Linux intern verstehen

Ein systemnaher Einblick in Aufbau, Kernel und Systemkomponenten

Kapitel 1

Linux-Grundlagen

Ein Betriebssystem hat im Hintergrund mehr zu tun, als nur Mausklicks entgegenzunehmen. Es startet den Computer sowie die nötigen Dienste, damit alles läuft, wie es soll. Meist stellt es eine grafische Oberfläche bereit, es steuert die Prozesse und regelt die Rechte der Benutzer.

Was im Hintergrund eines Betriebssystems passiert, ist recht komplex – in diesem Kapitel lesen Sie mehr darüber, welche Aufgaben Linux als Betriebssystem hat. Sie lernen die Verzeichnis-Hierarchie kennen, um zu verstehen, wo Sie die wichtigsten Dateien finden, und erfahren, wie Sie mit der Shell arbeiten.

1.1 Der Linux-Kernel

Der Kernel ist der Kern eines jeden Betriebssystems – unter Linux natürlich der Linux-Kernel. Der Kernel ist der erste Bestandteil, der vom Betriebssystem gestartet wird. Alles andere, was im Betriebssystem läuft, wird von diesem verwaltet bzw. gesteuert. Im Folgenden finden Sie die vier wichtigsten Aufgaben des Kernels, anschließend werden diese genauer beschrieben.

- **Prozess-Management** – Der Kernel verwaltet die Prozesse, die vom Betriebssystem oder vom Benutzer gestartet werden.
- **Hauptspeicher verwalten** – Der Kernel gibt an, welche Prozesse wie viel vom vorhandenen Hauptspeicher (auch Arbeitsspeicher oder RAM genannt) nutzen können.
- **Hardware verwalten** – Der Kernel ist die Schnittstelle zwischen Betriebssystem und Hardware. Hierfür nutzt der Kernel Treiber.
- **Systemaufrufe annehmen und abgeben** – Damit Anwendungen funktionieren, muss der Kernel von Prozessen bzw. Anwendungen Befehle – sogenannte *Systemaufrufe* – entgegennehmen und ebensolche den Prozessen/Anwendungen wieder zurücksenden.

1.1.1 Prozess-Management

In einem Betriebssystem laufen zur selben Zeit viele Prozesse, damit alles funktioniert, wie es soll. Starten Sie einen Webbrowser, ist dies mindestens ein Prozess (oft nutzen Anwendungen auch mehrere Prozesse für verschiedene Aufgaben).

Im Hintergrund laufen parallel jedoch viele weitere Prozesse – etwa einer für die Synchronisierung der Uhrzeit mit einem Server im Internet, ein anderer stellt die grafische Oberfläche bereit und andere warten darauf, dass Sie einen USB-Stick anschließen, um diesen nutzbar zu machen.

Der Kernel startet die nötigen Prozesse zur richtigen Zeit, pausiert solche, die gerade nicht benötigt werden, aber vielleicht später noch gebraucht werden. Natürlich beendet der Kernel solche Prozesse, die nicht mehr benötigt werden, auch wieder.

Er steuert auch, welche Prozesse wie viel Zeit von welchem Kern der CPU benutzen dürfen.

Man könnte meinen, alles am Computer würde zur selben Zeit geschehen, dem ist jedoch nicht so. Ein Prozessor kann nur eine gewisse Menge an Aufgaben erledigen (genauer gesagt errechnen). Wie schon beschrieben, laufen im Betriebssystem aber viele Prozesse zur selben Zeit. Der Kernel teilt ein, für wie lange ein Prozess die CPU nutzen kann. Oft wird ein Prozess kurz unterbrochen, damit ein anderer kurz die CPU nutzen kann, und wieder zurück. Dies geschieht in sehr kurzen Abständen, sodass man am Bildschirm von diesen kurzen Unterbrechungen nichts mitbekommt. Hier hat aber nicht der Kernel alleine die Kontrolle, sondern auch die CPU selbst hat einen gewissen Einfluss.

1.1.2 Hauptspeicher verwalten

Der Hauptspeicher, auch *Arbeitsspeicher* oder einfach *RAM* genannt, wird in aktuellen Computern zwar immer größer, doch auch dieser wird vom Kernel verwaltet, auch wenn moderne CPUs dem Kernel dabei helfen.

Jeder Prozess braucht nicht nur Zeit der CPU, sondern auch Platz im Hauptspeicher. Der Hauptspeicher ist im Grunde ein sehr schnelles Speichermedium, das jedoch nur begrenzt Platz hat. Prozesse können ihre Daten zwar auch auf der SSD oder der noch langsameren Festplatte speichern, können dadurch jedoch nur viel langsamer abgerufen werden, um von der CPU verarbeitet zu werden.

Damit der Hauptspeicher nicht irgendwann voll wird und zu berechnende Daten auf den langsameren Speicher ausgelagert werden, muss der Kernel den Hauptspeicher für jeden Prozess einteilen. Der Kernel regelt also, wie viel Platz jeder Prozess im Hauptspeicher bekommt sowie mit welchen anderen Prozessen diese Daten geteilt werden dürfen und wann die Daten aus diesem Speicher wieder entfernt werden.

Der Kernel selbst nutzt natürlich auch einen gewissen Bereich im Hauptspeicher – dies ist der sogenannte *Kernel-Space*.

1.1.3 Hardware verwalten

Der Kernel macht auch jegliche Hardware benutzbar – egal ob Tastatur oder Maus, Bildschirm, Webcam, USB-Stick und so weiter. Er sorgt dafür, dass Sie diese Geräte nutzen können.

Der Kernel erkennt eingebaute Hardware und wartet darauf, dass weitere Hardware – etwa über USB – angeschlossen wird, und nutzt dann die passenden Treiber, um mit der Hardware arbeiten zu können.

1.1.4 Systemaufrufe annehmen und abgeben

Innerhalb des Betriebssystems herrscht ständige Kommunikation. Starten Sie etwa eine Anwendung per Mausklick aus dem Anwendungsmenü oder per Befehl auf der Shell, muss der Kernel erfahren, was er tun soll.

Zur Kommunikation im Betriebssystem werden sogenannte *Systemaufrufe* genutzt. Systemaufrufe bestehen einfach gesagt aus Befehlen, mit denen der Kernel und die Anwendungen bzw. Prozesse umgehen können.

1.2 Die Verzeichnis-Hierarchie

Die Verzeichnis-Struktur von Linux ist anders aufgebaut als die von Windows. Unter Windows werden Partitionen oft *Laufwerke* genannt und haben als Bezeichnung einen Buchstaben. So nennt sich die Systempartition etwa »C«.

Unter Linux gibt es solche eigenen Laufwerke nicht. Stattdessen werden alle Partitionen unter dem sogenannten *Wurzelverzeichnis* – auch einfach »/« genannt – eingebunden. So gesehen gibt es unter Linux nur ein wichtiges Verzeichnis – eben das Wurzelverzeichnis. Abbildung 1.1 zeigt Ihnen die obersten Schichten des Wurzelverzeichnisses.

Eine genauere Beschreibung der Verzeichnisse finden Sie in folgender Liste:

- **/** – Das Wurzelverzeichnis, fälschlicherweise auch gerne »root« genannt. Alle möglichen weiteren Partitionen werden in diesem Verzeichnis eingebunden. Auf die meisten Unterverzeichnisse hat der normale Benutzer zwar lesenden Zugriff, kann aber nichts anlegen oder verändern.
- **bin** – Hier liegen die ausführbaren Dateien für Anwendungen, die darin liegenden Dateien kann jeder benutzen.
- **boot** – Das Verzeichnis für den Bootloader und seiner Konfiguration.
- **dev** – Dieses Verzeichnis beinhaltet sogenannte *Gerätedateien*, über die der Kernel mit der Hardware kommuniziert.

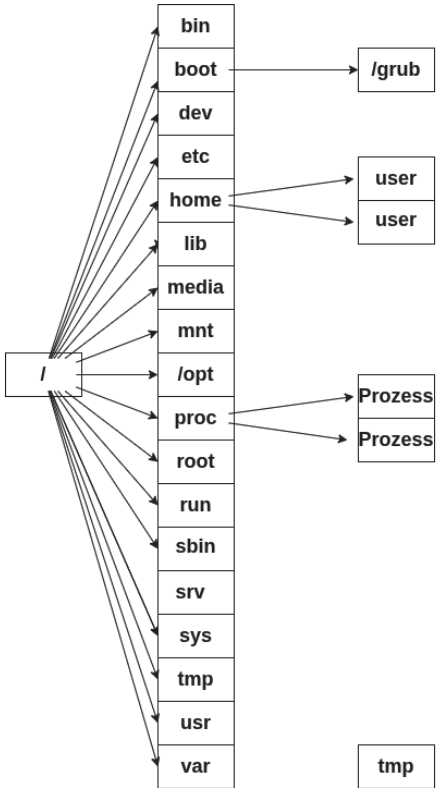

Abb. 1.1: Die Linux-Verzeichnis-Hierarchie

- **etc** – Eines der wichtigsten Verzeichnisse. Hier finden Sie in Textdateien und Unterverzeichnissen die Konfiguration des Systems sowie Dienste (Services) und die Grundkonfiguration bestimmter Anwendungen der Benutzer.
- **home** – Das Benutzer-Verzeichnis. Für jeden Benutzer gibt es hier ein Unterverzeichnis, das nach dem jeweiligen Benutzer benannt ist. Nur hier hat der normale Benutzer ohne administrative Rechte uneingeschränkten Zugriff.
- **lib**, **lib32** und **lib64** – Hier sind Bibliotheken des Betriebssystems zu finden, der Unterordner »firmware« beinhaltet für Treiber wichtige Dateien.
- **media** – In diesem Verzeichnis werden Wechselmedien eingehängt, die vom Betriebssystem automatisch eingebunden werden. Etwa USB-Sticks und externe Festplatten sowie CDs oder DVDs.
- **mnt** – Ähnlich wie »media«, hier werden jedoch keine Medien eingebunden, die automatisch vom System eingebunden werden. In diesem Verzeichnis bindet der Administrator etwa Verzeichnisse aus dem Netzwerk ein.
- **opt** – Das Verzeichnis für optionale Software, etwa für solche Software, die nicht aus den Quellen der genutzten Distribution stammt.

- **proc** – Dieses Verzeichnis liegt nicht auf der Festplatte, sondern im Hauptspeicher (Arbeitsspeicher), es handelt sich also um ein sogenanntes *virtuelles Dateisystem*. Es beinhaltet in Textdateien und Unterverzeichnissen Live-Informationen zum laufenden System und zu Prozessen.
- **root** – Das Verzeichnis des Administrators namens »root«. Hier liegt die Konfiguration der Anwendungen, die dieser benutzt. Nur root selbst hat hier Zugriff.
- **run** – Hier finden Sie in Unterverzeichnissen Dateien zu laufenden Prozessen, also solche Dateien, die von diesen Prozessen benötigt werden.
- **sbin** – In diesem Verzeichnis liegen die ausführbaren Dateien, die nur mit administrativen Rechten ausgeführt werden können, also vom Administrator root.
- **srv** – Hier legen Programmierer oft ihre Entwicklungen und deren Konfiguration ab (etwa Skripte, nicht fertige Anwendungen, Dateien für Docker und Podman).
- **sys** – Ähnlich wie unter »proc« liegen hier Informationen zum laufenden System in Textdateien und Unterverzeichnissen. Auch hierbei handelt es sich um ein virtuelles Dateisystem, da die darin liegenden Daten sich nicht auf der Festplatte, sondern im Hauptspeicher befinden.
- **tmp** – Das Verzeichnis der temporären Dateien. Alle Daten, die vom System und von Benutzern benötigt werden, werden in diesem Verzeichnis zwischengespeichert. Auf modernen Distributionen liegt dieses Verzeichnis nicht auf der Festplatte, sondern als virtuelles Dateisystem im Hauptspeicher.
- **usr** – In diesem Verzeichnis finden Sie die installierten Anwendungen und deren Dateien. Im Unterverzeichnis »bin« finden sich auch einige der ausführbaren Dateien dieser Anwendungen.
- **var** – Im Unterverzeichnis »log« finden Sie alle Log-Meldungen des Systems und installierter Anwendungen. Ist etwa ein Webserver installiert, ist hier auch »var/www/html« zu finden – dort liegen dann beispielsweise die Webseiten.

1.3 Die Shell – arbeiten auf dem Terminal

Wollen Sie sich tiefergehend mit Linux befassen, führt kein Weg an der Shell – also an der Kommandozeile – vorbei. Das Arbeiten auf der Shell zu erlernen, muss kein steiniger Weg sein. Mit einigen wichtigen Befehlen und dem Wissen, wie Sie weitere Hilfe am Terminal finden, werden Sie schnell zum Profi.

Die Shell ist ein sogenannter *Shell-Interpreter*, wir werden in diesem Buch jedoch bei der Bezeichnung »Shell« bleiben. Terminal, Konsole und ähnliche Namen sind nur die Bezeichnungen für die grafischen Fenster der jeweiligen Desktop-Umgebung. Die Shell nimmt Befehle von Ihnen entgegen und übersetzt diese in

eine für den Kernel verständliche Sprache und gibt (oft) für den Menschen verständliche Ausgaben zurück.

Unter Linux finden Sie viele Shells: die Bash, Zsh, Fish, Dash – man kann sie nicht alle aufzählen. Grundsätzlich ist die Bash (Bourne Again Shell) auf fast jeder Distribution vorinstalliert, sie wird auch am meisten genutzt. Aus diesem Grund wird in diesem Buch auch die Bash verwendet. Alle Shells haben ihre kleinen Eigenheiten. Alles, was Sie in diesem Kapitel und in diesem Buch lernen, können Sie jedoch mit jeder Shell ohne Anpassungen umsetzen.

Starten Sie die Shell, indem Sie das Terminal (die Konsole) der jeweiligen Desktop-Umgebung öffnen. Anschließend sehen Sie eine Ausgabe ähnlich dieser – den Eingabe-Prompt:

```
robertg@debian-hp:~$
```

Zu Beginn sehen Sie den Benutzernamen, mit dem Sie am System angemeldet sind, nach dem »@« finden Sie den Namen des Computers, nach dem Doppelpunkt wird Ihnen angezeigt, in welchem Verzeichnis Sie sich befinden – die Tilde »~« zeigt Ihnen, dass Sie sich in Ihrem Home-Verzeichnis aufhalten. Das »$«-Zeichen vermittelt Ihnen, dass Sie gerade als normaler Benutzer arbeiten und das »#«-Zeichen, dass Sie root-Rechte nutzen.

Sie können auch auf die Shell wechseln, indem Sie die grafische Oberfläche verlassen. Hierzu nutzen Sie die Tastenkombination Strg+Alt+F1 bis F7 – es ist unter den verschiedenen Distributionen unterschiedlich. Sie haben sechs Shells (sogenannte *virtuelle Terminals*) zur Verfügung. Mit einer dieser Tastenkombinationen gelangen Sie wieder auf die grafische Oberfläche zurück. Anders als bei der Shell im Fenster müssen Sie sich hier erst anmelden – dies sehen Sie an der Ausgabe:

```
login:
```

Geben Sie Ihren Benutzernamen ein, bestätigen Sie mit ↵, anschließend werden Sie nach dem Passwort gefragt – auch dieses bestätigen Sie mit ↵. Mit der Tastenkombination Strg+d können Sie sich wieder abmelden.

Wichtig beim Arbeiten auf der Shell!

Die Shell nimmt Ihre Befehle entgegen und führt diese aus. Bei den meisten Befehlen fragt die Shell nicht nach, ob Sie den Befehl wirklich ausführen wollen! Löschen Sie eine Datei oder ein Verzeichnis auf der Shell, landet diese nicht im Papierkorb, von wo Sie diese wiederherstellen können – es wird umgehend gelöscht. Seien Sie also vorsichtig.

Die Shell unterscheidet zwischen Groß- und Kleinschreibung. Dies gilt bei Befehlen, Optionen, Dateien und Verzeichnissen. So wären Verzeichnisse mit den Namen »Dokumente« und »dokumente« zwei unterschiedliche Verzeichnisse.

1.3.1 Befehle als root (als Administrator) ausführen

Als normaler Benutzer haben Sie unter Linux nur in Ihrem eigenen Home-Verzeichnis alle Rechte – hier können Sie Dateien und Verzeichnisse erstellen, verändern und wieder löschen. Außerhalb dieses Verzeichnisses kann man als normaler Benutzer zwar viele Verzeichnisse und Dateien einsehen, aber nichts an diesen verändern oder solche löschen. Um an Systemdateien oder Systemverzeichnissen Änderungen vornehmen zu können, müssen Sie administrative Rechte erwerben – zum Administrator namens »root« werden.

Unter Linux gibt es zwei Arten, administrative Rechte zu erwerben. Hierzu können folgende zwei Befehle genutzt werden:

```
su
sudo
```

Auf aktuellen auf Ubuntu basierenden Distributionen kommt noch folgender Befehl hinzu:

```
sudo-rs
```

Welche Befehle Sie ohne weitere Umstände nutzen können, hängt von der genutzten Distribution ab und davon, welche Optionen Sie bei der Installation gewählt haben. Unter einem traditionellem Debian (nicht als Live-System) wird bei der Installation das Anlegen eines root-Passworts verlangt, bei den meisten anderen Distributionen ist dies optional – nach der Installation können Sie ein solches immer anlegen.

In Tabelle 1.1 finden Sie die grundsätzlichen Unterschiede zwischen `su` und `sudo`.

Befehl	su	sudo
Zweck	Wechselt zwischen Benutzern	Führt einen einzelnen Befehl als root aus
Passwort	Ein root-Passwort für alle Benutzer	Jeder Benutzer nutzt sein eigenes Passwort
Sitzung	root-Rechte sind aktiv, bis man sich abmeldet	root-Rechte sind nur einige Minuten aktiv

Tabelle 1.1: Die Unterschiede zwischen `su` und `sudo`

Befehl	su	sudo
Kontrolle	Jeder Benutzer, der das Passwort kennt, kann administrative Befehle ausführen	Sie können bestimmen, welcher Benutzer welche Befehle nutzen darf
Sicherheit	Weniger sicher, wenn mehrere Benutzer das Passwort kennen	Sicherer, da Sie festlegen können, welcher Benutzer welche Befehle ausführen darf

Tabelle 1.1: Die Unterschiede zwischen `su` und `sudo` (Forts.)

Grundsätzlich ist root ein eigener Benutzer mit eigenem Passwort. Jeder Benutzer, der dessen Passwort kennt, kann zu root werden und somit alle administrativen Befehle nutzen, die verfügbar sind. Um zu root zu werden, nutzen Sie den Befehl `su`. Wurde die Linux-Distribution ohne root-Passwort installiert, kann der erste erstellte Benutzer mit seinem Passwort administrative Befehle ausführen und weiteren Benutzern zu administrativen Befehlen verhelfen (dazu mehr in Abschnitt 7.5).

Administrative Rechte mit su

Mit dem Befehl `su` werden Sie tatsächlich zu einem anderen Benutzer, und zwar zum Benutzer »root« (Substitute User). Sie haben damit alle Rechte im Linux-Betriebssystem und können das Betriebssystem so gesehen auch komplett unbrauchbar machen – und jeder Benutzer, der das Passwort von root kennt, kann dies ebenso.

Um zum Benutzer root zu werden, geben Sie im Terminal den Befehl `su` ein und bestätigen mit ↵, anschließend werden Sie nach dem Passwort von root gefragt:

```
su
Passwort:
```

Der Eingabe-Prompt ändert sich von:

```
benutzername@computername:~$
```

zu:

```
root@computername:/home/benutzername/#
```

Das Doppelkreuz »#« zeigt Ihnen, dass Sie jetzt administrative Rechte haben – auch root-Rechte genannt – und uneingeschränkt arbeiten können. Hierfür gilt

keine Zeitvorgabe. Erst wenn Sie sich mit der Tastenkombination Strg+d abmelden, arbeiten Sie wieder als normaler Benutzer auf der Shell.

Administrative Rechte mit sudo

Mit dem Befehl sudo führen Sie einfach gesagt nur kurzzeitig Befehle mit administrativen Rechten aus. Mittels sudo können Sie auch anderen Benutzern dasselbe Recht geben, administrative Befehle auszuführen – aber Sie können diese Rechte auch einschränken und anderen Benutzern etwa nur das Recht geben, gewisse administrative Befehle auszuführen (dazu mehr in Abschnitt 7.5).

Der Befehl sudo lässt sich auf mehrere Arten nutzen, dies dient auch ein wenig der Sicherheit. Erstens werden die administrativen Rechte nach einiger Zeit wieder entzogen (nach ungefähr fünf Minuten) und zweitens lassen sich die Rechte etwa auf ein Verzeichnis beschränken.

Grundsätzlich führt man mit dem Befehl sudo einen einzelnen Befehl aus. Mit folgendem Befehl würden Sie etwa den Shell-Editor Nano mit administrativen Rechten öffnen:

```
sudo nano
```

Der auszuführende Befehl wird also nach dem Befehl sudo geschrieben. Nach der Bestätigung mit ↵ werden Sie nach Ihrem Passwort gefragt. Für fünf weitere Minuten können Sie Befehle mit voranstehendem sudo ausführen, ohne nach dem Passwort gefragt zu werden.

Fügen Sie dem Befehl sudo die Option -i hinzu, können Sie dauerhaft administrative Befehle ohne zeitliche Einschränkungen ausführen – in diesem Fall geben Sie jedoch keinen anschließenden Befehl an:

```
sudo -i
```

Jetzt können Sie administrative Befehle ohne vorangehendes sudo ausführen, bis Sie sich mit Strg+d abmelden.

1.3.2 Grundlegende Shell-Befehle

Das Wichtigste voran: Die Shell unter Linux unterscheidet zwischen Groß- und Kleinbuchstaben! Dies ist sowohl bei Befehlen der Fall als auch bei Optionen, Dateien und Verzeichnissen und gilt genauso bei Benutzernamen und Passwörtern.

Ebenfalls wichtig: Denken Sie gut darüber nach, was Sie tun möchten, und handeln Sie danach! Auch wenn sich in den letzten Jahren viel verbessert hat, verzeiht die Shell keine Fehler: Eine gelöschte Datei wird tatsächlich gelöscht.

Befehle vervollständigen lassen

Die Shell hilft Ihnen bei der Arbeit. Sie müssen nicht alle Befehle vollständig ausschreiben, dies gilt auch bei Verzeichnissen und Dateien.

Geben Sie die ersten Zeichen eines Befehls ein, eines Verzeichnisses oder einer Datei, und drücken Sie anschließend die Taste Tabulator, vervollständigt die Shell den Befehl, das Verzeichnis oder die Datei automatisch. Sind mehrere Möglichkeiten vorhanden, die mit denselben Zeichen beginnen, tut die Shell dies nicht – in diesem Fall drücken Sie die Tabulator-Taste nochmals und die Shell zeigt Ihnen die vorhandenen Möglichkeiten:

```
ls Do
Docker/ Dokumente/ Downloads/
```

Vervollständigen Sie die Eingabe so lange, bis keine Gemeinsamkeiten mehr vorhanden sind, anschließend vervollständigt die Shell die Eingabe, dann bestätigen Sie den Befehl mit ↵.

Navigation im Dateisystem

Wenn Sie die Shell in einem Fenster öffnen oder sich in einem virtuellen Terminal (siehe Abschnitt 1.3) anmelden, landen Sie in Ihrem Home-Verzeichnis – dies zeigt Ihnen die Tilde »~« am Eingabe-Prompt:

```
benutzername@computername:~$
```

Um im Dateisystem zu navigieren, nutzen Sie den Befehl cd und geben danach das gewünschte Unterverzeichnis an. Um beispielsweise in das Unterverzeichnis »Dokumente« zu wechseln:

```
cd Dokumente/
```

Der Slash (»/«) zeigt der Shell, dass es sich dabei um ein Verzeichnis handelt. Sie müssen ihn nicht unbedingt angeben, solange Sie keine weiteren Unterverzeichnisse angeben. Sie können natürlich auch einige Unterverzeichnisse überspringen. Ein Beispiel:

```
cd Dokumente/Firma/PDF-Dateien/
```

Mehrere Unterverzeichnisse müssen Sie natürlich durch einen Slash trennen.

Die Shell zeigt Ihnen an, in welchem Verzeichnis Sie sich gerade befinden – wie schon beschrieben, bedeutet die Tilde, dass Sie sich in Ihrem Home-Verzeichnis

befinden. Haben Sie das Verzeichnis gewechselt, wird Ihnen dies nach der Tilde angezeigt:

```
benutzername@computername:~Dokumente/Bilder/
```

Möchten Sie in das Wurzelverzeichnis wechseln, geben Sie einfach nur den Slash an:

```
cd /
```

Befinden Sie sich in einem Unterverzeichnis und möchten ein Verzeichnis in der Hierarchie hochgehen, geben Sie nach dem Slash zwei Punkte an:

```
cd ..
```

Möchten Sie direkt in Ihr Home-Verzeichnis wechseln, egal wo Sie sich in der Verzeichnis-Hierarchie gerade befinden, nutzen Sie den Befehl alleine ohne weitere Optionen:

```
cd
```

Als Benutzer root landen Sie so natürlich im Verzeichnis »/root«, also im Verzeichnis des Administrators.

Verzeichnis-Inhalt anzeigen

Um den Inhalt des Verzeichnisses anzuzeigen, in dem Sie sich auf der Shell befinden, nutzen Sie folgenden Befehl:

```
ls
```

Als Ausgabe erhalten Sie Dateien und Verzeichnisse angezeigt, Verzeichnisse natürlich mit einem abschließenden Slash. Ein Beispiel zeigt sich in Abbildung 1.2.

Abb. 1.2: Der Inhalt eines Verzeichnisses auf der Shell

Sie können sich natürlich auch den Inhalt eines Verzeichnisses anzeigen lassen, in dem Sie sich nicht befinden – hierzu geben Sie den Pfad zu diesem Verzeichnis an. Ein Beispiel:

```
ls Dokumente/Firma/PDF/
```

Dasselbe gelingt natürlich auch mit Verzeichnissen im Wurzelverzeichnis – Sie geben hierzu den Slash für das Wurzelverzeichnis zu Beginn an und anschließend das gewünschte Unterverzeichnis:

```
ls /etc/
```

Optionen für Befehle

Nachdem Sie die wichtigsten Befehle zur Navigation im Dateisystem kennen, haben Sie auch schon den Begriff »Optionen« gelesen. Optionen dienen dazu, Befehle zu verfeinern, also genauer angepasste Ausgaben zu ermöglichen.

Zuvor haben Sie beispielsweise den Befehl `ls` kennengelernt, der dazu dient, sich den Inhalt eines Verzeichnisses anzeigen zu lassen. Im Home-Verzeichnis eines Benutzers gibt es versteckte Dateien und Verzeichnisse. Diese beginnen mit einem Punkt, werden mit dem Befehl `ls` jedoch nicht angezeigt, um eine bessere Übersicht zu haben. Um sich solche versteckten Dateien anzeigen zu lassen, nutzen Sie zusätzlich die Option `-a`:

```
ls -a
```

Weitere Informationen zu Dateien und Verzeichnissen sehen Sie, indem Sie die Option `-l` (kleines »L«) nutzen:

```
ls -l
```

```
robertg@debian-hp:~$ ls -l
insgesamt 1848
drwxr-xr-x  2 robertg robertg    4096 15. Dez 2024  1
drwxrwxr-x  2 robertg robertg    4096 28. Nov 19:19 Bentopdf
drwxr-xr-x 11 robertg robertg    4096 27. Aug 02:52 Bilder
-rw-rw-r--  1 robertg robertg     256  9. Nov 08:27 derstandard.at_ips.txt
drwxr-xr-x  3 robertg robertg    4096 15. Jun 09:54 Dify
drwxr-xr-x  5 robertg robertg    4096  1. Nov 19:13 Docker
```

Abb. 1.3: Nähere Informationen zu Dateien und Verzeichnissen auf der Shell

Welche Bedeutungen diese Informationen haben, lesen Sie in Abschnitt 1.5.

Noch weiter verfeinern lassen sich Befehle, indem Sie Optionen miteinander kombinieren. Nehmen wir an, Sie möchten sich auch die versteckten Inhalte ansehen – also mit der Option `-a` – und die näheren Informationen – Option `-l`. Geben Sie hierfür die beiden Optionen hintereinander an. Dies funktioniert in folgenden beiden Varianten, wobei die erste natürlich bequemer ist:

```
ls -la
ls -l -a
```

Die meisten Befehle auf der Shell verfügen über solche Optionen, manche davon sogar über sehr viele. Der Befehl `ls` kennt über 50 solcher Optionen. Zu beinahe jedem Befehl auf der Shell gibt es eine sogenannte »Manpage« – einfach gesagt, eine Hilfeseite. Um eine solche Manpage zu öffnen, nutzen Sie den Befehl `man`, gefolgt vom Befehl, zu dem Sie Hilfe benötigen – ein Beispiel:

```
man ls
```

Nähere Informationen zur Manpage finden Sie später in diesem Abschnitt.

Dateien löschen und erstellen

Zum Löschen von Dateien nutzen Sie den Befehl `rm`, gefolgt von der zu löschenden Datei – um etwa die Datei »dateiname.txt« zu löschen:

```
rm dateiname.txt
```

Wichtig hierbei – die Datei wird tatsächlich ohne Umweg über den Papierkorb gelöscht, eine Wiederherstellung ist also sehr umständlich!

Sie können auch mehrere Dateien auf einen Schlag löschen, geben Sie diese durch ein Leerzeichen getrennt voneinander an:

```
rm dateiname.txt dateiname.jpg dateiname.pdf
```

Mit dem Sternchen »*« als Platzhalter löschen Sie alle Dateien, die sich im Verzeichnis befinden:

```
rm *
```

Indem Sie das Sternchen als Platzhalter nutzen und die Dateiendung folgen lassen, können Sie etwa alle ».pdf«–Dateien löschen und alle anderen Dateiformate unbehelligt lassen. Auch hier gilt natürlich die Groß- und Kleinschreibung:

```
rm *.pdf
rm *.PDF
```

Um Dateien zu erstellen, nutzen Sie hingegen den Befehl `touch`, gefolgt von der zu erstellenden Datei. Um beispielsweise die Datei »dateiname.txt« zu erstellen:

```
touch dateiname.txt
```

Verzeichnisse löschen und erstellen

Um leere Verzeichnisse zu löschen, nutzen Sie den Befehl `rmdir`, gefolgt vom Verzeichnis und dem abschließenden Slash – möchten Sie zum Beispiel das Verzeichnis »Dokumente« löschen:

```
rmdir Dokumente/
```

Wie schon angedeutet, funktioniert dies nur bei leeren Verzeichnissen. Möchten Sie ein Verzeichnis löschen, in dem sich noch Daten vorhanden sind, nutzen Sie `rm` mit den Optionen `-rf`:

```
rm -rf Dokumente/
```

Möchten Sie Verzeichnisse erstellen, dient dazu der Befehl `mkdir`, weiterhin geben Sie den Namen des zu erstellenden Verzeichnisses an. Um das Verzeichnis »Dokumente« zu erstellen:

```
mkdir Dokumente
```

Beim Erstellen eines Verzeichnisses geben Sie keinen abschließenden Slash an.

Wollen Sie ein Unterverzeichnis erstellen in einem Verzeichnis, ohne in dieses zu wechseln, nutzen Sie zusätzlich die Option `-p` – um etwa das Verzeichnis »PDF« unter »Dokumente/Firma« zu erstellen:

```
mkdir -p Dokumente/Firma/PDF
```

Dateien und Verzeichnisse kopieren

Dateien und Verzeichnisse lassen sich auf der Shell mit dem Befehl `cp` kopieren. Sie geben die zu kopierende Datei an und den Pfad, also das Verzeichnis, in das die Datei kopiert werden soll:

```
cp dateiname.pdf Dokumente/Firma/PDF/
```

Dies gelingt auch für alle Dateien im Verzeichnis mit dem Platzhalter »*«:

```
cp * Dokumente/Firma/PDF/
```

Möchten Sie ein ganzes Verzeichnis kopieren, nutzen Sie zusätzlich die Option `-r`:

```
cp -r Neue-PDF-Dateien/ Dokumente/Firma/PDF/
```

Mit `cp` können Sie auch ganz einfach Backup-Dateien im selben Verzeichnis erstellen. Backup-Dateien haben unter Linux meist die Dateiendung ».bak«. Im folgenden Beispiel wird von der Datei »dateiname.txt« ein Backup erstellt:

```
cp dateiname.txt dateiname.txt.bak
```

Auf dieselbe Weise können Sie das Backup wieder zurückspielen:

```
cp dateiname.txt.bak dateiname.txt
```

Dateien und Verzeichnisse verschieben

Zum Verschieben von Dateien und Verzeichnissen nutzen Sie den Befehl `mv`, auch hier geben Sie nachstehend das Verzeichnis an, in das verschoben werden soll:

```
mv PDF/ Dokumente/Firma/PDF/
```

Beim Verschieben von Verzeichnissen müssen Sie im Gegensatz zum Befehl `cp` keine weitere Option angeben.

Dasselbe mit einer Datei:

```
mv dateiname.pdf Dokumente/Firma/PDF/
```

`mv` kann auch zum Umbenennen von Dateien und Verzeichnissen genutzt werden, hierzu geben Sie zu Beginn die aktuelle Datei oder das aktuelle Verzeichnis an. Es folgt der neue Dateiname oder der neue Name des Verzeichnisses:

```
mv dateiname.pdf neuer_dateiname.pdf
mv Verzeichnis/ neues_Verzeichnis
```