem betreffenden Verzeichnis können Dateien erzeugt, umbenannt oder entfernt werden
- *x* — in dem betreffenden Verzeichnis kann nach Dateien gesucht werden; z. B. können Dateien mit dem Befehl ‘less’ angeschaut werden

Die Zugriffsrechte werden in dieser Reihenfolge für jede Benutzerklasse wiederholt:

rwx	rwx	rwx
Benutzer	Gruppe	Andere

Wird ein Zugriffsrecht gesperrt, so steht statt des Buchstabens ein ‘-’.

Im obigen Beispiel besitzt der Benutzer ‘fdfeuers’, der zur Gruppe ‘stone’ gehört, Lese-, Schreib- und Ausführungsrechte für die Datei ‘hammer’, die in der Gruppe ‘stone’ zusammengefaßten Benutzer besitzen Lese- und Ausführungsrecht und alle anderen Benutzer nur Leserecht.

### 3.3 Zugriffsrechte einer Datei ändern — `chmod`

Mit dem Befehl

**`chmod`** *mode file*

können Sie die Zugriffsrechte für Dateien und Verzeichnisse ändern. *file* steht für den Namen der Datei, deren Zugriffsrechte geändert werden sollen. *mode* gibt die neuen Zugriffsrechte an und ist folgendermaßen aufgebaut:

class op permission

Der *class*-Teil ist eine Kombination der Buchstaben ‘u’ (für *user*, Besitzer-Zugriffsrechte), ‘g’ (Gruppe) und ‘o’ (*other*, andere). Für die angegebenen Benutzerklassen werden die Zugriffsrechte geändert. Wenn für *op* ein ‘+’ steht, werden Zugriffsrechte hinzugefügt, mit ‘-’ werden Zugriffsrechte gesperrt, und mit ‘=’ werden Zugriffsrechte definiert, d.h. nur die angegebenen Rechte sind vorhanden, die anderen werden gesperrt. *permission* steht für eine Kombination der Zugriffsrechte ‘r’, ‘w’ oder ‘x’. Mehrere solcher *modes* dürfen aufeinander folgen, wenn sie durch Kommas getrennt werden.

### Beispiel:

```
$ ll vuelog
-rw-r--r--  1 fdfeuers users          65 May 10 16:54 vuelog
```

```
$ chmod u=rx,g+x,o-r vuelog
```

```
$ ll vuelog
-r-xr-x---  1 fdfeuers users          65 May 10 16:54 vuelog
```

```
$ chmod u+w,o=rx vuelog
```

```
$ ll vuelog
-rwxr-xr-x  1 fdfeuers users          65 May 10 16:54 vuelog
```

## 3.4 Textdatei seitenweise anzeigen — less

*less file*

Das Programm *less* zeigt die Textdatei *file* an und stoppt, sobald es das Shell-Fenster (eine Seite) vollgeschrieben hat. Nun können Sie entweder die Ausgabe abbrechen, indem Sie ‘q’ eingeben, oder die nächste Zeile anschauen, wenn Sie RETURN drücken, oder mit der Leertaste die nächste Seite anschauen. Geben Sie ‘h’ ein, werden weitere Funktionen des Befehls erklärt.

## 3.5 Verzeichnisse anlegen und entfernen — mkdir, rmdir

Mit dem Befehl

**mkdir** *directory*

wird das Verzeichnis *directory* neu angelegt. *directory* ist ein beliebiger Pfadname. Es darf noch kein Verzeichnis und auch keine Datei mit dem angegebenen Namen (im entsprechenden Vater-Mutter-Verzeichnis) existieren.

### Beispiel:

```
[fdfeuers@clue18:/users/linguistik/fdfeuers]
$ ll -d Test
```

```

Test not found
[fdfeuers@clue18:/users/linguistik/fdfeuers]
$ mkdir Test
[fdfeuers@clue18:/users/linguistik/fdfeuers]
$ ll -d Test
drwx-----  2 fdfeuers users          24 May 11 19:59 Test

```

(Die Option ‘-d’ zum Befehl *ll* gibt an, daß Informationen zum Verzeichnis und nicht zu ihren Unterverzeichnissen ausgegeben werden sollen.)

Ein Verzeichnis wird mit dem Befehl

### **rmdir** *directory*

entfernt. Vorher müssen Sie allerdings alle Unterverzeichnisse und Dateien aus diesem Verzeichnis löschen, sonst erscheint die Meldung “Cannot remove current directory”.

#### **Beispiel:**

```

[fdfeuers@clue18:/users/linguistik/fdfeuers]
$ rmdir Test
[fdfeuers@clue18:/users/linguistik/fdfeuers]
$ ll -d Test
Test not found
[fdfeuers@clue18:/users/linguistik/fdfeuers]
$ rmdir /users/linguistik/fdfeuers
rmdir: /users/linguistik/fdfeuers: Cannot remove current directory

```

## **3.6 Kopieren von Dateien — cp**

Der Befehl

### **cp** *woher wohin*

kopiert die Datei bzw. das Verzeichnis, das durch *woher* bezeichnet wird, an die durch *wohin* angegebene Stelle im Dateibaum. Ist *woher* ein Verzeichnisname, so muß *wohin* ebenfalls für ein Verzeichnis stehen. Wenn es schon eine Datei namens *wohin* gibt, wird sie beim Kopieren "überschrieben.

#### **Beispiel:**

```

[fdfeuers@clue18:/users/linguistik/fdfeuers]
$ mkdir Test
[fdfeuers@clue18:/users/linguistik/fdfeuers]
$ cp /etc/passwd Test/topsecret
[fdfeuers@clue18:/users/linguistik/fdfeuers]
$ ll Test
total 30
-r--r--r--  1 fdfeuers users      15291 May 11 20:33 topsecret

```

```

[fdfeuers@cluel18:/users/linguistik/fdfeuers]
$ cp /etc/hosts Test
[fdfeuers@cluel18:/users/linguistik/fdfeuers]
$ ll Test
total 36
-r--r--r--  1 fdfeuers users      2345 May 11 20:34 hosts
-r--r--r--  1 fdfeuers users     15291 May 11 20:33 topsecret
[fdfeuers@cluel18:/users/linguistik/fdfeuers]
$ cp /etc/che* Test
$ ll Test
total 74
-rw-r--r--  1 fdfeuers users      323 May 11 20:37 checklist
-rw-rw-rw-  1 fdfeuers users      248 May 11 20:37 checklist.old
-rw-rw-rw-  1 fdfeuers users       77 May 11 20:37 checklist.venus
-rw-r--r--  1 fdfeuers users      326 May 11 20:37 checklist~
-r--r--r--  1 fdfeuers users      2345 May 11 20:34 hosts
-r--r--r--  1 fdfeuers users     15291 May 11 20:33 topsecret

```

### 3.7 Entfernen von Dateien — rm

Der Befehl

**rm** *file*

löscht die Datei *file*. Da es keine Möglichkeit gibt, gelöschte Dateien wiederherzustellen, empfiehlt sich die Verwendung der Option *-i*, denn dann wird vor dem Löschen jeder einzelnen Datei noch einmal nachgefragt. Verzeichnisse können mit diesem Befehl nicht gelöscht werden; hierfür ist der Befehl *rmdir* zuständig.

#### Beispiel:

```

[fdfeuers@cluel18:/users/linguistik/fdfeuers]
$ cd Test
[fdfeuers@cluel18:/users/linguistik/fdfeuers/Test]
$ rm hosts
hosts: ? (y/n) y
[fdfeuers@cluel18:/users/linguistik/fdfeuers/Test]
$ ll
total 68
-rw-r--r--  1 fdfeuers users      323 May 11 20:37 checklist
-rw-rw-rw-  1 fdfeuers users      248 May 11 20:37 checklist.old
-rw-rw-rw-  1 fdfeuers users       77 May 11 20:37 checklist.venus
-rw-r--r--  1 fdfeuers users      326 May 11 20:37 checklist~
-r--r--r--  1 fdfeuers users     15291 May 11 20:33 topsecret
[fdfeuers@cluel18:/users/linguistik/fdfeuers/Test]
$ rm -f che*
[fdfeuers@cluel18:/users/linguistik/fdfeuers/Test]
$ ll

```

```
total 60
-r--r--r--  1 fdfeuers users      15291 May 11 20:33 topsecret
```

### 3.8 Umbenennen von Dateien und Verzeichnissen — mv

Mit dem Befehl

```
mv alter_name neuer_name
```

können Dateien und Verzeichnisse umbenannt werden. Da der neue Name ein beliebiger Pfad sein kann, können hiermit auch Dateien in beliebige andere Verzeichnisse bewegt (engl. *move*) werden. Wenn schon eine Datei oder ein Verzeichnis mit dem neuen Namen existiert, bricht der Befehl mit einer Fehlermeldung ab.

#### Beispiel:

```
[fdfeuers@clue18:/users/linguistik/fdfeuers/Test]
$ mv topsecret passwd
[fdfeuers@clue18:/users/linguistik/fdfeuers/Test]
$ ll
total 30
-r--r--r--  1 fdfeuers users      15291 May 11 20:33 passwd
```

### 3.9 Kennwort ändern — passwd

Mit dem Befehl

```
passwd
```

(kurz für *password*) können Sie Ihr Kennwort für den Login ändern. Haben Sie den Befehl eingetippt, fragt der Computer Sie nach Ihrem alten Kennwort, um unbefugtes Ändern zu verhindern. Danach müssen Sie das neue Kennwort eingeben und, sicherheitshalber, noch ein zweites Mal.

Ihr Kennwort muß folgendermaßen aussehen:

- Es muß mindestens sechs Zeichen lang sein
- Mindestens zwei Zeichen müssen Buchstaben sein
- Mindestens ein Zeichen muß eine Ziffer oder ein Sonderzeichen sein

### 3.10 Dateien ausdrucken — lp

Mit dem Befehl

```
lp datei
```

kann der Inhalt von *datei* ausgedruckt werden. Dazu steht hier in der Computerlinguistik der Drucker „ariel“ zur Verfügung.

Will man genau definieren, auf welchem Drucker die Datei ausgegeben werden soll, kann man die Funktion folgendermaßen aufrufen:

**lp -ddruckername datei**

Man beachte: Hinter der Option **-d** folgt kein Leerzeichen. Der Druckername wird kleingeschrieben (also z.B. ariel). Um in der Bismarckstr. 6 zu drucken, lautet der Befehl also:

```
lp -dariel datei
```

Da das Ausdrucken mit Papier- und Tonerverbrauch verbunden ist, ist es nicht kostenlos, sondern jede gedruckte Seite kostet 0,05 Euro.

Wird ein Druckauftrag abgeschickt, erscheint in etwa folgende Meldung:

```
Auftragskennung ist ariel-316 (1 Datei)
```

Mit dieser Auftragskennung ist es auch möglich, einen Druckbefehl noch zu stoppen. Dies ist z.B. dann notwendig, wenn man bemerkt, daß man versehentlich eine falsche Datei ausdrucken wollte, oder wenn man aus Zeitmangel nicht mehr auf den Ausdruck warten kann. Verwendet wird folgender Befehl:

**cancel auftrag,**

wobei *auftrag* die Auftragskennung (z.B. ariel-316) ist.

### 3.11 Dateien archivieren — tar

Mit dem Befehl

**tar -cf archivname dateiname**

kann man ein Archiv aus einer oder mehreren Dateien anlegen (d.h. Dateien zu einem größeren File zusammenfassen). Will man aus dem Archiv wieder die Dateien herstellen, benützt man folgenden Aufruf:

**tar -xf archivname**

### 3.12 Dateien komprimieren — gzip

Um mehr Festplattenplatz zu gewinnen oder Dateien ressourcensparend im Internet zu übertragen, kann man die Daten zuvor komprimieren. Dazu benutzt man folgenden Befehl:

**gzip dateiname**

Damit wird die Datei *dateiname* durch die gepackte Datei *dateiname.gz* ersetzt. Diese ist dann teilweise weniger als halb so groß wie das Original. Um die Daten wieder benutzen zu können, müssen sie entpackt werden. Dies geschieht mit folgendem Aufruf:

**gunzip dateiname.gz**

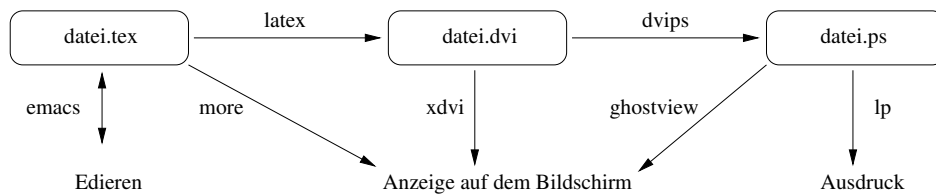


Figure 3: Befehle zum Umwandeln und Ausgeben von Tex-, DVI- und PostScript-Dateien  
ps}

### 3.13 Verschicken von Dateien — mail

Mit dem Befehl

```
mail adressat < datei
```

Wird die Datei *datei* an den elektronischen Briefkasten vom Benutzer mit dem Login-Namen *adressat* geschickt. Wird ‘<*datei*’ fortgelassen, so wird die Eingabe von Tastatur verschickt. Sie müssen in diesem Fall Ihre Eingabe mit CTRL-D abschließen.

#### Beispiel:

```
[fdfeuers@clue18:/users/linguistik/fdfeuers]
$ mail bnbeutel <loesungen
```

### 3.14 Verketteten von Dateien — cat

Mit dem Befehl

```
cat datei1 datei2 ...
```

werden alle angegebenen Dateien hintereinander auf der Standardausgabe (also normalerweise der Bildschirm) ausgegeben. Mittels Ausgabeumlenkung können Sie diese Ausgabe auf eine Datei umlenken.

### 3.15 Umwandeln des Textformats — latex, dvips

Zur Erzeugung wissenschaftlicher Dokumente bedient man sich häufig des Textsatzprogramms  $\text{\LaTeX}$  (genaueres hierzu in der  $\text{\LaTeX}$ -Einführung). Damit wird aus einer normalen Textdatei eine sogenannte DVI-Datei generiert. DVI ist ein Kürzel für *Device independent*, was übersetzt *geräteunabhängig* bedeutet. Das heißt, die erzeugte Datei kann auf allen Rechnern und mit allen Systemen interpretiert werden.

DVI-Dateien lassen sich mit dem Kommando **xdvi** auf dem Bildschirm betrachten. Will man sie allerdings ausdrucken, muß zuerst eine PostScript-Datei erzeugt werden. Diese hat die Endung *.ps* und wird durch folgenden Aufruf aus der DVI-Datei generiert:

```
dvips -o dateiname.ps dateiname.dvi
```

Für `dateiname` muß man den jeweils gewünschten Namen für die PostScript-Ausgabedatei und die DVI-Eingabedatei angeben.

Hat man schließlich einen PostScript-Text erzeugt, läßt sich dieser mit dem normalen **lp**-Kommando ausdrucken. Mit **ghostview** kann man sich auch hier den Text auf dem Bildschirm anzeigen lassen.

Eine Zusammenfassung der Möglichkeiten ist in Abb. 3ps } dargestellt.