
Dateibefehle

Grundlegende Dateioperationen

<code>ls</code>	Listet Dateien in einem Verzeichnis auf.
<code>cp</code>	Kopiert eine Datei.
<code>mv</code>	Benennt eine Datei um (»verschiebt« die Datei).
<code>rm</code>	Löscht (»entfernt«) eine Datei.
<code>ln</code>	Erzeugt Links (alternative Namen) für eine Datei.

ls `stdin` `stdout` `-datei` `--opt` `--help` `--version`

`ls [optionen] [dateien]`

Der `ls`-Befehl (so gesprochen, wie er geschrieben wird: »ell es«) listet die Attribute von Dateien und Verzeichnissen auf. Sie können die Dateien im aktuellen Verzeichnis auflisten:

→ `ls`

in bestimmten Verzeichnissen:

→ `ls ver1 ver2 ver3`

oder einzeln:

→ `ls datei1 datei2 datei3`



Wenn sich `ls` anders als erwartet verhält, hat Ihre Distro eventuell einen Alias dafür definiert (siehe »Aliase« auf Seite 43). Das Vorhandensein eines Alias können Sie wie folgt prüfen:

→ **alias ls**

`alias ls='/bin/ls -FHN'` *Ja, es gibt einen Alias.*

Um den eigentlichen Befehl und nicht den Alias auszuführen, setzen Sie einen Backslash davor (`\ls`). Um den Alias zu entfernen, rufen Sie `unalias ls` auf. Dann prüfen Sie Ihre Konfigurationsdateien für die Shell darauf, ob der Alias dort definiert ist (siehe »Das Shell-Verhalten anpassen« auf Seite 59), und entfernen ihn. Sehen Sie dort keine Definition, fügen Sie entweder den Befehl `unalias` hinzu, oder Sie definieren einen neuen Alias, der so funktioniert, wie Sie es sich wünschen, zum Beispiel `alias ls='/bin/ls'`.

Die wichtigsten `ls`-Optionen sind `-a`, `-l` und `-d`. Standardmäßig verbirgt `ls` Dateien, deren Namen mit einem Punkt beginnen, wie im Kasten »»Punktdateien« auf Seite 39« erläutert wurde. Die Option `-a` zeigt alle Dateien an. Abhängig von den Einstellungen Ihres Kontos wird `ls` Punktdateien am Anfang der Ausgabeliste anzeigen (Sortierung basiert auf dem Punkt) oder sie in die Liste einmischen (Sortierung basiert auf den Zeichen nach dem Punkt).

→ **ls**

`meinedatei1 meinedatei2`

→ **ls -a**

`.verborgene_datei meinedatei1 meinedatei2`

Die Option `-l` erzeugt eine lange Liste:

→ **ls -l meinedatei1**

`-rw-r--r-- 1 smith users 1168 Oct 28 2015 meinedatei1`

Von links nach rechts enthält sie folgende Informationen: Dateiberechtigungen (`-rw-r--r--`), Anzahl der Hardlinks (1), Besitzer (smith), Gruppe (users), Größe (1168 Byte), Datum der letzten Änderung (Oct 28 2015) und den Namen. Mehr Informationen über die Dateiberechtigungen finden Sie in »Dateiberechtigungen« auf Seite 36.

```
→ ls -ld meinverz1
drwxr-xr-x 1 smith users 4096 Oct 29 2015 meinverz1
```

- a Listet alle Dateien auf, auch die, deren Namen mit einem Punkt beginnen.
- l Lange Liste, einschließlich der Dateiattribute. Fügen Sie die Option -h hinzu (von Menschen lesbar), um die Dateigrößen in Kilobyte, Megabyte und Gigabyte auszugeben anstatt in Byte.
- h Gibt in einer langen Liste die Dateigrößen statt in Byte in besser lesbaren KB, MB und GB aus.
- G Lange Liste, aber ohne Gruppenangabe.
- F Verziert bestimmte Dateinamen mit bedeutungsvollen Symbolen, die deren Typ anzeigen. Hängt »/« an Verzeichnisse, »*« an ausführbare Dateien, »@« an symbolische Links, »|« an benannte Pipes und »=« an Sockets an. Diese Indikatoren sind rein informativ und nicht Teil der Dateinamen!
- S Sortiert die Dateien anhand ihrer Größe.
- t Sortiert die Dateien anhand ihres letzten Bearbeitungszeitpunkts.
- r Dreht die Sortierreihenfolge um.
- R Listet beim Auflisten eines Verzeichnisses dessen Inhalt rekursiv auf.
- d Beim Auflisten eines Verzeichnisses wird nicht dessen Inhalt, sondern nur das Verzeichnis selbst angeführt.

```
cp [optionen] quelledatei zieldatei
cp [optionen] (dateien | verzeichnisse) verzeichnis
```

```
→ cp datei1 datei2 datei3 datei4 zielverzeichnis
```

Mithilfe der Optionen `-a` oder `-r` können Sie Verzeichnisse auch rekursiv kopieren, inklusive aller Unterverzeichnisse und deren Inhalte. Für ausgefeiltere Kopiervorgänge schauen Sie sich `rsync` in »Backups und externe Speicherung« auf Seite 238 an.

Nützliche Optionen

- p Kopiert nicht nur den Dateiinhalt, sondern auch die Dateiberechtigungen, die Zeitstempel und, falls Sie die passenden Rechte dafür haben, ihren Eigentümer und die Gruppe. (Normalerweise gehören die Kopien Ihnen, sie haben den Zeitstempel der aktuellen Zeit, und die Berechtigungen werden gesetzt, indem Ihr `umask` auf die ursprünglichen Berechtigungen angewandt wird.)
- a Kopiert eine Verzeichnishierarchie rekursiv, wobei alle Dateiattribute und `-links` beibehalten werden.
- r Kopiert eine Verzeichnishierarchie rekursiv. Diese Option behält die Attribute der Dateien, wie Berechtigungen und Zeitstempel, nicht bei. Symbolische Links dagegen werden bewahrt.
- i Interaktiver Modus. Fragt, bevor Zieldateien überschrieben werden.
- f Erzwingt die Kopie. Falls eine Zieldatei existiert, wird diese bedingungslos überschrieben.

mv **stdin** **stdout** **- datei** **-- opt** **--help** **--version**

`mv [optionen] quellen ziel`

Der `mv`-Befehl (*move*) kann eine Datei umbenennen:

→ `mv einedatei anderedatei`

oder Dateien und Verzeichnisse in ein Zielverzeichnis verschieben:

→ `mv datei1 datei2 verz3 verz4 zielverzeichnis`

Nützliche Optionen

- i Interaktiver Modus. Fragt vor dem Überschreiben der Zieldateien.
- f Erzwingt das Verschieben. Falls eine Zieldatei existiert, wird diese bedingungslos überschrieben.

rm stdin stdout -datei --opt --help --version

rm [optionen] dateien | verzeichnisse

Der rm-Befehl (*remove*) kann Dateien löschen:

→ **rm datei1 datei2 datei3**

oder rekursiv Verzeichnisse löschen:

→ **rm -r verz1 verz2**



Setzen Sie **rm -r** mit Bedacht ein. Es kann sehr schnell eine große Anzahl an Dateien löschen, insbesondere wenn es mit **-f** kombiniert wird, um Nachfragen zu unterdrücken und Fehler zu ignorieren.

Nutzen Sie **sudo rm -r** mit *noch viel mehr* Bedacht – es kann Ihr gesamtes Betriebssystem zerstören.

Nützliche Optionen

- i Interaktiver Modus. Fragt vor dem Löschen der einzelnen Dateien.
- f Erzwingt das Löschen, ignoriert dabei alle Fehler- und Warnmeldungen.
- r Entfernt rekursiv ein Verzeichnis und seinen Inhalt.

ln stdin stdout -datei --opt --help --version

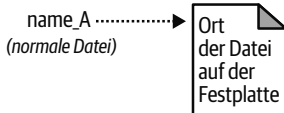
ln [optionen] quelle ziel

Der ln-Befehl erstellt *Links*, mit denen eine Datei im Dateisystem an mehreren Orten gleichzeitig existieren kann. Es gibt zwei Arten von Links (siehe Abbildung 2-1). Ein *harter Link* (oder *Hardlink*) ist ein zweiter Name für eine physische Datei auf einer Festplatte. Technisch ausgedrückt, verweist er auf denselben Inode – eine Datenstruktur, die den Inhalt einer Datei auf der Festplatte referenziert. Der folgende Befehl erzeugt einen harten Link *meinharderlink* für die Datei *meinedatei*:

→ **ln meinedatei meinharderlink**

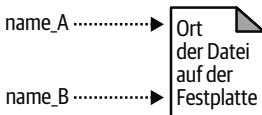
Eine Datei anlegen:

```
$ touch name_A
```



Harter Link:

```
$ ln name_A name_B
```



Symbolischer Link:

```
$ ln -s name_A name_B
```

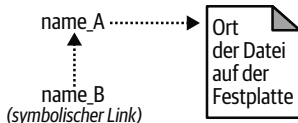


Abbildung 2-1: Ein harter Link verweist auf die Daten einer Datei. Ein symbolischer Link verweist auf einen Dateipfad.

Ein *symbolischer Link* (auch *Symlink* oder *Softlink*) verweist auf den *Pfad* (nicht den Inode) einer anderen Datei oder eines anderen Verzeichnisses. Er ähnelt Aliasen in Windows oder macOS, die ebenfalls auf eine andere Datei oder einen anderen Ordner verweisen. Um einen symbolischen Link anzulegen, nutzen Sie die Option `-s`:

```
→ ln -s meinedatei meinsoftlink
```

Harte und symbolische Links besitzen deutliche Unterschiede:

- Wenn Sie die Originaldatei umbenennen oder löschen, ist ein harter Link davon nicht beeinträchtigt – er verweist weiterhin auf die gleichen *Daten* wie die ursprüngliche Datei. Ein symbolischer Link wird dagegen nicht mehr funktionieren. Er verweist auf

einen nicht mehr existierenden *Dateipfad*. Der symbolische Link hängt nun nutzlos herum, und wenn Sie ihn in einem Befehl verwenden, werden Sie vermutlich einen »File not found«-Fehler erhalten.

- Harte Links können nur auf demselben Device wie die ursprüngliche Datei existieren, weil Inodes nur auf solch einem Device eine Bedeutung haben. Symbolische Links können auf Dateien auf anderen Devices verweisen, da es sich bei ihnen nur um Referenzen auf Dateipfade handelt.
- Symbolische Links können auf Verzeichnisse verweisen, harte Links wiederum nicht. (Auf manchen Systemen kann der Administrator mit der Option `-d` einen harten Link auf ein Verzeichnis erzeugen.)

Nützliche Optionen

- s Erzeugt einen symbolischen Link statt eines harten Links.
- i Interaktiver Modus. Fragt vor dem Überschreiben von Zieldateien.
- f Erzwingt den Link. Falls eine Zieldatei existiert, wird diese bedingungslos überschrieben.
- b Erstellt ein Backup. Existiert die Zieldatei schon, wird sie umbenannt, indem ein Tilde an ihren Namen angefügt wird – danach wird dann der Link erzeugt.
- d Erzeugt einen harten Link auf ein Verzeichnis (nur für Administratoren).

Um herauszufinden, wohin ein symbolischer Link zeigt, führen Sie einen der folgenden Befehle aus (mit denen Sie sehen, dass der Link *beispiellink* auf die Datei *meinedatei* verweist):

```
→ readlink beispiellink
```

```
meinedatei
```

```
→ ls -l beispiellink
```

```
lrwxrwxrwx 1 smith ... beispiellink -> minedatei
```

Symbolische Links können auf andere symbolische Links verweisen. Um der ganzen Linkkette bis zu ihrem Ende zu folgen, verwenden Sie `readlink -f`.

pwd	stdin	stdout	- datei	-- opt	--help	--version
------------	--------------	---------------	----------------	---------------	---------------	------------------

pwd

Der Befehl `pwd` gibt den absoluten Pfad Ihres aktuellen Arbeitsverzeichnisses aus:

→ `pwd`
`/users/smith/linuxpocketguide`

basename	stdin	stdout	- datei	-- opt	--help	--version
-----------------	--------------	---------------	----------------	---------------	---------------	------------------

basename *pfad* [*erweiterung*]

Der Befehl `basename` gibt die letzte Komponente in einem Dateipfad aus. Dabei muss der Pfad im Dateisystem gar nicht existieren.

→ `basename /users/smith/finances/money.txt`
`money.txt`
→ `basename any/string/you/want` *beliebiger String*
`want`

Wenn Sie eine zusätzliche Erweiterung angeben, wird diese vom Ergebnis abgetrennt:

→ `basename /users/smith/finances/money.txt .txt`
`money`

dirname	stdin	stdout	- datei	-- opt	--help	--version
----------------	--------------	---------------	----------------	---------------	---------------	------------------

dirname *pfad*

Der Befehl `dirname` gibt einen Dateipfad an, von dem seine letzte Komponente entfernt wurde:

→ `dirname /users/smith/meinverz`
`/users/smith`

`dirname` ändert nicht das aktuelle Arbeitsverzeichnis. Der Befehl manipuliert einfach einen String, genau wie `basename`, und gibt ihn aus.

mkdir **stdin** **stdout** **-datei** **--opt** **--help** **--version**

`mkdir [optionen] verzeichnisse`

mkdir legt ein oder mehrere Verzeichnisse an.

→ `mkdir verzeichnis1 verzeichnis2 verzeichnis3`

Nützliche Optionen

- p Wird ein Verzeichnispfad angegeben (nicht nur ein einfacher Verzeichnisname), werden automatisch die notwendigen übergeordneten Verzeichnisse erzeugt. Der Befehl
→ `mkdir -p eins/zwei/drei`
erzeugt *eins*, *eins/zwei* und *eins/zwei/drei*, falls diese noch nicht existieren.
- m *modus* Erzeugt das Verzeichnis mit den angegebenen Berechtigungen:
→ `mkdir -m 0755 meinverz`
Standardmäßig steuert die *umask* Ihrer Shell die Berechtigungen (siehe den *chmod*-Befehl in »Dateieigenschaften« auf Seite 85 und »Dateiberechtigungen« auf Seite 36).

rmdir **stdin** **stdout** **-datei** **--opt** **--help** **--version**

`rmdir [optionen] verzeichnisse`

Der Befehl *rmdir* (*remove directory*) löscht ein oder mehrere leere Verzeichnisse, deren Namen Sie angegeben haben:

- `mkdir /tmp/junk` *Verzeichnis anlegen*
- `rmdir /tmp/junk` *und wieder löschen*

Nützliche Optionen

- p Wenn Sie einen Verzeichnispfad angeben (nicht lediglich einen einfachen Verzeichnisnamen), wird nicht nur das angegebene Verzeichnis gelöscht, sondern automatisch auch die übergeordneten Verzeichnisse, die aber alle leer sein müssen. *rmdir -p eins/zwei/drei* löscht also nicht nur *eins/zwei/drei*, sondern auch *eins/zwei* und *eins*.

Um ein nicht leeres Verzeichnis und dessen Inhalt zu löschen, benutzen Sie (vorsichtig!) `rm -r verzeichnis`. Nehmen Sie `rm -ri`, um

interaktiv zu löschen, oder `rm -rf`, um ohne Fehlermeldungen oder Bestätigung alles auszulöschen.

Dateien betrachten

<code>cat</code>	Dateien komplett anschauen.
<code>less</code>	Textdateien seitenweise anschauen.
<code>nl</code>	Textdateien mit nummerierten Zeilen anschauen.
<code>head</code>	Die ersten Zeilen einer Textdatei anschauen.
<code>tail</code>	Die letzten Zeilen einer Textdatei anschauen.
<code>strings</code>	Text anzeigen, der in eine Binärdatei eingebettet ist.
<code>od</code>	Daten im Oktalformat (Basis 8) oder in anderen Formaten anschauen.

Manche Dateien enthalten normalen Text, andere Binärdaten. Schauen wir uns an, wie Sie sie so einfach wie möglich darstellen können.

cat	stdin	stdout	-datei	--opt	--help	--version
------------	--------------	---------------	---------------	--------------	---------------	------------------

`cat [optionen] [dateien]`

Der einfachste Betrachter ist `cat`, der seine Dateien auf der Standardausgabe ausgibt, indem er sie aneinanderhängt (*concatenate*).

- `cat meinedatei` *Eine Datei ausgeben.*
- `cat meinedatei*` *Viele Dateien ausgeben.*
- `cat meinedatei* | wc` *Dateien verketten und an wc leiten.*

Enthält eine Datei mehr Zeilen, als auf Ihrem Bildschirm angezeigt werden können, nutzen Sie `less`, um immer nur eine Seite gleichzeitig zu sehen.

Nützliche Optionen

- T Druckt Tabs als `^I`.
- E Druckt Newlines als `$`.
- v Druckt andere nicht druckbare Zeichen in einem für Menschen lesbaren Format.

- n Stellt jeder Zeile eine Zeilennummer voran. (Der Befehl n1 ist noch mächtiger.)
- b Stellt nicht leeren Zeilen eine Zeilennummer voran.
- s Setzt jede Folge von Leerzeilen in eine einzelne Leerzeile.

less

stdin **stdout** **- datei** **-- opt** **--help** **--version**

```
less [optionen] [dateien]
```

Nutzen Sie less, um einen Text »seitenweise« anzuschauen (d. h. jeweils ein Fenster oder einen Bildschirm füllend).

→ less meinedatei

Das ist hervorragend geeignet für Textdateien oder als letzter Befehl in einer Shell-Pipeline mit einer längeren Ausgabe:

→ *befehl1* | *befehl2* | *befehl3* | *befehl4* | less

Wenn Sie less ausführen, tippen Sie h, um eine Hilfenachricht zu erhalten, in der all seine Eigenschaften beschrieben werden. Hier sind einige nützliche Tastenkürzel, um durch die Dateien zu blättern:

Tastenkürzel	Bedeutung
h, H	Anzeigen einer Hilfeseite.
Leertaste, f, ^V, ^F	Einen Bildschirm nach vorn bewegen.
Enter	Eine Zeile nach vorn bewegen.
b, ^B, Esc-v	Einen Bildschirm zurück bewegen.
/	Suchmodus starten. Geben Sie dann einen regulären Ausdruck an und drücken Sie Enter. Less sucht nach der ersten passenden Zeile.
?	Wie /, allerdings wird rückwärts in der Datei gesucht.
n	Nächster Treffer: Wiederholt die neueste Suche vorwärts.
N	Wiederholt die neueste Suche rückwärts.
v	Bearbeitet die aktuelle Datei mit Ihrem Standardtexteditor. (Das ist der Wert der Umgebungsvariablen VISUAL oder, falls nicht definiert, EDITOR oder, falls nicht definiert, der Standardeditor des Systems.)

Nützliche Optionen

-b [a t n pR]	Stellt allen Zeilen (a), nicht leeren Zeilen (t), keinen Zeilen (n) oder Zeilen, die den regulären Ausdruck R enthalten, Zeilennummern voran. (Standard = a)
-v N	Beginnt die Nummerierung mit dem Integerwert N. (Standard = 1)
-i N	Erhöht die Nummer pro Zeile um N, sodass Sie z. B. nur ungerade (-i2) oder nur gerade Zahlen benutzen könnten (-v2 -i2). (Standard = 1)
-n [ln rn rz]	Formatiert die Nummern linksbündig (ln), rechtsbündig (rn) oder rechtsbündig mit führenden Nullen (rz). (Standard = ln)
-w N	Erzwingt eine Breite der Nummer von N Spalten. (Standard = 6)
-s S	Fügt den String S zwischen die Zeilennummer und den Text ein. (Standard = Tab)

head	stdin	stdout	-datei	--opt	--help	--version
-------------	--------------	---------------	---------------	--------------	---------------	------------------

head [optionen] [dateien]

Der Befehl head zeigt die ersten zehn Zeilen einer Datei an – hervorragend für eine Vorschau auf den Inhalt:

→ head **meinedatei**
→ head **meinedatei*** | **less** *Vorschau auf alle Dateien*

Der Befehl eignet sich auch gut als Vorschau auf die ersten Zeilen einer Pipeline-Ausgabe – zum Beispiel die zehn zuletzt veränderten Dateien im aktuellen Verzeichnis:

→ **ls -lta** | **head**

Nützliche Optionen

-n N	Gibt die ersten N Zeilen anstelle der ersten zehn Zeilen aus.
-N	Wie -nN.
-c N	Gibt die ersten N Bytes der Datei aus.
-q	Stillschweigend (<i>quiet</i>): Beim Verarbeiten von mehr als einer Datei wird über den einzelnen Dateien kein Banner (mit dem Dateinamen) angezeigt.

tail	stdin	stdout	-datei	--opt	--help	--version
-------------	--------------	---------------	---------------	--------------	---------------	------------------

tail [optionen] [dateien]

Der Befehl tail gibt die letzten zehn Zeilen einer Datei aus:

→ **tail** meinedatei

→ **nl** meinedatei | **tail** *Zeilennummern mit ausgeben*

Die supernützliche Option -f veranlasst tail, eine Datei aktiv zu überwachen, während ein anderes Programm in sie hineinschreibt, wobei neue Zeilen erscheinen, wenn sie in die Datei geschrieben werden. Das ist unglaublich wichtig, wenn man Logdateien im aktiven Einsatz beobachtet:

→ **tail -f /var/log/syslog** *oder eine andere Logdatei*

Nützliche Optionen

-n *N* Gibt die letzten *N* Zeilen anstelle der letzten zehn Zeilen der Datei aus.

-*N* Wie -n *N*.

-n +*N* Gibt die Zeilen ab Zeile *N* bis zum Ende der Datei aus.

+*N* Wie -n +*N*.

-c *N* Gibt die letzten *N* Bytes der Datei aus.

-f Hält die Datei offen und gibt Zeilen aus, die an die Datei angehängt werden. Fügen Sie die Option --retry hinzu, falls die Datei noch nicht existiert. Mit ^C beenden Sie die Ausgabe.

-q Stillschweigend (*quiet*): Wenn mehr als eine Datei verarbeitet wird, wird kein Banner (mit dem Dateinamen) über den einzelnen Dateien angezeigt.

strings	stdin	stdout	-datei	--opt	--help	--version
----------------	--------------	---------------	---------------	--------------	---------------	------------------

strings [optionen] [dateien]

Binärdateien, wie z.B. ausführbare Programme, enthalten normalerweise ein bisschen lesbaren Text, wie Versionsinformationen, die Namen der Autorinnen und Autoren und Dateipfade. Das Programm strings extrahiert diesen Text:

Kombinieren Sie `strings` -n und `grep`, um Ihre Erkundungen effizienter zu machen. Hier suchen wir nach E-Mail-Adressen:

Nützliche Optionen

```
od          stdin  stdout  -datei  --opt  --help  --version
```

Wenn Sie Binärdateien betrachten wollen, sollten Sie `od` (*octal dump*) dafür in Betracht ziehen. Es kopiert deren Inhalte auf die Standardausgabe, wobei es die Daten in ASCII, oktal, dezimal, hexadezimal oder als Fließkommazahl in verschiedenen Größen (Byte, Short, Long) anzeigt. So zeigt z.B. der Befehl

die Bytes in der Binärdatei `/usr/bin/who` im Oktalformat an, und zwar acht Bytes pro Zeile. Die Spalte ganz links enthält den Datei-Offset jeder Zeile, hier wieder im Oktalformat.

Falls Ihre Binärdatei auch Text enthält, sollten Sie die Option `-tc` benutzen, die Zeichendaten darstellt. Zum Beispiel enthalten ausführbare Binärdateien wie `who` am Anfang den String »ELF«:

78 | Kapitel 2: Dateibefehle

```
0000010  \0  \0  \0  \0  \0  \0  \0  \0
0000020  003  \0  >  \0  001  \0  \0  \0
```

Nützliche Optionen

- N *B* Zeigt nur die ersten *B* Bytes jeder Datei an, und zwar dezimal, hexadezimal (indem *0x* vorangestellt wird), in 512-Byte-Blöcken (indem *b* angehängt wird), in Kilobyte (indem *k* angehängt wird) oder in Megabyte (indem *m* angehängt wird). Standardmäßig wird die gesamte Datei angezeigt.
- j *B* Beginnt die Ausgabe bei Byte *B* + 1 jeder Datei; akzeptable Formate sind identisch mit der Option -N. (Standard = 0)
- w [*B*] Zeigt *B* Bytes pro Zeile an; akzeptable Formate sind identisch mit der Option -N. Nutzt man -w für sich allein, ist dies äquivalent zu -w32. (Standard = 16)
- s [*B*] Gruppirt alle Bytezeilen in Folgen aus *B* Bytes, getrennt durch White-space; akzeptable Formate sind identisch mit der Option -N. Nutzt man -s für sich allein, ist dies äquivalent zu -s3. (Standard = 2)
- A (*d|o|x|n*) Zeigt Datei-Offsets in der ganz linken Spalte an, und zwar dezimal (*d*), oktal (*o*), hexadezimal (*x*) oder gar nicht (*n*). (Standard = *o*)
- t(*a|c*)[*z*] Zeigt die Ausgabe in einem Zeichenformat an, wobei nicht alphanumerische Zeichen als Escape-Folgen (*a*) oder nach dem Namen (*c*) dargestellt werden.
- t(*d|o|u|x*)[*z*] Zeigt die Ausgabe in einem Integerformat an, einschließlich oktal (*o*), vorzeichenbehaftet dezimal (*d*), nicht vorzeichenbehaftet dezimal (*u*), hexadezimal (*x*).

Wird *z* an die Option -t angehängt, erscheint auf der rechten Seite der Ausgabe eine neue Spalte, die die druckbaren Zeichen auf jeder Zeile anzeigt.

Dateierzeugung und -bearbeitung

- nano Ein einfacher Texteditor, der in so gut wie allen Linux-Distributionen zu finden ist.
- emacs Ein leistungsfähiger Texteditor von der Free Software Foundation.
- vim Ein leistungsfähiger Texteditor, Erweiterung des Unix-vi.

Um mit Linux etwas zu erreichen, müssen Sie sich mit einem seiner Texteditoren vertraut machen. Die drei wichtigsten sind nano,

Inhalt

	Das Wichtigste zuerst	11
1	Grundlegende Konzepte.....	19
	Was ist Linux?	19
	Struktur von Befehlen.....	24
	User und Administratoren	26
	Das Dateisystem	27
	Ausgewählte bash-Features	37
	Hilfe erhalten.....	60
2	Dateibefehle.....	63
	Grundlegende Dateioperationen.....	63
	Verzeichnisoperationen	70
	Dateien betrachten	73
	Dateierzeugung und -bearbeitung	79
	Dateieigenschaften.....	85
	Dateisuche	99
	Dateitextmanipulation	108
	Dateikomprimierung und -verpackung	127
	Dateivergleiche	135
	Dateien in andere Formate umwandeln.....	141
	PDF- und PostScript-Dateien	151
	Drucken	156
	Rechtschreibkontrolle.....	157

3	Grundlagen für die Systemadministration	161
	Administrator werden	161
	Prozesse betrachten	163
	Prozesse steuern	169
	Jobs zeitlich planen	174
	Anmelden, abmelden und herunterfahren	180
	Benutzer und deren Umgebung	183
	Verwaltung der Benutzerzugänge	187
	Gruppen verwalten	191
	Software installieren	195
	Software aus Quellcode installieren	206
4	Wartung des Dateisystems	211
	Festplatten und Dateisysteme verwenden	211
	Dateisysteme erstellen und anpassen	217
	RAID-Arrays für Redundanz	221
	Logische Volumes für flexible Speicherlösungen	226
	ZFS: Ein modernes und universelles Dateisystem	232
	Backups und externe Speicherung	238
5	Netzwerkbefehle	245
	Informationen über Ihren Rechner	245
	Hostpositionen	248
	Netzwerkverbindungen	252
	E-Mails im Alltag	258
	E-Mail-Server	262
	Im Internet surfen	266
6	Dinge erledigen	273
	Bildschirmausgabe	273
	Kopieren und Einfügen	279
	Mathematik und Berechnungen	281
	Daten und Zeiten	286
	Versionsverwaltung	290
	Container	295

Bilder anzeigen und verarbeiten	298
Audio und Video	301
Mit Shell-Skripten programmieren	308
Abschließende Worte	325
Index	327