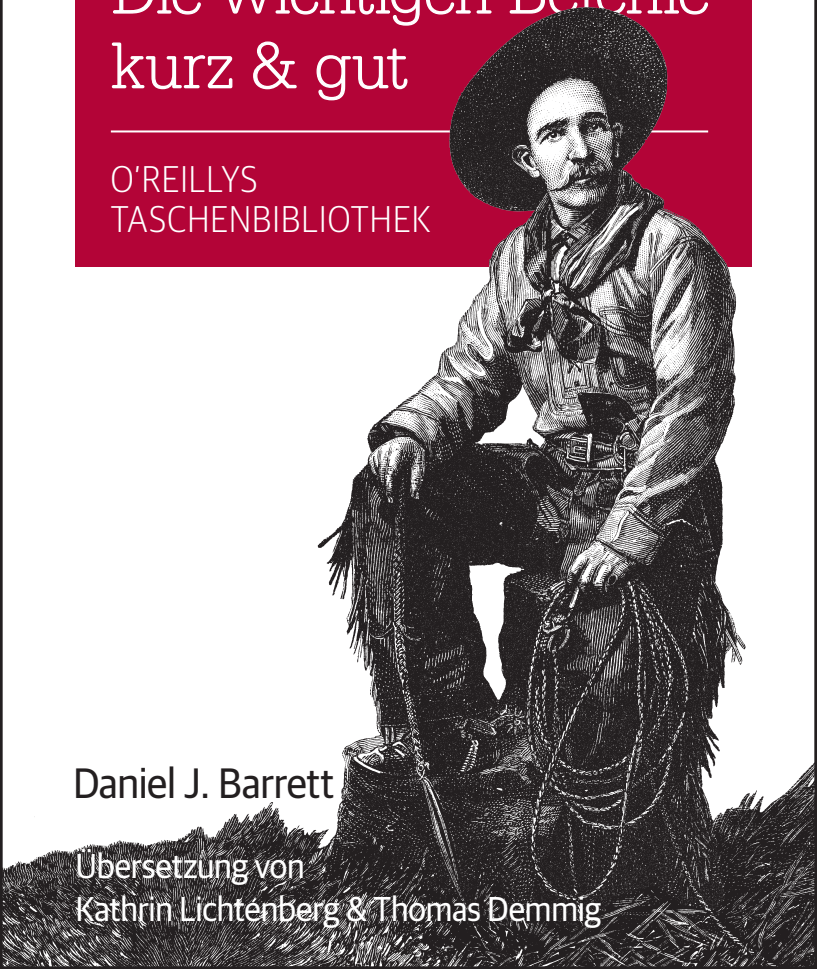
O'REILLY®

3. Auflage

LINUX

Die wichtigen Befehle kurz & gut

O'REILLYS
TASCHENBIBLIOTHEK



Daniel J. Barrett

Übersetzung von
Kathrin Lichtenberg & Thomas Demmig

3. AUFLAGE

Linux – kurz & gut

Die wichtigen Befehle

Daniel J. Barrett

O'REILLY®

Daniel J. Barrett

Lektorat: Ariane Hesse
Korrektorat: Sibylle Feldmann
Satz: III-Satz, www.drei-satz.de
Herstellung: Susanne Bröckelmann
Umschlaggestaltung: Michael Oréal, www.oreal.de
Druck und Bindung: Media-Print Informationstechnologie,
www.mediaprint-druckerei.de

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über *http://dnb.d-nb.de* abrufbar.

ISBN:
Print 978-3-96009-034-2
PDF 978-3-96010-093-5
ePub 978-3-96010-094-2
mobi 978-3-96010-095-9

Dieses Buch erscheint in Kooperation mit O'Reilly Media, Inc. unter dem Imprint »O'REILLY«. O'REILLY ist ein Markenzeichen und eine eingetragene Marke von O'Reilly Media, Inc. und wird mit Einwilligung des Eigentümers verwendet.

3. Auflage

Wieblinger Weg 17
69123 Heidelberg

Authorized German translation of the English edition of Linux Pocket Guide, 3rd Edition, ISBN 9781491927571 © 2016 Daniel J. Barrett. This translation is published and sold by permission of O'Reilly Media, Inc., which owns or controls all rights to publish and sell the same.

5 4 3 2 1 0

Inhalt

Linux – kurz & gut

Willkommen bei Linux! Falls Sie Linux-Einsteiger sind, kann dieses Buch Ihnen als Schnelleinstieg sowie als Referenz zu gebräuchlichen und praktischen Befehlen dienen. Haben Sie dagegen bereits Erfahrungen mit Linux, überspringen Sie die Einführung einfach.

Was erwartet Sie in diesem Buch?

Dieses Buch ist eine Kurzeinführung, *keine umfassende Referenz*. Wir behandeln wichtige, nützliche Aspekte von Linux, damit Sie produktiv arbeiten können. Wir stellen jedoch nicht jeden einzelnen Befehl und auch nicht die allerletzte Option vor (es tut uns leid, falls wir Ihren Liebling weggelassen haben) und verlieren uns auch nicht in den Details der Interna des Betriebssystems. Kurz und prägnant – das ist unser Motto.

Wir konzentrieren uns auf *Befehle*, diese nützlichen kleinen Wörter, die Sie auf einer Kommandozeile eingeben, um einem Linux-System mitzuteilen, was es tun soll. Hier ist z. B. ein Befehl, der die Textzeilen in einer Datei namens *meinedatei* zählt:

```
wc -l meinedatei
```

Wir behandeln die für den durchschnittlichen Nutzer wichtigsten Linux-Befehle, wie etwa `ls` (zum Auflisten von Dateien), `grep` (zum Suchen von Text), `mplayer` (zum Abspielen von Audio- und Videodateien) und `df` (zum Ermitteln des freien Plattenplatzes). Wir werden uns nur am Rande mit grafischen Fensterumgebungen befassen, wie GNOME und KDE, die jede für sich ein »kurz & gut« füllen könnten.

Für einen einfacheren Lernweg haben wir das Material nach Funktionen organisiert. Um Sie z. B. beim Betrachten des Inhalts einer Datei zu unterstützen, führen wir viele der dazu vorhandenen Befehle zusammen ein: `cat` für kurze Textdateien, `less` für längere, `od` für Binärdateien, `acroread` für PDF-Dateien und so weiter. Dann erläutern wir nacheinander die einzelnen Befehle und stellen kurz ihre gebräuchlichsten Einsatzfälle und Optionen vor.

Wir gehen davon aus, dass Sie Zugriff auf ein Linux-System haben und wissen, wie Sie sich mit Ihrem Benutzernamen und Ihrem Passwort dort anmelden. Falls nicht, organisieren Sie sich eine Linux-Live-DVD, wie zum Beispiel *Ubuntu (http://ubuntu.com/download)*, *Fedora (http://getfedora.org/de)* oder *Knoppix (http://www.knopper.net/knoppix/)*, die Sie auf den meisten Computern booten können, um dann mit Linux zu spielen.

Was ist neu in der 3. Auflage?

Neue Befehle
: Die Technologie schreitet schnell voran, und manche Befehle, die in den ersten beiden Auflagen noch gut aufgehoben waren, sind heutzutage kaum mehr im Einsatz. Wir haben sie durch solche ersetzt, die Sie auf einem modernen Linux-System deutlich sinnvoller finden werden.

Lauffähige Beispiele
: Sie können sich nun von *http://linuxpocketguide.com/* einen Satz Dateien herunterladen und die Beispiele aus dem (englischsprachigen) Buch nachstellen.

Ade, ihr GUI-Anwendungen
: Wir behandeln in diesem Buch keine Anwendungen mehr, die eine grafische Benutzeroberfläche haben, wie zum Beispiel Bildbearbeiter und Webbrowser. Stattdessen konzentrieren wir uns mehr auf die Befehle. Hilfe zu den grafischen Anwendungen finden Sie heutzutage leicht im Internet.

Was ist Linux?

Linux ist ein beliebtes Open-Source-Betriebssystem, das in direkter Konkurrenz zu Microsoft Windows und Mac OS X steht. Wie diese besitzt auch Linux eine grafische Benutzeroberfläche mit Fenstern, Symbolen und Mausbedienung. Aber um alle Möglichkeiten von Linux ausschöpfen zu können, brauchen Sie die Kommandozeilenschnittstelle – die sogenannte *Shell* – zum Eingeben und Ausführen von Befehlen wie dem oben gezeigten `wc`.

Auch Windows- und Mac OS X-Computer können über die Kommandozeile betrieben werden (Windows über seine `cmd`- und PowerShell-Befehlswerkzeuge und OS X über das Programm Terminal), die meisten ihrer Benutzer kommen jedoch auch ohne das Eingeben von Befehlen sehr gut klar. Unter Linux ist die Shell hingegen wichtig. Wenn Sie Linux ohne Shell benutzen, verpassen Sie etwas.

Was ist eine Distribution?

Linux ist außerordentlich konfigurierbar und umfasst Tausende von Programmen. Aus diesem Grund sind verschiedene Varianten von Linux entstanden, die unterschiedlichen Bedürfnissen und Geschmäckern dienen. Sie alle besitzen bestimmte identische Kernkomponenten, sehen aber möglicherweise unterschiedlich aus und enthalten unterschiedliche Programme und Dateien. Diese Varianten bezeichnet man jeweils als *Distribution*. Beliebte Distributionen sind Ubuntu Linux, OpenSUSE, Mint, Red Hat Enterprise Linux und andere. Dieses Buch behandelt Kernmaterial, das für alle Distributionen gelten sollte.

Was ist ein Befehl?

Ein Linux-Befehl besteht typischerweise aus einem **Programmnamen**, gefolgt von **Optionen** und **Argumenten**, die in einer Shell eingetippt werden, etwa so:

```
wc -l meinedatei
```

Der Programmname (`wc`, für »word count«) bezieht sich auf ein Programm, das sich irgendwo auf der Festplatte befindet und das von

der Shell gesucht und ausgeführt wird. Optionen, die normalerweise mit einem Bindestrich beginnen, beeinflussen das Verhalten des Programms. Im vorangegangenen Befehl sagt die Option `-l` dem Programm `wc`, dass es Zeilen und nicht Wörter zählen soll. Das Argument `meinedatei` gibt die Datei an, die `wc` lesen und verarbeiten soll.

Befehle können mehrere Optionen und Argumente haben. Es ist möglich, Optionen einzeln anzugeben:

```
$ wc -l -w meinedatei                zwei einzelne Optionen
```

oder kombiniert hinter einem einzelnen Bindestrich:

```
$ wc -lw meinedatei                  identisch mit -l -w
```

Allerdings erkennen manche Programme kombinierte Optionen nicht. Mehrere Argumente sind auch okay:

```
$ wc -l meinedatei1 meinedatei2      Zählt Zeilen in zwei Dateien.
```

Optionen sind nicht standardisiert. Sie können aus einem Bindestrich und einem Buchstaben (zum Beispiel `-l`) oder zwei Bindestrichen und einem Wort (`--lines`) bestehen oder auch ein ganz anderes Format besitzen. Der gleiche Optionsbuchstabe (etwa `-l`) kann in unterschiedlichen Programmen unterschiedliche Bedeutungen haben: In `wc` bedeutet `-l` »zähle die Textzeilen (lines)«, in `ls`steht `-l` für »erzeuge längere Ausgaben (long)«. Zwei Programme können auch unterschiedliche Optionen für dieselbe Sache verwenden, wie etwa das Ausführen ohne Ausgaben: `-q` und `-s` für »leise ausführen (run quietly bzw. run silently)«. Auf manche Optionen folgt ein Wert wie zum Beispiel `-s 10`, wobei das Leerzeichen dazwischen nicht immer erforderlich ist (`-s10`).

Genauso sind Argumente nicht standardisiert. Normalerweise repräsentieren sie Dateinamen zur Ein- oder Ausgabe, es kann sich aber auch um andere Dinge handeln, etwa um Verzeichnisnamen oder reguläre Ausdrücke.

Befehle können interessanter sein als ein einzelnes Programm mit Optionen:

- Befehle können mehrere Programme gleichzeitig ausführen, entweder der Reihe nach (ein Programm nach dem anderen) oder in einer *Pipeline*, bei der die Ausgabe eines Befehls zur Eingabe des nächsten wird. Linux-Experten nutzen ständig Pipelines.
- In die Linux-Kommandozeilenschnittstelle – der **Shell** – ist eine Programmiersprache eingebaut. Anstelle eines Befehls, der sagt: »Führe dieses Programm aus«, könnte sie sagen: »Führe dieses Programm aus, schreibe seine Ausgabe in eine Datei meiner Wahl, und wenn ein Fehler auftritt, schicke mir eine E-Mail mit dem Ergebnis.«

Shell-Prompts

Bevor Sie einen Befehl eingeben können, müssen Sie darauf warten, dass die Shell ein bestimmtes Symbol ausgibt – den **Prompt**. Ein Prompt bedeutet: »Ich warte auf deinen nächsten Befehl.« Prompts kann es in allen Formen und Farben geben, es hängt davon ab, wie Ihre Shell konfiguriert ist. Ihr Prompt kann ein Dollarzeichen sein:

```
$
```

Es kann sich ebenfalls um einen komplexen String handeln, in dem Computername, Benutzer und eventuell andere Informationen und Symbole enthalten sind:

```
myhost:~smith$
```

Der Prompt kann aber auch ganz anders aussehen. Allen ist gemein, dass die Shell nun auf Ihre Benutzereingabe wartet.

In diesem Buch werden wir für den Prompt das Symbol → (einen Pfeil) verwenden, damit Sie ihn nicht unabsichtlich als Teil eines Befehls missverstehen. So sieht der Prompt gefolgt von einem Befehl aus:

```
→ wc -l meinedatei
```

Manche Programme geben während ihrer Ausführung Text aus. Um Ihren (eigentippten) Befehl von der (nicht eingetippten) Ausgabe zu unterscheiden, werden wir ihn fett gedruckt anzeigen:

```
→ wc -l meinedatei      der von Ihnen eingegebene Befehl
18 meinedatei           die dadurch erzeugte Ausgabe
```

Manche Befehle in diesem Buch kann nur ein **Administrator** ausführen – ein besonderer Benutzer, der die Berechtigung hat, alles mit dem System zu machen. (Solch ein Benutzer wird auch als **Superuser** oder **root** bezeichnet.) In diesen Fällen schreiben wir vor den Befehl noch ein sudo:

```
→ sudo Hier steht der Superuser-Befehl
```

Wir werden sudo noch ausführlicher in »Administrator werden« auf Seite 169 behandeln. Aktuell müssen Sie nur wissen, dass sudo Sie sehr mächtig macht und manchmal nach Ihrem Passwort fragen wird. Um zum Beispiel die Anzahl der Zeilen in einer geschützten Datei namens */etc/shadow* mit und ohne sudo zu zählen, können Sie diesen Befehl verwenden:

```
→ wc -l /etc/shadow                    Das klappt nicht.
wc: /etc/shadow: Permission denied
→ sudo wc -l /etc/shadow               Jetzt mit sudo.
Password: ********
51 /etc/shadow                         Es klappt!
```

Warmlaufen für die Befehle

Um Ihnen ein Gespür für Linux zu geben, finden Sie hier zehn einfache Befehle, die Sie direkt ausprobieren können. Sie müssen sie ganz genau wie angegeben eintippen – einschließlich korrekter Groß- und Kleinschreibung, Leerzeichen und allen Symbolen nach dem Prompt.

Einen Kalender für den April 2017 anzeigen:

```
→ cal apr 2017
     April 2017
Su Mo Tu We Th Fr Sa
                   1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30
```

Den Inhalt des Verzeichnisses */bin* ausgeben (das viele Befehle enthält):

```
→ ls /bin
bash      less        rm
bunzip2   lessecho    rmdir
busybox   lessfile    rnano
…
```

Die Anzahl der sichtbaren Elemente in Ihrem Home-Verzeichnis ausgeben (das hier durch eine spezielle Variable `HOME` vertreten wird, auf die wir später noch eingehen):

```
→ ls $HOME | wc -l
8
```

Herausfinden, wie viel Platz auf einer Partition Ihrer Festplatte belegt ist:

```
→ df -h /
Filesystem  Size  Used Avail Use% Mounted on
/dev/sdb1    78G   30G   48G  61% /
```

Anzeigen, welche Prozesse auf Ihrem Computer laufen (mit »q« beenden Sie den Befehl wieder):

```
→ top -d1
```

Die Datei */etc/hosts* ausgeben, in der die Namen und Adressen von Computern zu finden sind, aber auch Ihr Standarddrucker, sofern Sie einen eingerichtet haben:

```
→ lpr /etc/hosts
```

Wie lange sind Sie schon angemeldet?

```
→ last -1 $USER
smith  pts/7 :0  Tue Nov 10 20:12  still logged in
```

Eine Datei *sample.pdf* von der Website dieses Buchs in Ihr aktuell genutztes Verzeichnis herunterladen, ohne einen Webbrowser dafür bemühen zu müssen:

```
→ wget http://linuxpocketguide.com/sample.pdf
```

Informationen über die IP-Adresse Ihres Computers anzeigen:

```
→ ip addr show eth0
…
inet 192.168.1.47
```

Herausfinden, wem der Domainname *oreilly.com* gehört (mit der Leertaste blättern Sie seitenweise weiter, mit »q« beenden Sie den Befehl):

```
→ whois oreilly.com | less
Domain Name: OREILLY.COM
Registrar: GODADDY.COM, LLC
…
```

Und schließlich noch den Inhalt des Fensters löschen:

```
→ clear
```

Okay, das waren mehr als zehn Befehle ... Aber herzlichen Glückwunsch: Sie sind nun ein Linux-Shell-Anwender!

Wie Sie dieses Buch lesen

Wir beschreiben in diesem Buch viele Linux-Befehle. Jede Beschreibung beginnt mit einer Standardüberschrift zu dem Befehl; Abbildung 1 zeigt eine für den `ls`-Befehl (Dateien auflisten). Diese Zeile demonstriert die allgemeine Benutzung in einem einfachen Format:

`ls` [*optionen*] [*dateien*]

Das bedeutet, Sie würden »ls« eintippen, wahlweise gefolgt von Optionen und dann von Dateinamen. Die eckigen Klammern [und] tippen Sie nicht ein: Diese deuten nur an, dass ihr Inhalt optional ist. Kursiv gesetzte Wörter bedeuten, dass Sie Ihre eigenen Werte einsetzen müssen, z. B. Namen von wirklichen Dateien. Wenn Sie einen senkrechten Strich zwischen Optionen oder Argumenten sehen, die möglicherweise in Klammern zusammengefasst sind, wie:

(*datei* | *verzeichnis*)

heißt das, dass Sie die Wahl haben: Sie können als Argument entweder einen Dateinamen oder einen Verzeichnisnamen angeben.

Die besondere Überschrift enthält darüber hinaus sechs Eigenschaften des Befehls, die schwarz (unterstützt) oder grau (nicht unterstützt) gedruckt sind:

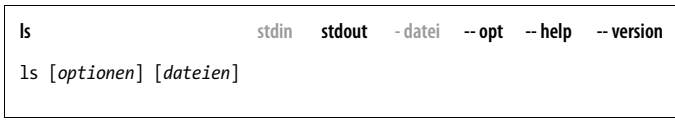


Abbildung 1: Standardbefehlsüberschrift

stdin
: Der Befehl liest normalerweise von der Standardeingabe, d. h. Ihrer Tastatur. Siehe »Eingaben und Ausgaben« auf Seite 22.

stdout
: Der Befehl schreibt normalerweise auf die Standardausgabe, d. h. Ihren Bildschirm. Siehe »Eingaben und Ausgaben« auf Seite 22.

- datei
: Wenn anstelle eines Eingabedateinamens ein Bindestrichargument (-) angegeben wird, liest der Befehl von der Standardeingabe; in gleicher Weise schreibt der Befehl auf die Standardausgabe, wenn der Bindestrich als Ausgabedateiname angegeben wird. So liest z. B. der folgende `wc`-Befehl die Dateien *datei1* und *datei2*, dann die Standardeingabe und dann *datei3*:

  ```
  → wc datei1 datei2 - datei3
  ```

-- opt
: Wenn Sie die Kommandozeilenoption -- angeben, bedeutet das »Ende der Optionen«: Alles, was anschließend auf dieser Kommandozeile folgt, ist keine Option. Das ist manchmal nötig, wenn man mit einer Datei arbeitet, deren Name mit einem Bindestrich beginnt und die ansonsten fälschlicherweise als Option behandelt werden würde. Falls Sie z. B. eine Datei namens *-foo* haben, würde der Befehl `wc -foo` fehlschlagen, weil `-foo` als (ungültige) Option behandelt werden würde. `wc -- -foo` funktioniert. Falls ein Befehl -- nicht unterstützt, können Sie dem Dateinamen den aktuellen Verzeichnispfad `./` voranstellen, damit der Bindestrich nicht mehr das erste Zeichen ist:

  ```
  → wc ./-foo
  ```

`--help`
: Die Option `--help` lässt den Befehl eine Hilfe ausgeben, die die korrekte Benutzung erläutert; anschließend wird der Befehl beendet.

`--version`
: Die Option `--version` lässt den Befehl seine Versionsinformationen ausgeben; anschließend wird der Befehl beendet.

Tastenkürzel

Im Laufe dieses Buchs benutzen wir bestimmte Symbole, um Tastenkürzel anzudeuten. Wie in vielen anderen Linux-Dokumenten verwenden wir das ^-Symbol für »Drücke und halte die Control- oder Steuerung-(Ctrl- bzw. Strg-)Taste«. So bedeutet z. B. `^D` (gesprochen »Control D«): »Drücke und halte die Control-Taste und tippe D.« Wir schreiben außerdem ESC für »Drücke die Escape-Taste«. Tasten wie Enter und die Leertaste sollten selbsterklärend sein.

Ihr Freund, der echo-Befehl

In vielen unserer Beispiele schreiben wir Informationen mit dem `echo`-Befehl auf den Bildschirm. Wir beschreiben diesen Befehl »offiziell« in »Bildschirmausgabe« auf Seite 200. `echo` ist einer der einfachsten Befehle: Er gibt lediglich seine Argumente auf der Standardausgabe aus, sobald diese Argumente von der Shell verarbeitet wurden.

```
→ echo Mein Hund hat einen Floh
Mein Hund hat einen Floh
→ echo Mein Name ist $USER              Shell-Variable USER
Mein Name ist smith
```

Lange Befehlszeilen

Manchmal wird ein Befehl so lang sein, dass er im Buch nicht mehr in eine Zeile passt. In diesen Fällen werden wir ihn auf mehrere Zeilen aufteilen, die mit einem Backslash enden:

```
→ echo Dies ist ein langer Befehl, der nicht auf \
eine Zeile passt.
Dies ist ein langer Befehl, der nicht auf eine Zeile passt.
```

Das ist nicht nur ein Hilfskonstrukt für dieses schmale Büchlein – das Backslash-Zeichen dient in der Linux-Shell tatsächlich als »Befehlsfortsetzungszeichen«. Ist Ihr Terminalfenster breiter als diese Seite, können Sie den Backslash natürlich weglassen und den Befehl vollständig in einer Zeile eingeben.

Mit diesem Buch üben

Zu diesem Buch gehört ein Satz (englischsprachiger) Dateien, die Ihnen beim Üben mit Linux helfen können. Laden Sie sich diese herunter und installieren sie auf einem Linux-Rechner, können Sie die meisten der Beispielbefehle aus diesem Buch direkt ausprobieren. Wenn Sie sie erstmalig herunterladen wollen, führen Sie diese Befehle aus:[1]

```
→ cd
→ wget http://linuxpocketguide.com/LPG-stuff.tar.gz
→ tar -xf LPG-stuff.tar.gz
```

Mit diesen Befehlen wird in Ihrem Home-Verzeichnis ein neuer Ordner *linuxpocketguide* erzeugt. Immer dann, wenn Sie nun in diesem Buch einen Beispielbefehl sehen, suchen Sie dieses Verzeichnis auf:

```
→ cd ~/linuxpocketguide
```

und probieren den Befehl aus.

Sofern Sie fleißig mit den Dateien arbeiten, werden Sie sie eventuell neu herunterladen und installieren wollen (etwa wenn Sie etwas an den Dateien geändert haben und sie zurücksetzen möchten). Das brauchen Sie aber nicht, es reicht, das Skript `reset-lpg` im Verzeichnis *linuxpocketguide* auszuführen:

```
→ cd ~/linuxpocketguide
→ bash reset-lpg
```

Haben Sie die Dateien an einem anderen Ort als in Ihrem Home-Verzeichnis abgelegt (was wahrscheinlich bedeutet, dass Sie mit

1 Alternativ können Sie die Beispieldateien auch von *https://github.com/oreillymedia/linux_pocket_guide* laden.

Verzeichnissen in Linux schon Erfahrung haben), übergeben Sie diesen Verzeichnisnamen einfach als Argument an den Befehl `reset-lpg`:

```
→ bash reset-lpg /tmp/beispiele
```

Dieser Befehl erstellt die Beispiele im Verzeichnis */tmp/beispiele/linuxpocketguide* neu oder setzt sie zurück.

Hilfe erhalten

Sollten Sie mehr Informationen brauchen, als dieses Buch bietet, haben Sie verschiedene Möglichkeiten:

Den `man` *-Befehl ausführen*
: Der `man`-Befehl zeigt für ein bestimmtes Programm eine Handbuchseite oder **Manpage** an. Um z. B. mehr über das Zählen von Wörtern in einer Datei mit `wc` zu lernen, führen Sie

  ```
  → man wc
  ```

  aus. Um anhand von Stichwörtern zu einem bestimmten Thema nach Manpages zu suchen, benutzen Sie die Option `-k`, gefolgt vom Stichwort (hier wird das Ergebnis noch an den Befehl `less` weitergegeben, um es seitenweise anzuzeigen – mit der Leertaste blättern Sie vorwärts, mit »q« beenden Sie den Befehl):

  ```
  → man -k database | less
  ```

Den `info` *-Befehl ausführen*
: Der `info`-Befehl ist ein erweitertes Hypertext-Hilfesystem, das viele Linux-Programme behandelt.

  ```
  → info ls
  ```

  Es gibt für `info` einige nützliche Tastenkürzel:

  - Um Hilfe zu erhalten, tippen Sie `h`.
  - Zum Beenden tippen Sie `q`.
  - Um seitenweise vorwärts oder zurückzublättern, nutzen Sie die Leertaste oder die Backspace-Taste.
  - Um zwischen Hyperlinks zu springen, drücken Sie die Tab-Taste.
  - Um einem Hyperlink zu folgen, drücken Sie Enter.

Sollte `info` keine Dokumentation zu einem Programm besitzen, zeigt es die Manpage des Programms an. Eine Auflistung der verfügbaren Dokumentationen erhalten Sie, wenn Sie `info` allein eintippen. Um zu lernen, wie Sie das Infosystem nutzen, geben Sie `info info` ein.

Die `--help`-Option benutzen (falls vorhanden)
: Viele Linux-Befehle reagieren auf die Option `--help`, indem sie eine kurze Hilfemeldung ausgeben. Probieren Sie:

  → **`wc --help`**

  Ist die Ausgabe länger als der Bildschirm, leiten Sie sie in das `less`-Programm weiter, um sie seitenweise anzuzeigen (drücken Sie `q` zum Beenden):

  → **`wc --help | less`**

Das Verzeichnis /usr/share/doc durchsuchen
: Dieses Verzeichnis enthält unterstützende Dokumente für viele Programme, normalerweise organisiert nach Programmnamen und -version. So finden Sie z. B. Dateien für den Texteditor emacs, Version 24, vermutlich (je nach Distribution) in */usr/share/doc/emacs24*.

Distributionsspezifische Websites
: Für die meisten Linux-Distributionen gibt es eine offizielle Website, die Dokumentationen, Diskussionsforen für Fragen und Antworten und andere Ressourcen umfasst. Geben Sie einfach den Namen der Distribution (z. B. »Ubuntu«) in Ihre Lieblingssuchmaschine ein, um die Website zu finden.

Linux-Hilfe-Sites
: Auf vielen Websites werden Fragen zu Linux beantwortet, wie etwa auf *http://www.linuxquestions.org*, *http://unix.stackexchange.com*, *http://www.linuxhelp.net* und *http://www.linuxforums.org*.

Websuche
: Um eine bestimmte Linux-Fehlermeldung zu entziffern, kopieren Sie diese Meldung und fügen sie in eine Suchmaschine im Web ein. Wahrscheinlich erhalten Sie viele hilfreiche Ergebnisse.

Linux: ein erster Überblick

Linux besitzt vier wesentliche Bestandteile:

Kernel
: Der Kernel ist das systemnahe Betriebssystem, das mit Dateien, Festplatten, dem Netzwerk und anderen notwendigen Elementen umgeht, die wir als gegeben hinnehmen. Die meisten Benutzer bemerken den Kernel gar nicht.

Mitgelieferte Programme
: Es gibt Tausende von Programmen für Dateimanipulationen, Textbearbeitung, Mathematik, das Surfen im Web, Audio, Video, Computerprogrammierung, Schriftsatz, Verschlüsselung, das Brennen von DVDs u. v. m.

Shell
: Die Shell ist eine Benutzerschnittstelle zum Eintippen und Ausführen von Befehlen und zum Anzeigen der Ergebnisse. Linux besitzt verschiedene Shells: die Bourne-Shell, die Korn-Shell, die C-Shell und andere. Dieses Buch konzentriert sich auf bash, die Bourne-Again Shell, die für Benutzerkonten oft als Standard-Shell agiert. All diese Shells verfügen jedoch über ähnliche Grundfunktionen.

X
: X ist ein grafisches System, das eine Mausunterstützung sowie Fenster, Menüs, Icons und andere vertraute grafische Elemente bereitstellt. Aufbauend auf X, gibt es komplexere grafische Umgebungen. Die beliebtesten sind KDE und GNOME. Wir werden nur einige wenige Programme betrachten, die bei der Ausführung X-Fenster öffnen.

Dieses Buch konzentriert sich auf den zweiten und den dritten Teil: auf mitgelieferte Programme und auf die Shell.

Eine Shell ausführen

Sind Sie über das Netzwerk mit einem Linux-Rechner verbunden, werden Sie sich einer Shell gegenübersehen, in die Sie Befehle ein-

tippen können. Sitzen Sie hingegen direkt vor einem Linux-Rechner und melden sich an diesem an, werden Sie sehr wahrscheinlich von einem grafischen Desktop begrüßt, auf dem erst einmal keine Shell zu sehen ist. Für viele Anwender ist dies die bevorzugte Arbeitsumgebung, und die Icons und Menüs sind für einfache Aufgaben wie das Lesen von E-Mails oder das Browsen im Web völlig ausreichend. Aber um alles aus Linux herauszukitzeln, müssen Sie die grafische Oberfläche hinter sich lassen und zur Shell wechseln. Das mag zu Anfang schwieriger sein als all die bunten Icons, aber wenn Sie sich einmal daran gewöhnt haben, arbeitet es sich sehr angenehm mit der Shell und verschafft Ihnen *deutlich* mehr Macht.

Wie führen Sie nun eine Shell auf der grafischen Oberfläche aus? Nun, das kommt darauf an. Linux bietet viele verschiedene Oberflächen, von denen die bekanntesten GNOME und KDE sind, und in jedem Linux-System können sie anders zu konfigurieren sein. Es liegt an Ihnen, ein Icon oder einen Menüpunkt zu finden, mit dem Sie ein Shell-Fenster öffnen können. Suchen Sie im Haupt- oder Startmenü Ihres Systems nach einer Anwendung namens Terminal, Konsole, xterm, gnome-terminal, uxterm oder etwas Ähnlichem und starten Sie sie.

Das Fensterprogramm (Terminal, Konsole usw.) ist nicht die Shell. Es handelt sich dabei nur um ein grafisches Programm – eventuell mit eigenen, coolen Features –, in dem die Shell läuft. Und Letztere führt Ihre Befehle aus.

In Abbildung 2 ist dieser Unterschied dargestellt.

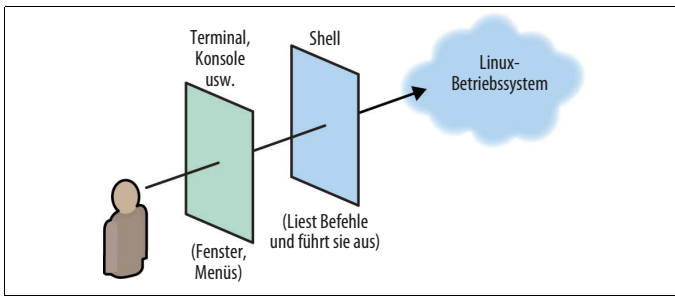


Abbildung 2: Der Unterschied zwischen einem Shell-Fenster und der Shell

Dies war nur eine schnelle Einführung. Mehr Einzelheiten besprechen wir in »Shell-Features« auf Seite 33 und behandeln leistungsstärkere Konstrukte in »Mit Shell-Skripten programmieren« auf Seite 231.

Eingaben und Ausgaben

Die meisten Linux-Befehle akzeptieren Eingaben und erzeugen Ausgaben. So erwartet zum Beispiel der Befehl `wc` die Eingabe von einer Datei, während die Ausgabe (die Anzahl der Zeilen, Wörter oder Zeichen) am Bildschirm erscheint.

Linux-Befehle sind in dieser Hinsicht sehr flexibel. Eingaben können aus Dateien oder von der **Standardeingabe** kommen, bei der es sich normalerweise um Ihre Tastatur handelt. Ausgaben werden entsprechend in Dateien oder auf die **Standardausgabe** geschrieben, die üblicherweise Ihr Shell-Fenster oder Bildschirm ist. Fehlermeldungen werden gesondert behandelt und auf der **Standardfehlerausgabe** angezeigt, die normalerweise ebenfalls Ihr Bildschirm ist. Die Standardfehlerausgabe wird allerdings von der Standardausgabe getrennt.[2]

Wir werden uns später anschauen, wie man Standardeingabe, -ausgabe und -fehlerausgabe zu oder aus Dateien oder Pipes *umleiten* kann. Erläutern wir aber zunächst ein paar Begriffe. Wenn wir sagen, ein Befehl »liest«, dann meinen wir, er liest von der Standardeingabe – solange nichts anderes gesagt wird. Und wenn ein Befehl »schreibt« oder »ausgibt«, dreht es sich immer um die Standardausgabe (es sei denn, wir beziehen uns ausdrücklich auf einen Drucker).

Benutzer und Administratoren

Linux ist ein Multi-User-Betriebssystem: Mehrere Leute können zur gleichen Zeit einen einzelnen Linux-Computer benutzen. Auf einem Computer wird jeder Benutzer durch seinen eindeutigen

2 Sie können z. B. die Standardausgabe in einer Datei erfassen und dennoch die Meldungen der Standardfehlerausgabe auf dem Bildschirm erscheinen lassen.

Benutzernamen identifiziert, wie etwa »smith« oder »lustigertyp«, und er besitzt einen (halbwegs) privaten Teil des Systems für seine Arbeit.

Darüber hinaus gibt es einen besonderen Benutzer namens **root** – den **Administrator** –, der das Recht besitzt, alles auf dem System zu machen. Normale Benutzer haben eingeschränkte Rechte: Sie dürfen zwar die meisten Programme ausführen, im Allgemeinen aber können sie nur die Dateien modifizieren, die ihnen gehören. Der Administrator dagegen kann alle Dateien erzeugen, verändern oder löschen und darf alle Programme ausführen – wir gehen darauf ausführlicher in »Administrator werden« auf Seite 169 ein.

Das Dateisystem

Um ein Linux-System benutzen zu können, müssen Sie sich mit **Linux-Dateien** und -**Verzeichnissen** (d. h. Ordnern) vertraut machen. In einem »Fenster und Icons«-System sind die Dateien und Verzeichnisse auf dem Bildschirm eindeutig zu erkennen. Auf einem Kommandozeilensystem wie der Linux-Shell hingegen sind die gleichen Dateien und Verzeichnisse zwar vorhanden, aber nicht ständig sichtbar, sodass Sie sich häufig merken müssen, in welchem Verzeichnis Sie sich gerade befinden und in welcher Beziehung dieses zu anderen Verzeichnissen steht. Sie verwenden Shell-Befehle wie `cd` und `pwd`, um sich zwischen Verzeichnissen zu »bewegen« und den Überblick darüber zu bewahren, wo Sie gerade sind.

Kommen wir zu einigen Begriffen. Wie wir gesagt haben, sind Linux-Dateien in Verzeichnissen organisiert. Die Verzeichnisse bilden eine Hierarchie (auch **Baum** genannt), wie in Abbildung 3 gezeigt: Ein Verzeichnis kann andere Verzeichnisse enthalten, die sogenannten **Unterverzeichnisse**, die wiederum andere Dateien und Unterverzeichnisse enthalten können usw. – bis in die Unendlichkeit. Das oberste Verzeichnis wird als **Root-Verzeichnis** bezeichnet und durch einen Schrägstrich oder auch Slash (/) gekennzeichnet.[3]

3 In Linux stammen alle Dateien und Verzeichnisse von Root ab. Das ist anders als unter Windows oder DOS, wo unterschiedliche Geräte über Laufwerkbuchstaben angesprochen werden.

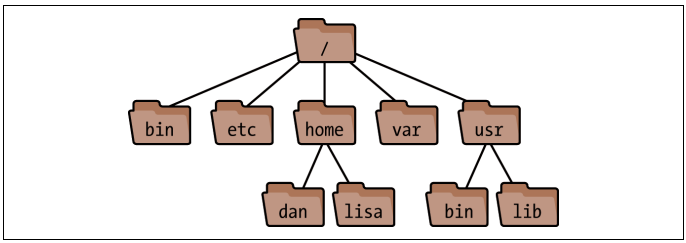


Abbildung 3: Ein Teil eines Linux-Dateisystems. Der Root-Ordner befindet sich ganz oben. Der absolute Pfad zum Ordner »dan« lautet /home/dan.

Wir beziehen uns auf die Dateien und Verzeichnisse mittels einer »Namen-und-Schrägstriche-Syntax«, die **Pfad** genannt wird. Zum Beispiel bezeichnet Pfad

```
/eins/zwei/drei/vier
```

das Root-Verzeichnis /, das ein Verzeichnis namens *eins* enthält, in dem sich das Verzeichnis *zwei* befindet, das wiederum ein Verzeichnis *drei* enthält, in dem schließlich die letzte Datei oder das letzte Verzeichnis zu finden ist, nämlich *vier*. Beginnt ein Pfad mit dem Root-Verzeichnis, bezeichnet man ihn als **absoluten** Pfad.

Pfade müssen nicht absolut sein – sie können relativ zu einem anderen Verzeichnis angegeben werden. In Abbildung 3 gibt es zwei verschiedene Verzeichnisse namens *bin*, deren absolute Pfade */bin* und */usr/bin* lauten. Beziehen wir uns einfach auf das »*bin*-Verzeichnis«, ist nicht klar, welches wir meinen (und es kann noch viel mehr *bin*-Verzeichnisse geben). Wir brauchen mehr Kontext. Immer wenn Sie sich auf einen Pfad beziehen, der nicht mit einem Schrägstrich beginnt – zum Beispiel *bin* –, wird dieser als **relativer** Pfad bezeichnet.

Damit ein relativer Pfad Sinn ergibt, müssen Sie wissen, wo Sie sich im Linux-Dateisystem befinden. Diese Position wird als Ihr **aktuelles Arbeitsverzeichnis** (oder einfach als aktuelles Verzeichnis) bezeichnet. Jede Shell besitzt ein aktuelles Arbeitsverzeichnis, und wenn Sie Befehle in dieser Shell ausführen, arbeiten diese relativ zu dem Verzeichnis. Befindet sich Ihre Shell zum Beispiel »im« Ver-

zeichnis */usr* und führen Sie einen Befehl aus, der sich auf einen relativen Pfad *bin* bezieht, geht es in Wirklichkeit um den Pfad */usr/bin*. Ganz allgemein: Ist Ihr aktuelles Verzeichnis */eins/zwei/drei*, würde ein relativer Pfad *a/b/c* dem absoluten Pfad */eins/zwei/drei/a/b/c* entsprechen.

Zwei besondere relative Pfade werden mit . (einem einzelnen Punkt) und .. (zwei Punkten hintereinander) gekennzeichnet. Der erste Ausdruck steht für Ihr aktuelles Verzeichnis, der zweite Ausdruck bezieht sich auf das **Elternverzeichnis**, das sich eine Ebene darüber befindet. Ist Ihr aktuelles Verzeichnis also */eins/zwei/drei*, bezieht sich . auf dieses Verzeichnis und .. auf */eins/zwei*.

Sie »bewegen« Ihre Shell von einem zum anderen Verzeichnis mit dem cd-Befehl:

```
→ cd /usr/local/bin
```

Technischer ausgedrückt: Dieser Befehl ändert das aktuelle Arbeitsverzeichnis Ihrer Shell auf */usr/local/bin*. Das ist eine absolute Änderung (da das Verzeichnis mit / beginnt). Haben Sie das Verzeichnis mit den Buchbeispielen in Ihrem Home-Verzeichnis installiert, können Sie dort jederzeit wie folgt hinspringen:

```
→ cd ~/linuxpocketguide
```

(Die Tilde ist hier eine Abkürzung, auf die wir im nächsten Abschnitt zu sprechen kommen). Natürlich können Sie auch relative Bewegungen machen:

```
→ cd d              Wechsle ins Unterverzeichnis d.
→ cd ../meinVerz    Gehe nach oben zu meinem Elternverzeichnis
                    und von dort in das Verzeichnis meinVerz.
```

Datei- und Verzeichnisnamen können die meisten Zeichen enthalten, die man auch erwartet: Groß- und Kleinbuchstaben,[4] Zahlen, Punkte, Bindestriche, Unterstriche und die meisten Symbole (aber nicht »/«, weil dieser für das Trennen der Verzeichnisse reserviert ist). Vermeiden Sie im praktischen Einsatz jedoch auch Leerzei-

4 Linux-Dateinamen unterscheiden zwischen Klein- und Großschreibung, sie sind also nicht äquivalent.

chen, Sternchen, Dollarzeichen, Klammern und andere Zeichen, die für die Shell eine besondere Bedeutung haben. Ansonsten müssen Sie diese Zeichen schützen (siehe »Quoting« auf Seite 42).

Home-Verzeichnisse

Die persönlichen Dateien der Benutzer sind oft im Verzeichnis */home* (für normale Benutzer) oder */root* (für den Administrator) zu finden. Ihr Home-Verzeichnis ist typischerweise */home/Ihr-Benutzername*, also: */home/smith*, */home/jones* usw. Es gibt verschiedene Möglichkeiten, das Home-Verzeichnis zu finden oder anzusprechen:

`cd`

Ohne Argumente benutzt, bringt der `cd`-Befehl Sie zu Ihrem Home-Verzeichnis zurück (d. h., er stellt das Arbeitsverzeichnis der Shell entsprechend ein).

`HOME` *-Variable*

Die Umgebungsvariable `HOME` (siehe »Shell-Variablen« auf Seite 37) enthält den Namen Ihres Home-Verzeichnisses:

```
→ echo $HOME          Der echo-Befehl gibt sein Argument aus.
/home/smith
```

`~`

Wird eine Tilde anstelle eines Verzeichnisses benutzt, wird sie von der Shell durch den Namen Ihres Home-Verzeichnisses ersetzt.

```
→ echo ~
/home/smith
```

Folgt auf die Tilde ein Schrägstrich, ist der Pfad relativ zu `HOME`:

```
→ echo ~/linuxpocketguide
/home/smith/linuxpocketguide
```

Folgt auf die Tilde ein Benutzername (wie in *~fred*), erweitert die Shell diesen String zum Home-Verzeichnis des Benutzers:

```
→ cd ~fred
→ pwd                 der Befehl zum Ausgeben des Arbeitsverzeichnisses
home/fred             (print working directory)
```