

Linux Handbuch für Einsteiger

Der leichte Weg zum Linux-Experten

Inhaltsverzeichnis

1. Einführung	7
1.1 Was ist das Betriebssystem?	8
1.2 Die Entstehungsgeschichte von Linux.....	12
1.3 Linux und UNIX	16
1.4 Vielfältige Einsatzbereiche für Linux-Betriebssysteme	18
1.5 Der erste Kontakt mit Linux: Ein Live-System erzeugen	20
1.6 Die Linux-Community: So entsteht das Betriebssystem	28
1.7 Übungsaufgaben	30
2. Der Linux-Kernel	33
2.1 Was ist ein Kernel?	33
2.2 Die technischen Eigenschaften des Linux-Kernels	37
2.3 Verschiedene Kernel-Versionen	38
2.4 Übungsaufgaben	40
3. Linux-Distributionen	43
3.1 Was ist eine Linux-Distribution?	43
3.2 Verschiedene Arten von Distributionen	49
3.3 Die wichtigsten Linux-Distributionen im Überblick	51
3.4 Eine passende Linux-Distribution auswählen	56
3.5 Übungsaufgaben	60
4. Vorüberlegungen und Vorbereitungsmaßnahmen	63
4.1 Ausschließlich Linux verwenden oder ein Multi-Boot-System implementieren?	63
4.2 Die Partitionen der Festplatte vorbereiten	67
4.3 Weitere Vorbereitungsmaßnahmen	76
4.4 Übungsaufgaben	77
5. Installationsbeispiele für verschiedene Linux-Distributionen	80
5.1 Linux Mint	81
5.2 OpenSUSE	91
5.3 Debian	98
5.4 Manjaro Linux.....	102
5.5 Ubuntu	109
5.6 Übungsaufgaben	116

6. Die grundlegenden Funktionen von Linux	119
6.1 Das Dateisystem in Linux	119
6.2 Programme und Prozesse	125
6.3 Textverarbeitung und weitere Office-Programme	131
6.4 Drucker anschließen und Treiber installieren	138
6.5 Texteditor zum Programmieren	142
6.6 Multimedia-Anwendungen	147
6.7 Übungsaufgaben	150
7. Die Shell: Wichtiges Werkzeug für die Arbeit mit Linux	153
7.1 Was ist eine Shell?	153
7.2 Verschiedene Shells in Linux	157
7.3 Dateien und Ordner verwalten	163
7.4 Anwender und Zugriffsrechte verwalten	173
7.5 Programme installieren	189
7.6 Wildcards	194
7.7 Pipes und Filter	202
7.8 Die Befehle cat, head und tail: Inhalte von Dateien anzeigen	209
7.9 Nano Editor, VIM und weitere VI-Editoren über die Shell aufrufen	218
7.10 Informationen mit help, man und info abrufen	224
7.11 Übungsaufgaben	233
8. Shell-Programme erstellen	236
8.1 Was ist ein Shell-Script?	236
8.2 Die ersten Shell Scripts erstellen	239
8.3 Variablen verwenden	248
8.4 Arrays	255
8.5 Einnahmen durch den Anwender aufnehmen	259
8.6 Verzweigungen	261
8.7 Schleifen in der Shell ausführen	272
8.8 Funktionen in der Shell verwenden	283
8.9 Übungsaufgaben	291
9. Automatischer Aufruf von Programmen mit cron und at	296
9.1 Cron und crontab: Was versteckt sich hinter diesen Begriffen?	297
9.2 Aufgaben mit cron regelmäßig ausführen	298
9.3 Das Programm at als Alternative für eine einmalige Ausführung	303
9.4 Übungsaufgaben	306
10. Netzwerke in Linux erstellen	309
10.1 Die Vorbereitungsmaßnahmen für die Einrichtung des Netzwerks	310
10.2 Dateien mit anderen Computern teilen	318

10.3	Einen Print-Server zum Drucken einrichten.....	324
10.4	Übungsaufgaben	330
11.	Ubuntu aktualisieren	333
11.1	So nehmen Sie die Aktualisierung vor.....	334
11.2	Übungsaufgaben	336
12.	Glossar	340
13.	Index	347

Kapitel 1

Einführung

Jeder Computer benötigt ein *Betriebssystem*. Dieses ist unter anderem dazu notwendig, die *Hardware* zu verwalten, die Ausführung von Anwenderprogrammen zu organisieren und dem Anwender eine geeignete Oberfläche für die Steuerung des Geräts zur Verfügung zu stellen. Ein Betriebssystem ist nicht nur für gewöhnliche PCs erforderlich. Auch Smartphones, Tablets und sogar Smart-TVs, NAS-Speicher und viele weitere programmierbare Geräte benötigen diese Art von Software. Sie hat eine enorme Bedeutung für die Funktionsweise, die Performance und die Handhabung der entsprechenden Produkte.

Es gibt viele unterschiedliche Betriebssysteme. Im Bereich der Desktop-Computer und Laptops ist beispielsweise Windows führend. Die große Mehrheit dieser Geräte verwendet das bekannte Betriebssystem von Microsoft. Im Bereich der mobilen Endgeräte nimmt hingegen Android von Google eine marktbeherrschende Stellung ein. In beiden Bereichen gibt es auch Konkurrenz von Apple: macOS für gewöhnliche Computer und iOS für Smartphones und Tablets. Diese Betriebssysteme zeichnen sich dadurch aus, dass sie sich nur für die Geräte dieses Herstellers eignen. Das führt dazu, dass die Verbreitung deutlich geringer ist.

Dieses Buch beschäftigt sich jedoch mit dem Betriebssystem Linux. Dieses hat einen deutlich geringeren Bekanntheitsgrad. Viele Anwender haben diesen Ausdruck zwar bereits gehört, doch haben sie noch nie bewusst Linux verwendet. Dennoch spielt das Betriebssystem Linux eine enorme Rolle. Im PC-Segment ist es zwar nicht allzu weit verbreitet. Doch ist es für den Betrieb von Servern von sehr großer Bedeutung. Insbesondere Webserver verwenden häufig Linux-Betriebssysteme. Das hat eine herausragende Rolle für die Gestaltung des Internets zur Folge. Hinzu kommt, dass Linux die Grundlage vieler weiterer Betriebssysteme darstellt. Beim Smartphone-Betriebssystem Android handelt es sich beispielsweise um eine Weiterentwicklung von Linux. Daher hat

ein Großteil der Bevölkerung das Betriebssystem Linux doch schon einmal verwendet – auch wenn sich die Anwender darüber meistens überhaupt nicht darüber bewusst sind.

Dieses Buch stellt das Betriebssystem Linux vor. Dabei erfahren Sie, durch welche Eigenschaften es sich auszeichnet, was bei der Verwendung zu beachten ist und wie Sie es installieren. Darüber hinaus erhalten Sie vielfältige Informationen zur Anwendung und zur Konfiguration des Systems.

1.1 Was ist das Betriebssystem?

Da es sich bei Linux um ein Betriebssystem handelt, besteht der erste Schritt für die Annäherung an dieses Thema darin, zu beschreiben, was ein Betriebssystem überhaupt ist und welche Aufgaben es hat. Auf diese Weise fällt es dann leichter, die Eigenheiten von Linux zu verstehen und die Anwendung zu erlernen.

Das Betriebssystem – das entsprechend der englischsprachigen Bezeichnung Operating System häufig auch als OS abgekürzt wird – ist eine Software, die für den Betrieb eines jeden Computers notwendig ist. Sie ist für die Steuerung sowohl der Systemressourcen als auch der Anwenderprogramme zuständig. Diese beiden Aufgaben sind jedoch eng miteinander verknüpft und lassen sich nicht vollkommen unabhängig voneinander betrachten. Darüber hinaus stellt sie eine Oberfläche für den Anwender zur Verfügung, mit deren Hilfe er das Gerät steuern und die gewünschten Funktionen abrufen kann. Die folgenden Abschnitte stellen die wesentlichen Aufgaben des Betriebssystems etwas detaillierter vor.

Prozessor-Management

Der *Prozessor* stellt das zentrale Element des Computers dar. Es ist möglich, diesem einen bestimmten Befehl zuzuführen. Dieser besteht aus einem Binärcode. Dabei kann es sich beispielsweise um eine Anweisung handeln, eine Information aus einem bestimmten Speicherbereich abzurufen oder eine Berechnung durchzuführen. Jeder Befehl,

der an den Prozessor übergeben wird, hat ein ganz bestimmtes Ergebnis zur Folge. Dabei kann es sich um eine Ausgabe über den Bildschirm oder ein bestimmtes Peripheriegerät handeln oder es ist möglich, den entsprechenden Wert im Arbeitsspeicher abzuspeichern.

Das Betriebssystem ist für die Steuerung dieser Prozesse notwendig. Wenn ein *Anwenderprogramm* ausgeführt werden soll, benötigt dieses selbstverständlich Zugriff auf den Prozessor. Hierfür bestehen mehrere Möglichkeiten. Zum einen bietet das Betriebssystem hierfür eine geeignete Schnittstelle an. Ein Programm kann einen sogenannten System Call durchführen. Dabei handelt es sich um eine Anweisung an das Betriebssystem. Dieses führt sie dann auf dem Prozessor aus und gibt das Ergebnis an das Programm zurück. Die meisten Programme nutzen hingegen einen direkten Zugriff auf den Prozessor und werden so unmittelbar ausgeführt. Dennoch ist das Betriebssystem hierbei involviert. Dieses muss der entsprechenden Anwendung die Berechtigung zum Zugriff auf den Prozessor erteilen. Die Organisationsform, die das Betriebssystem hierbei wählt, hat einen großen Einfluss auf die Performance des gesamten Systems. Daher ist diese Software von großer Bedeutung für die Leistung des Rechners.

Speicherverwaltung

Auch beim Zugriff auf den Arbeitsspeicher ist es notwendig, für Ordnung zu sorgen. Das Betriebssystem muss bestimmte Speicherbereiche für die einzelnen Anwendungen reservieren und gewährleisten, dass andere Programme keinen Zugriff darauf erhalten. Würden zwei Programme den gleichen Speicherbereich parallel nutzen, würden sich die Berechnungen gegenseitig beeinflussen, sodass eine korrekte Ausführung nicht möglich wäre.

Darüber hinaus handelt es sich beim *Arbeitsspeicher* um eine begrenzte Ressource. Es ist nicht ungewöhnlich, dass die verschiedenen Anwendungen, die zu einem bestimmten Zeitpunkt auf dem Computer ablaufen, deutlich mehr Speicherplatz einfordern als vorhanden ist. Auch in diesem Fall ist das Betriebssystem gefordert. Es muss genaue Kriterien

aufstellen, nach denen der Speicherplatz zugeteilt wird. Diese Funktion ist ebenfalls für die Performance des gesamten Systems von enormer Bedeutung.

Steuerung von Peripheriegeräten

Ein Computer verfügt in der Regel über zahlreiche *Peripheriegeräte* – beispielsweise über einen Bildschirm, eine Tastatur, eine Maus und einen Drucker. Auch hierbei ist eine Steuerung erforderlich. Für die genauen Details kommen Hardwaretreiber zum Einsatz. Doch auch hierbei ist das Betriebssystem für die Organisation erforderlich. Es muss beispielsweise bestimmen, welche Anwendung Zugriff auf welches Peripheriegerät erhält. Daraufhin muss es mit dem Treiber kommunizieren, um eine korrekte Abwicklung der entsprechenden Aufgabe sicherzustellen.

Verwaltung der Dateien

Das Betriebssystem ist auch für die Verwaltung der *Dateien* auf der Festplatte verantwortlich. Es muss hierbei passende Strukturen erstellen – beispielsweise durch Ordner. Hierbei spricht man vom Dateisystem – oder auf Englisch vom File System. Auch in diesem Bereich ist es wichtig, dass das Betriebssystem kontrolliert, welche Anwendung Zugriff auf welche Bereiche erhält. Es muss die entsprechenden Ressourcen zuweisen und wieder freigeben.

Sicherheit

Eine immer wichtigere Aufgabe des Betriebssystems besteht darin, die Sicherheit aufrechtzuerhalten. Dabei ist insbesondere die Kontrolle der Zugriffsrechte wichtig. Das OS muss stets überprüfen, ob eine Anwendung dazu berechtigt ist, auf eine Ressource zuzugreifen. Darüber hinaus fordert es eine Bestätigung durch den Anwender ein, wenn ein neues Programm installiert werden soll. Viele moderne Systeme verfügen außerdem über integrierte Funktionen, die Schadsoftware und andere Angriffe erkennen und abwehren können.

Bereitstellen einer Benutzeroberfläche

Bei den bisher vorgestellten Aufgaben handelte es sich ausschließlich um die Organisation der internen Prozesse. Das ist sicherlich von großer Bedeutung für die Funktionsweise des Computers. Der Anwender kommt damit in der Regel jedoch nicht in Kontakt. Der Berührungspunkt besteht in der Benutzeroberfläche. Ein Betriebssystem muss es dem Anwender erlauben, den Computer zu steuern – beispielsweise ein Programm aufzurufen oder eine Datei zu speichern. Bei frühen Betriebssystemen kam hierfür ausschließlich ein sogenannter Kommandozeileninterpreter zum Einsatz. Dieser ermöglichte es, schriftliche Befehle einzugeben. Diese Form der Steuerung ist jedoch recht umständlich und außerdem erfordert sie eine genaue Kenntnis der möglichen Befehle. Daher verwenden moderne Betriebssysteme in der Regel eine *grafische Benutzeroberfläche* (GUI = Graphical User Interface). Diese ermöglicht eine einfache und intuitive Steuerung per Maus oder bei vielen Geräten auch per Touchscreen.

Hierbei lässt sich bereits auf eine Besonderheit des Betriebssystems Linux hinweisen. Bei den meisten Konkurrenzprodukten – wie etwa Windows oder macOS – sind die Steuerung der internen Prozesse und die grafische Benutzeroberfläche fest miteinander verbunden und stellen eine Einheit dar. Bei Linux ist dies jedoch nicht der Fall. Diese Bezeichnung bezieht sich eigentlich nur auf den sogenannten Linux-Kernel – den Teil der Software, der die internen Prozesse steuert. Darüber hinaus gibt es sogenannte Linux-Desktops. Diese setzen eine grafische Benutzeroberfläche des Linux-Kernels um. Hierbei gibt es jedoch zahlreiche verschiedene Ausführungen. Daher gibt es Linux-Systeme, die auf den Anwender ganz unterschiedlich wirken.

Integrierte Software

Fast jedes moderne Betriebssystem wird mit einer Reihe verschiedener Software-Produkte ausgeliefert – beispielsweise Texteditoren, Programme für die Multimedia-Wiedergabe oder für die Darstellung von Fotos. Diese sind jedoch eigentlich kein Bestandteil des Betriebssystems. Dennoch stellen sie ein wichtiges Merkmal der verschiedenen

Angebote dar, sodass dieser Aspekt in diesem Buch ebenfalls gelegentlich zur Sprache kommen wird.

Das Betriebssystem – die zentrale Steuerungs-Software des Computers

Die Zusammenstellung der Aufgaben des Betriebssystems hat gezeigt, dass diesem eine zentrale Rolle bei der Verwendung eines Computers zukommt. Es ist für die Steuerung aller Prozesse zuständig, die darauf ablaufen. Es bestimmt, welches Programm welche Ressourcen verwenden kann, und sorgt dabei für eine gute Performance des Systems. Es stellt die Schnittstelle zwischen der Hardware auf der einen Seite und den Programmen und dem Anwender auf der anderen Seite dar. Dazu zählt auch die Bereitstellung einer Benutzeroberfläche, die die Steuerung des Systems erlaubt.

1.2 Die Entstehungsgeschichte von Linux

Bevor wir uns den Anwendungsmöglichkeiten von Linux zuwenden, ist es sinnvoll, einen kurzen Blick auf die Entstehung des Betriebssystems zu werfen. Das gibt bereits wertvolle Einblicke, wie Linux aufgebaut ist und für welche Zwecke sich das Betriebssystem verwenden lässt.

Die Entwicklung von Linux geht auf zwei Stränge zurück. Zum einen gibt es das *GNU-Project*. Dieses war bereits seit Anfang der 80-er Jahre darum bemüht, ein freies UNIX-ähnliches Betriebssystem zu gestalten. Zum anderen hat der finnische Informatiker Linus Torvalds zu Beginn der 90er Jahre ein privates Projekt gestartet und dabei die wesentlichen Bestandteile der ersten Linux-Version entwickelt. Aus der Kombination dieser beiden Stränge entstand das heute bekannte Betriebssystem Linux.

Das GNU-Project

Um die chronologische Reihenfolge der beiden Projekte beizubehalten, beginnen wir mit der Vorstellung des GNU-Projects. Dieses entstand

1983 unter der Federführung von Richard Stallman – einem US-amerikanischen Informatiker und Aktivisten für freie Software. Eines der wesentlichen Ziele dieses Projekts bestand darin, ein freies Betriebssystem zu entwickeln, das ein ähnliches Verhalten wie das Betriebssystem UNIX aufweist.

Dieses Ziel konnte das GNU-Projekt jedoch bis heute nicht erreichen. Die Entwicklung dieses neuen Betriebssystems verlief sehr langsam. Dennoch entstanden dabei zahlreiche wichtige Software-Produkte. Besonders erwähnenswert ist hierbei die GNU Compiler Collection (GCC). Diese enthält freie Compiler für zahlreiche Programmiersprachen – insbesondere für C. Sie kam bereits für die Erstellung vieler Millionen Computerprogramme zum Einsatz und hatte einen großen Einfluss auf die Entwicklung der modernen Informatik.

1

Private Terminal-Emulation von Linus Torvalds

Die zweite Grundlage für das Betriebssystem Linux entstand Anfang der 90er Jahre. Der finnische Informatik-Student Linus Torvalds entwickelte damals eine sogenannte Terminal-Emulation. Dabei handelt es sich um ein Programm, das die wesentlichen Funktionen eines UNIX-Terminals nachstellt. Dabei handelt es sich um einen Bestandteil dieses Betriebssystems, der es erlaubt, es mithilfe schriftlicher Kommandos zu steuern. Das *Terminal* spielt eine entscheidende Rolle für die Verwendung von UNIX und für die Gestaltung von Software für dieses Betriebssystem.

Der Grund dafür bestand darin, dass Torvalds dieses UNIX-Terminal für sein Studium an der Universität nutzte. Das dortige Rechenzentrum verwendete, wie zu dieser Zeit üblich, UNIX-Server. Torvalds selbst nutzte auf seinem privaten Rechner das Betriebssystem Minix – ein unixoides Betriebssystem, das ausschließlich für Bildungszwecke eingesetzt wurde.

Die Terminal-Emulation war notwendig, um sich per Internet mit dem Computer seiner Universität zu verbinden. Minix stellte diese Funktion zwar zur Verfügung, allerdings wies das Programm zahlreiche

Schwachstellen auf. Das führte zu dem Problem, dass es für die Studenten damals sehr schwierig war, die Anwendungen, die sie für die Universität gestalteten, auf ihren privaten Rechnern nachzustellen. Torvalds schrieb den Emulator, um dieses Problem zu beheben und, wie er selbst sagte, um seinen Computer besser zu verstehen.

Im Laufe der Zeit entstand daraus jedoch ein immer umfangreicheres Programm, das immer mehr Ähnlichkeiten mit einem Betriebssystem aufwies. Als sich Torvalds darüber bewusst wurde, veröffentlichte er am 25. August 1991 im Diskussionsforum Usenet ein Posting, das enorme Auswirkungen auf die moderne Informatik hatte. Darin kündigte er an, dass er ein privates Betriebssystem als Alternative zu Minix erstellt habe und bat andere Entwickler um Vorschläge für Funktionen, die dabei umgesetzt werden sollten. Auf diese Weise wurde bereits eine beträchtliche Anzahl an Entwicklern auf dieses Projekt aufmerksam. Diese machten Vorschläge für die Verbesserung der ersten Entwürfe und unterstützten Torvalds bei der Umsetzung.

Torvalds verwendete für sein Betriebssystem ursprünglich die Bezeichnung Freax. Diese Wortschöpfung stellte eine Kombination aus den Begriffen „Freak“ und „Free“ dar – mit dem abschließenden „X“, das die Verbindung zu UNIX symbolisiert. Die Bezeichnung Linux in Anspielung an seinen eigenen Vornamen ist Torvalds ebenfalls bereits in den Sinn gekommen. Er hatte sie jedoch verworfen, da er sie als zu egozentrisch empfand. Als er das Betriebssystem jedoch auf dem Server der Universität in Helsinki veröffentlichte, um es auch anderen Menschen zur Verfügung zu stellen, entschloss sich der hierfür zuständige Mitarbeiter Ari Lemmke dazu, die Bezeichnung Linux zu verwenden. Diese Änderung führte er ohne Absprache mit Torvalds durch. Dieser akzeptierte die Entscheidung jedoch nachträglich und musste zugeben, dass dieser Name deutlich eingängiger ist.

Veröffentlichung unter der GNU-Lizenz

Zunächst handelte es sich bei Linux und GNU um eigenständige Entwicklungen. Allerdings verwendete Torvalds bei der Gestaltung des Betriebssystems bereits GNU-Software. Torvalds gestaltete Linux zwar als

freie Software, doch verbreitete er das Betriebssystem unter einer eigenen Lizenz, die die kommerzielle Verwendung ausschloss.

Von großer Bedeutung für die Entstehung des neuen Betriebssystems war die Veröffentlichung von Linux unter der GNU General Public License (GPL). Das hatte wichtige Auswirkungen auf die verwendete Technik. Bei Linux handelt es sich lediglich um den Systemkern. Die Annäherung an das GNU-Projekt erlaubte es jedoch, die GNU-Software darauf aufzusetzen und auf diese Weise ein komplettes Betriebssystem zu erstellen.

Darüber hinaus erlangte das Betriebssystem auf diese Weise deutlich größere Bekanntheit. Durch die Veröffentlichung unter GNU GPL wurden Programmierer in aller Welt auf das neue Projekt aufmerksam. Das wirkte sich nicht nur positiv auf die Verbreitung aus. Darüber hinaus entstand auf diese Weise eine große Community, die an der Entwicklung mitwirkte. Torvalds sagte rückblickend zu diesem Schritt: „Making Linux GPL'd was definitely the best thing I ever did.“ (Interview mit Hiroo Yamagata, August 1997).

Trotz dieser Entwicklung nutzten während der Anfangszeit fast ausschließlich Informatik-Studenten und professionelle Programmierer Linux. Der Grund dafür bestand zum einen darin, dass der Installations-Prozess damals ausgesprochen kompliziert war. Es war in der Regel notwendig, den Code in einem Rechenzentrum in einer Universität oder in einem Unternehmen über einen FTP-Server herunterzuladen, um diesen anschließend auf bis zu 30 Disketten zu überspielen. Zum anderen brachte Linux diesem Anwenderkreis den größten Nutzen. Da an Universitäten und in größeren Unternehmen fast ausschließlich UNIX-Server zum Einsatz kamen, arbeiteten die Programmierer normalerweise mit diesem System. Das führte jedoch dazu, dass sie ihre Programme auf ihrem privaten PC nicht ausprobieren konnten, da hierbei ein anderes Betriebssystem zum Einsatz kam. Erst mit der Entwicklung von Linux, das viele Gemeinsamkeiten mit UNIX aufweist, wurde dies möglich.

Bis Linux massentauglich wurde, sollten noch einige Jahre vergehen. Doch entstanden immer neue Linux-Distributionen, die die Installation und die Anwendung deutlich erleichterten. Mittlerweile stellt die

Verwendung dieses Betriebssystems auch für Nicht-Informatiker kein großes Problem mehr dar.

1.3 Linux und UNIX

In den bisherigen Abschnitten wurde mehrfach das Betriebssystem *UNIX* erwähnt. Das zeigt bereits, dass dieses einen großen Einfluss auf die Entwicklung von Linux hatte. Häufig werden die Begriffe Linux und UNIX sogar synonym zueinander verwendet – obwohl dies eigentlich nicht korrekt ist. Aufgrund der Verbindungen zwischen den beiden Betriebssystemen ist es aber dennoch sinnvoll, einen kurzen Blick auf UNIX zu werfen und die Gemeinsamkeiten und Verschiedenheiten zu Linux herauszuarbeiten.

UNIX – ein wichtiges Betriebssystem der 70-er und 80-er Jahre

Die Geschichte von UNIX beginnt in den frühen 60-er Jahren. Damals schlossen sich die Unternehmen General Electrics und AT & T Bell Labs mit dem Massachusetts Institute of Technology (MIT) zusammen, um ein gemeinsames Betriebssystem zu entwickeln. Dieses sollte den Namen Multics erhalten. Eines der wesentlichen Ziele bestand dabei darin, dass dieses mehrbenutzerfähig sein sollte. Allerdings war der gewählte Ansatz ausgesprochen komplex, sodass es bei diesem Projekt nur zu geringen Fortschritten kam.

Aus diesem Grund beschloss AT & T Bell Labs, aus diesem Projekt auszusteigen und ein eigenes Betriebssystem zu entwickeln. Zu der Gruppe, die das gemeinschaftliche Projekt verließ, gehörten unter anderem Ken Thompson und Dennis Ritchie – zwei der einflussreichsten Informatiker des 20. Jahrhunderts. Unter ihrer Federführung entstand ein neues Projekt, das im Prinzip die gleichen Ziele verfolgte wie Multics, aber wesentlich einfacher aufgebaut war. Das Ergebnis daraus war das Betriebssystem UNIX, das im August 1969 erschien.

Die erste Version war in Assemblersprache geschrieben. Dabei handelt es sich um eine Programmiersprache, die ein sehr niedriges Abstrak-

tionsniveau aufweist. Das bedeutet, dass sich die Befehle genau an den Kommandos im Binärcode orientieren, die dem Prozessor zugeführt werden. Das hat wiederum zur Folge, dass der Programmcode genau auf die verwendete Hardware abgestimmt sein muss. Aus diesem Grund war das erste UNIX-Betriebssystem lediglich auf dem Rechner PDP-7 lauffähig, für den es entwickelt wurde.

Dennis Ritchie entwickelte später die Programmiersprache C, die bis heute zu den beliebtesten Programmiersprachen zählt. Thompson und Ritchie verwendeten diese daraufhin auch für die Entwicklung ihres Betriebssystems. Das hatte zur Folge, dass sich dieses deutlich leichter auf andere Rechnerarchitekturen übertragen ließ.

UNIX kam in den 70-er Jahren hauptsächlich in Universitäten zum Einsatz. Zu diesem Zeitpunkt war das Betriebssystem frei verfügbar und der Quellcode offen zugänglich. Ab den 80-er Jahren begann AT & T mit der kommerziellen Nutzung der Software. Fortan fielen erhebliche Lizenzgebühren für die Nutzung an. Außerdem war es nicht mehr erlaubt, den Quellcode zu veröffentlichen. Das stellte insbesondere für den Lehrbetrieb an den Universitäten eine erhebliche Einschränkung dar. Hier kam der UNIX-Quellcode häufig zum Einsatz.

Linux als UNIX-ähnliches System

UNIX war bereits seit den 70-er Jahren das gebräuchlichste Betriebssystem für die Server an Universitäten in aller Welt. Durch die Änderung der Lizenzierung zu Beginn der 80-er Jahre war die Verwendung des Quellcodes in den Vorlesungen jedoch nicht mehr möglich. Darüber hinaus machten es die hohen Lizenzgebühren für die meisten Studenten und Studentinnen unmöglich, ein UNIX-System auf ihren privaten Computern zu installieren. Aus diesem Grund war es schwierig, die Programme, die für die Universität entstanden, in einem privaten Umfeld nachzustellen.

Ein erster Schritt, um eine Alternative zum proprietären UNIX zu erstellen, bestand darin, einen allgemeinen Standard für Betriebssysteme zu

entwickeln. Treibende Kraft hinter diesem Bestreben war Richard Stallman, der auch für das GNU-Projekt verantwortlich war. Das führte zur Verabschiedung des Standards POSIX im Jahre 1988. Dieser beschreibt einheitliche Anforderungen an die Schnittstelle, die für die Kommunikation zwischen dem Betriebssystem und der Anwendersoftware zuständig ist. POSIX orientiert sich an den Funktionen von UNIX. Das bedeutet, dass sich Programme, die für UNIX geschrieben sind, auch für andere Systeme kompilieren lassen, die POSIX umsetzen.

Das dominierende Betriebssystem für private Computer war zu dieser Zeit jedoch MS-DOS. Dieses beachtet jedoch den POSIX-Standard nicht. Das führte zu der beschriebenen Inkompatibilität zwischen den privaten Rechnern und den Servern an der Universität.

Linux schloss diese Lücke. Es erfüllte zahlreiche Anforderungen, die viele Studenten der damaligen Zeit an ihr Betriebssystem stellten: Es war kostenfrei nutzbar, quellenoffen, für die Verwendung auf dem PC geeignet und POSIX-konform. Daraus geht auch die Verbindung zwischen Linux und UNIX hervor. Bei Linux handelt es sich zwar um ein selbstständiges Betriebssystem, das von Grund auf neu entwickelt wurde und das praktisch keinen UNIX-Code verwendet. Da beide Systeme jedoch dem POSIX-Standard entsprechen, ist eine Kompatibilität zwischen den Systemen gegeben.

Die Ähnlichkeiten zwischen UNIX und Linux beschränken sich jedoch nicht auf die Beachtung des POSIX-Standards. Darüber hinaus verwendet Linux zahlreiche Tools, deren Verwendung identisch zu UNIX-Systemen ist. In erster Linie ist hierbei die Shell zu nennen, die im weiteren Verlauf dieses Buchs noch ausführlicher besprochen wird. Hierbei handelt es sich um das zentrale Steuerungswerkzeug der beiden Betriebssysteme. Die Verwendung ist unter UNIX und Linux beinahe identisch.

1.4 Vielfältige Einsatzbereiche für Linux-Betriebssysteme

Linux kommt in vielen verschiedenen Bereichen zum Einsatz. Ursprünglich wurde das Betriebssystem für die Verwendung auf einem

Desktop-Computer entwickelt. Es ist hierbei möglich, Linux als alleiniges Betriebssystem zu installieren. Sehr beliebt ist jedoch auch die Einrichtung eines *Multi-Boot-Systems*. Dieses ermöglicht es, Linux neben einem anderen Betriebssystem zu verwenden – beispielsweise neben Windows oder macOS. Obwohl die Verwendung auf dem Desktop-Computer dem ursprünglichen Einsatzzweck von Linux entspricht, ist der Marktanteil hierbei nur recht gering. Laut Erhebungen von netmarketshare.com liegt der Anteil von Linux für Desktop-PCs und Laptops aktuell (August 2020) bei 2,69 Prozent. Allerdings ist hierbei eine erhebliche Steigerung zu beobachten. Vor etwa einem Jahrzehnt lag der Anteil noch unter einem Prozent.

1

Besonders beliebt ist die Verwendung von Linux für Server. Dieses Betriebssystem ist sehr stabil, einfach zu warten und es eignet sich für beinahe alle Server-Architekturen. Daher erfüllt es die wesentlichen Anforderungen an ein Server-Betriebssystem. Außerdem ist es modular aufgebaut, sodass sich damit auch sehr gut dedizierte Server einrichten lassen. Darüber hinaus spielt es auch hierbei sicherlich eine Rolle, dass für die Verwendung von Linux keine Lizenzkosten anfallen. Verlässliche Nutzungsstatistiken sind in diesem Bereich nur schwer erhältlich. Viele Beobachter gehen jedoch davon aus, dass mindestens 60 Prozent aller Webseiten auf Linux-Servern gehostet werden.

Auch in anderen Bereichen ist Linux von enormer Bedeutung. Dabei sind insbesondere Smartphones und Tablets zu nennen. Das Betriebssystem Android, das einen Marktanteil von rund 70 Prozent in diesem Bereich aufweist, verwendet beispielsweise einen Linux-Kernel. Allerdings ist diese Verwandtschaft für den Anwender kaum noch zu bemerken. Android verzichtet auf zahlreiche GNU-Tools, die den typischen Charakter gewöhnlicher Linux-Distributionen prägen und die eine starke Verbindung zu UNIX aufweisen. Aufgrund dieser deutlichen Unterschiede wird das Android-Betriebssystem in diesem Buch auch nicht weiter behandelt.

Besonders beeindruckend ist die Position von Linux im Bereich der Supercomputer. Nach Angaben der Organisation TOP500, die Statistiken zu den 500 leistungstärksten Computern der Welt erhebt, nutzen

derzeit (Stand 2020) ausnahmslos alle hier gelisteten Rechner Linux als Betriebssystem. Allerdings handelt es sich hierbei um Versionen mit einigen Abwandlungen, die auf die spezifischen Architekturen dieser Supercomputer ausgerichtet sind.

1.5 Der erste Kontakt mit Linux: Ein Live-System erzeugen

Eine wichtige Rolle nimmt Linux auch im Bereich der Live-Systeme ein. Bei einem Live-System handelt es sich um ein Betriebssystem, das keine feste Installation erforderlich macht. Stattdessen wird es auf einem externen bootfähigen Medium abgespeichert. Heutzutage kommt hierfür meistens ein USB-Stick zum Einsatz. Früher kamen hierfür jedoch häufig DVDs, CDs und Disketten zum Einsatz. Während viele andere Betriebssysteme keine Live-Version anbieten, hält Linux hierfür unzählige Möglichkeiten bereit.

Allerdings stellt sich noch die Frage, welche Vorteile es bietet, ein Live-System zu verwenden. Dieses bietet viele Anwendungsmöglichkeiten. In der Praxis kommt es häufig zum Einsatz, wenn Probleme mit dem eigentlich installierten Betriebssystem auftreten. Wenn sich dieses nicht mehr starten lässt, stellt das Live-System eine gute Möglichkeit dar, um den Computer dennoch in Betrieb zu nehmen, ohne gleich ein neues System zu installieren, das alle bisherigen Daten überschreiben würde. Das ist insbesondere für die Datenrettung sehr praktisch. Auch wenn es zu einem Befall mit Malware gekommen ist, kann dies hilfreich sein. Das Live-System bietet Zugriff auf alle Dateien der Festplatte – selbst auf die Bereiche, die in der eigentlichen Betriebssystem-Umgebung unter einem besonderen Schutz stehen. Da sich die Schadsoftware häufig in diesen Bereichen einnistet, ist es deutlich einfacher, sie zu entfernen. Auch hinsichtlich der Sicherheit kann die Verwendung eines Live-Systems sinnvoll sein. Sollten Sie etwa sensible Informationen an einem Rechner bearbeiten, auf den auch andere Personen Zugriff haben, ist es möglich, dass einer dieser Nutzer eine Schadsoftware installiert hat, um die Daten auszuspähen. Durch die Verwendung eines Live-Systems stellen Sie jedoch sicher, dass keine anderen Prozesse auf dem Computer ablaufen.

Um die ersten Kapitel, die in erster Linie Grundlagenwissen vermitteln und kaum praktische Bestandteile beinhalten, etwas aufzulockern, wollen wir nun die Verwendung eines solchen Live-Systems ausprobieren. Hierfür müssen wir eine passende Linux-Distribution auswählen. Was das genau ist, wird in Kapitel 3 noch ausführlich beschrieben. An dieser Stelle sei lediglich gesagt, dass es sich hierbei um unterschiedliche Ausführungen des Linux-Betriebssystems handelt, die sich durch verschiedene Anwendungsmöglichkeiten und Gestaltungsweisen auszeichnen.

Zahlreiche Linux-Distributionen bieten die Möglichkeit, ein Live-System zu erstellen. Wir entscheiden uns bei unserem Beispiel für Knoppix. Hierbei handelt es sich um eine der ersten Linux-Distributionen, die ein Live-System anboten. Diese Software gelangte bei diesem speziellen Anwendungszweck zu einer sehr hohen Beliebtheit und trug erheblich zur Verbreitung von Linux-Live-Systemen bei. Obwohl mittlerweile auch eine feste Installation möglich ist, kommt Knoppix bis heute fast ausschließlich als Live-System zum Einsatz.

Der erste Schritt hierfür besteht darin, das Betriebssystem Knoppix herunterzuladen. Dafür bietet es sich an, die Seite des Entwicklers aufzusuchen und dort die Download-Möglichkeiten abzurufen: <https://www.knopper.net/knoppix-mirrors/index-en.html>. Das zeigt, dass hierbei eine große Auswahl besteht. Nun ist es möglich, eine dieser Download-Möglichkeiten aufzurufen. Wir entscheiden uns für den folgenden Link: <http://ftp.uni-kl.de/pub/linux/knoppix-dvd/>. Selbstverständlich wäre es jedoch auch möglich, eine andere Alternative auszuwählen. Es ist empfehlenswert, hierbei stets einen Link auszuwählen, der mit DVD gekennzeichnet ist. Die Version für CDs wird bereits seit längerer Zeit nicht mehr aktualisiert. Sollten Sie keinen FTP-Client installiert haben, ist es außerdem ratsam, einen der HTTP-Download-Links auszuwählen. Die folgende Abbildung zeigt, dass hier nun viele verschiedene Dateien zum Download bereitstehen.

dem USB-Stick gespeichert waren, verlorengehen. Falls Sie hier wichtige Dateien aufbewahren, sollten Sie diese zuvor an einem anderen Ort abspeichern. Nachdem Sie die Auswahl bestätigt haben, kommt es zu einem Installationsprozess, der einige Minuten dauern kann.

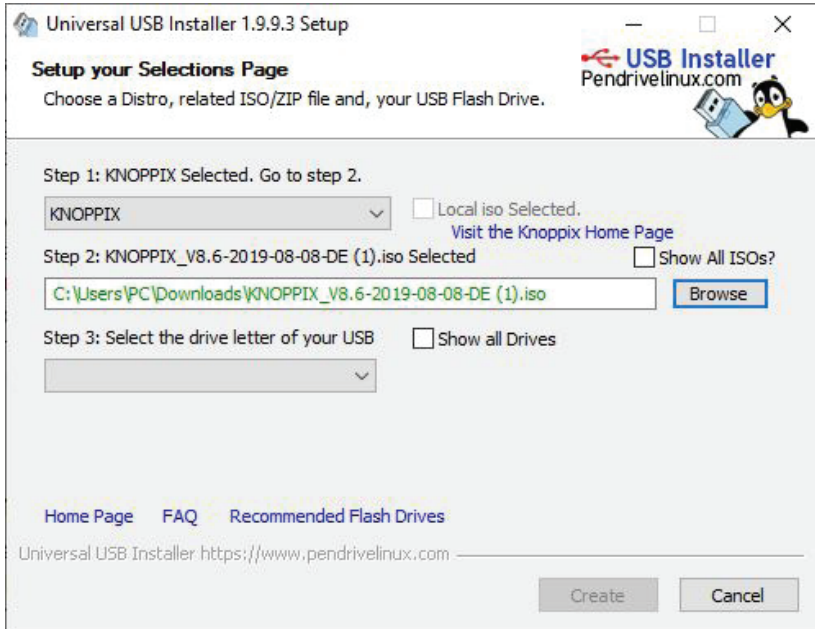


Abb. 1.2 Einen bootfähigen USB-Stick erstellen

Nun können Sie Ihr erstes Linux-Betriebssystem ausprobieren. Dazu müssen Sie den soeben erstellen USB-Stick in den Computer einführen und diesen daraufhin erneut starten. In der Regel erscheint nun nicht das gewohnte Betriebssystem, sondern KNOPPIX. Sollte dies nicht der Fall sein, liegt dies in der Regel daran, dass der USB-Port an Ihrem Computer nicht zum Booten vorgesehen ist. Um dies zu ändern, müssen Sie beim Anschalten zum Boot-Menü wechseln. Die folgenden Abschnitte zeigen, wie Sie dabei vorgehen. Sollte Ihr Computer das Live-System bereits ordnungsgemäß gestartet haben, können Sie diesen Bereich überspringen.

Bei Computern, die mit *BIOS* arbeiten, ist es hierfür notwendig, direkt nach dem Start eine bestimmte Taste oder Tastenkombination zu drücken. Um welche Tasten es sich dabei handelt, ist jedoch von Gerät zu Gerät unterschiedlich. Häufig handelt es sich hierbei um die Entf-Taste (auf Englisch Del) oder um die F2-Taste. Es bestehen jedoch noch weitere Möglichkeiten. Normalerweise erscheint jedoch direkt nach dem Starten eine kurze Nachricht, die angibt, welche Taste hierfür notwendig ist wie etwa: „Press to enter setup“.

Die meisten neueren Rechner arbeiten jedoch mittlerweile mit einer *UEFI*-Firmware. Hierbei ist es notwendig, zunächst Windows wie gewohnt zu starten. Daraufhin bestehen zwei verschiedene Möglichkeiten. Zum einen können Sie bei gedrückter Umschalt-Taste (Shift) auf die Schaltfläche „Neu Starten“ klicken. Die andere Alternative besteht darin, die Windows-Einstellungen über das Zahnrad-Symbol im Start-Menü aufzurufen. Wählen Sie daraufhin „Update und Sicherheit“ und anschließend „Wiederherstellung“ aus. Unter der Überschrift „Erweiterter Start“ entdecken Sie nun die Schaltfläche „Jetzt neu starten“. Bei beiden vorgestellten Möglichkeiten erscheint anschließend ein blaues Fenster mit mehreren Optionen. Hierbei müssen Sie zunächst „Problembehandlung“ und anschließend „Erweiterte Optionen“ und abschließend „UEFI Firmwareeinstellungen“ auswählen.

Unabhängig davon, ob Sie der Anweisung für Rechner mit BIOS oder mit UEFI gefolgt sind, gelangen Sie im nächsten Schritt zum Boot-Menü. Achten Sie dabei darauf, dass bei diesem Schritt der USB-Stick bereits eingelegt ist. Bei manchen Rechnern ist dies notwendig, damit diese den USB-Port als bootfähiges Laufwerk erkennen.

Das Boot-Menü kann - je nach Gerät - einige Unterschiede aufweisen. Allerdings sind stets konkrete Anweisungen vorhanden, wie Sie das System steuern können. Im Folgenden wird eine Möglichkeit dafür vorgestellt, wie Sie die Einstellungen vornehmen können. Es ist aber möglich, dass Ihr Rechner davon in manchen Punkten abweicht. In diesem Fall müssen Sie die Anweisungen im Boot-Menü beachten.

Über die Pfeiltasten müssen Sie zunächst zum Reiter „Boot“ navigieren. Hier erscheint eine Zeile mit der Beschriftung „USB Boot“. Sollte diese bereits als „Enabled“ ausgezeichnet sein, sind keine weiteren Aktionen notwendig. Sollte dies nicht zutreffen, müssen Sie sie mit der Enter-Taste auswählen und daraufhin den entsprechenden Wert vorgeben.

Weiter unten erscheint die Überschrift „Boot Priority Order“. Hier sind nun verschiedene Laufwerke angegeben, die zum Booten infrage kommen. Hier müssen Sie dafür sorgen, dass das USB-Laufwerk in der Liste vor der Festplatte erscheint. Um die einzelnen Laufwerke zu verschieben, kommen häufig die F5- und die F6-Taste zum Einsatz. Nachdem Sie diese Einstellung vorgenommen haben, können Sie „Exit“ und daraufhin „Exit Saving Changes“ auswählen, um die entsprechenden Einstellungen zu übernehmen.

Wenn Sie daraufhin den Computer bei eingelegtem USB-Stick neu starten, sollte nun das Linux Live-System erscheinen. Abbildung 3 zeigt, wie dieses bei der Version 8.6 aussieht.



Abb. 1.3 Die Oberfläche von KNOPPIX

Damit haben Sie bereits zum ersten Mal ein Linux-Betriebssystem gestartet. Um zu zeigen, was damit alles möglich ist, können Sie einmal auf den Ordner mit der Beschriftung „KNOPPIX“ klicken, der sich links oben auf der Desktop-Oberfläche befindet. Dessen Inhalt ist in Abbildung 4 zu sehen.

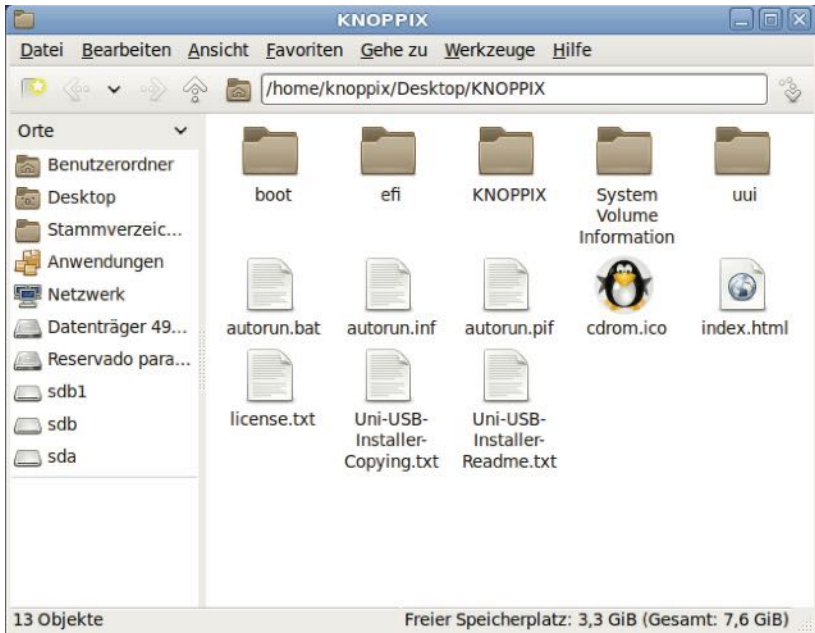


Abb. 1.4 Die Dateien der Festplatte

Wenn Sie hier in der linken Spalte auf den Begriff „Datenträger“ mit dem Festplattensymbol klicken, gelangen Sie zu den Dateien, die auf der Festplatte des Computers abgespeichert sind. Das bedeutet, dass Sie dadurch auf alle Inhalte zugreifen können, die sich auf Ihrem PC befinden. Das ist beispielsweise für die Datenrettung bei einem defekten Betriebssystem ausgesprochen hilfreich.

Im nächsten Schritt lernen wir die Programme kennen, die KNOPPIX zu bieten hat. Klicken Sie hierfür auf das Symbol in der linken unteren Ecke. Daraufhin öffnet sich ein Auswahlménü, das die unter KNOPPIX

verfügbaren Programme anzeigt. Hier können Sie nun einmal die verschiedenen Optionen ausprobieren. In Abbildung 5 sind beispielsweise die Angebote für Büro-Software zu sehen. Darüber hinaus gibt es auch verschiedene Spiele, Entwicklungswerkzeuge, Grafik-Programme, Internetanwendungen und vieles mehr.

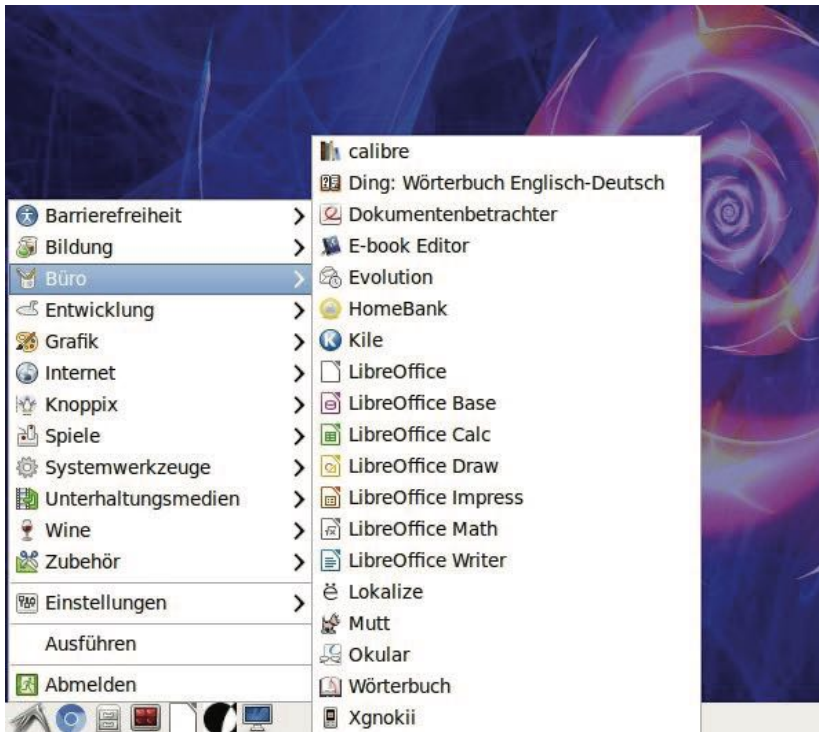


Abb. 1.5 Vielfältige Programme unter Linux

Das zeigt bereits einen der wesentlichen Vorteile bei der Verwendung von Linux. Hierbei stehen Ihnen unzählige Programme bereits vorinstalliert zur Auswahl. Diese können Sie vollkommen kostenlos nutzen. Nicht einmal der Aufwand für die Installation ist notwendig.

Zum Abschluss dieses ersten kleinen Rundgangs klicken wir in diesem Menü einmal auf „Systemwerkzeuge“ und anschließend auf „Terminal“.

Daraufhin öffnet sich das Programm, das in Abbildung 6 zu sehen ist. Hierbei handelt es sich um eine Bash-Shell. Die Shell stellt eines der wesentlichen Steuerungswerkzeuge unter Linux dar. Die Funktionsweise wird an dieser Stelle noch nicht genauer erklärt. Das folgt noch im weiteren Verlauf dieses Buchs. Doch können Sie sich bereits einmal mit diesem Werkzeug vertraut machen, da es später noch häufig zum Einsatz kommen wird.

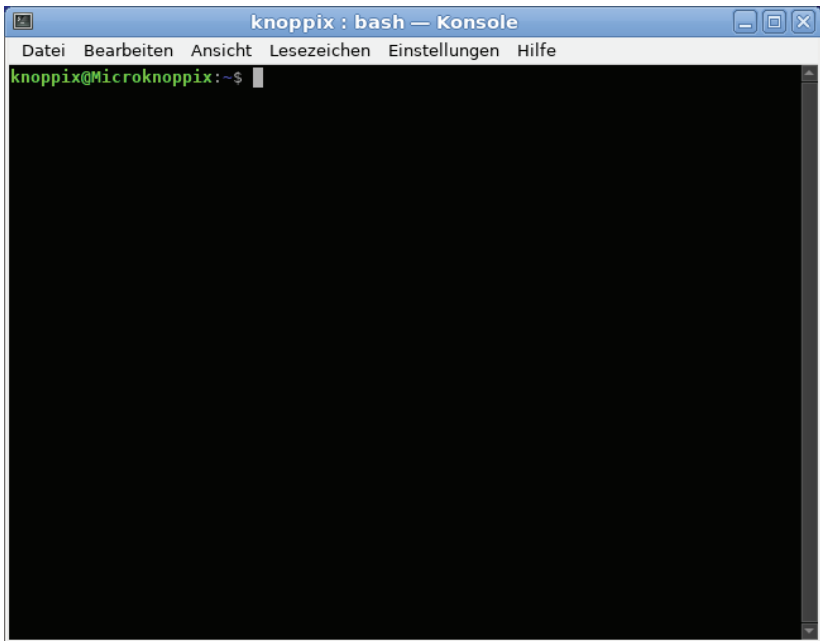


Abb. 1.6 Die Oberfläche der Bash-Shell

1.6 Die Linux-Community: So entsteht das Betriebssystem

Zum Abschluss des Einleitungs-Kapitels soll noch ein kurzer Blick darauf geworfen werden, wie die Entwicklung von Linux abläuft. Wie bei fast jeder anderen Software auch, kommt es hierbei zu einer ständigen Weiterentwicklung. Das ist nicht nur notwendig, um das Betriebssystem immer wieder an den aktuellen Stand der Technik anzupassen.

Darüber hinaus ist dies erforderlich, um die Funktionen zu verbessern und um Fehler zu beseitigen.

Bei dieser Betrachtung ist es wichtig, zwischen dem Linux-Kernel und den einzelnen Linux-Distributionen zu unterscheiden. Es wurde bereits dargestellt, dass ein Betriebssystem in zwei Bereiche aufgeteilt ist: in den Kernel, der die zentralen Funktionen des Computers steuert, und in die einzelnen Distributionen, die eine Oberfläche für die Anwender und die *Systemprogramme* bereitstellen. Während die Entwicklung des Kernels einen einheitlichen Prozess darstellt, der Auswirkungen auf alle Ausführungen von Linux hat, handelt es sich bei der Entwicklung der Distributionen um unabhängige Projekte.

1

Obwohl es sich beim Linux-Kernel um ein einheitliches System handelt, gibt es für die Entwicklung kaum feste Strukturen. Jeder Programmierer, der daran Interesse hat, kann hier mitwirken. Die Kommunikation der Linux-Community findet größtenteils per E-Mail statt. Diese Form wird insbesondere von Linus Torvalds unterstützt, da sie - seiner Meinung nach - verhindert, dass unterschiedliche Meinungen direkt aufeinanderprallen. Das soll dazu beitragen, Konflikte in der Entwicklergemeinschaft zu vermeiden. Die Mailing-Listen stehen jedem offen. Hier ist es möglich, eigene Vorschläge einzubringen und die Ideen anderer Entwickler zu bewerten. Darüber hinaus ist es problemlos möglich, selbst an der Entwicklung des Linux-Kernels mitzuwirken. Jeder Programmierer kann eigene Programme einfügen, um den Kernel zu verbessern. Bei der Linux-Community handelt es sich um eine große Gemeinschaft, an der viele Programmierer aus aller Welt teilnehmen. Diese erhalten für ihre Tätigkeit keinerlei Bezahlung. Das ist eine wichtige Voraussetzung dafür, dass das Betriebssystem kostenlos zur Verfügung steht.

Aufgrund der großen Gemeinschaft kommt es zu vielfältigen Verbesserungsvorschlägen. Zahlreiche Entwickler reichen Programmteile ein, die in den Linux-Kernel aufgenommen werden sollen. Da nicht alle Vorschläge sinnvoll sind, ist ein Entscheidungsgremium notwendig, das darüber urteilt, welche Ideen in den offiziellen Linux-Kernel aufgenommen werden. Diese Funktion übt Linus Torvalds selbst aus – mit

der Unterstützung einer kleinen Gruppe aus Programmierern, die sich bei der Entwicklung des Linux-Kernels bereits verdient gemacht haben.

Ein ganz anderes Bild ergibt sich bei den einzelnen Linux-Distributionen. Hierbei gibt es viele verschiedene Entwicklungsmodelle. In vielen Fällen sind Unternehmen für die Gestaltung zuständig, die eine kommerzielle Absicht verfolgen. Doch gibt es auch in diesem Bereich Gruppen, die unentgeltlich arbeiten und an der Verbreitung von Open-Source-Software interessiert sind. Häufig arbeiten die Entwickler der einzelnen Distributionen auch an der Gestaltung des Linux-Kernels mit. Das gibt ihnen die Möglichkeit, die Funktionen umzusetzen, die sie für ihre jeweilige Distribution für wichtig erachten. Oftmals kommt es aber auch vor, dass diese Zusätze nicht ihren Weg in den offiziellen Linux-Kernel finden. Da es sich hierbei um Open-Source-Software handelt, kann jeder Anwender den Quellcode nach seinen Bedürfnissen anpassen. Daher kommt es immer wieder vor, dass einzelne Distributionen nicht mit dem offiziellen Linux-Kernel arbeiten, sondern leichte Abwandlungen verwenden.

1.7 Übungsaufgaben

Am Ende der einzelnen Kapitel entdecken Sie immer eine oder mehrere Übungsaufgaben. Diese dienen dazu, das erworbene Wissen zu vertiefen und zu erweitern. Bei den Kapiteln, die sich mit praktischen Anwendungen befassen, sollte es möglich sein, die Aufgabe mit dem im jeweiligen Kapitel erworbenen Wissen zu lösen. In Kapiteln, die theoretische Inhalte oder Grundlagenwissen vermitteln, handelt es sich hingegen um weiterführende Inhalte, die nicht im Text zur Sprache kommen. Die Antworten können Sie dann über eine kurze Internetrecherche finden.

Dabei ist jeweils eine kurze Musterlösung angegeben. Das bedeutet jedoch nicht, dass Ihre Antwort genau gleich aussehen muss. Häufig führen in der Informatik mehrere Lösungswege ans Ziel. Die Musterlösung dient lediglich zum Abgleich mit Ihrer eigenen Lösung. Für das erste Kapitel sollen Sie die folgende Aufgabe erledigen:

Ermitteln Sie die wesentlichen Gründe, die für die Nutzung von Linux sprechen.

Musterlösung:

Unter anderem sprechen die folgenden Gründe für die Verwendung von Linux:

- ▶ Kostenfreie Nutzung möglich
- ▶ Quellenoffene Software – optimal für Lernzwecke geeignet
- ▶ Zahlreiche Entwicklungswerkzeuge für Computerprogramme enthalten
- ▶ Hohe Sicherheit
- ▶ Auch für Computer mit geringer Hardware-Ausstattung geeignet
- ▶ Dauerhaft aktuelles Betriebssystem durch ständige Weiterentwicklung
- ▶ Betriebssystem mit guter Performance
- ▶ Einfache Automatisierbarkeit von Routinen, um wiederkehrende Aufgaben automatisch ausführen zu lassen

Kapitel 2

Der Linux-Kernel

Das Betriebssystem Linux spaltet sich – genau wie alle übrigen PC-Betriebssysteme – in mehrere Ebenen auf. Diese greifen bei den Anwendungen ineinander und ermöglichen auf diese Weise die Nutzung des Computers. Eine Besonderheit bei Linux besteht darin, dass hierbei eine klare Trennung zwischen den einzelnen Ebenen besteht. Das liegt daran, dass es sich bei Linux eigentlich nur um den Kernel handelt. Damit daraus ein voll funktionsfähiges Betriebssystem entsteht, ist es notwendig, weitere Schichten hinzuzufügen. Hierfür gibt es aber zahlreiche unterschiedliche Versionen, die als Distributionen bezeichnet werden. Bei vielen anderen Betriebssystemen – beispielsweise bei Windows – ist diese Trennung nicht so deutlich. Dabei handelt es sich um ein einheitliches Produkt, bei dem der Kernel, die Systemsoftware und die Benutzeroberfläche fest miteinander verbunden sind und sich nicht trennen lassen.

Dieses Kapitel beschäftigt sich mit dem Linux-Kernel – also mit der untersten Softwareschicht. Es geht zunächst der Frage nach, welche grundlegenden Eigenschaften und Funktionen ein Kernel aufweisen muss. Danach befasst es sich mit den technischen Besonderheiten des Linux-Kernels und stellt die Entwicklungsgeschichte der verschiedenen Versionen vor.

2.1 Was ist ein Kernel?

Beim Kernel handelt es sich um den Kern eines Betriebssystems. Dementsprechend sind hierfür auch die Bezeichnungen Betriebssystemkern und Systemkern verbreitet. Er stellt die unterste Schicht der Software dar. Er hat direkten Zugriff auf die gesamte Hardware. Seine wesentliche Aufgabe besteht darin, die Hardware zu abstrahieren. Das bedeutet, dass er den übrigen Anwendern Zugang zu deren Funktionen bietet,

ohne dass es notwendig ist, dass diese die genauen Befehle für ihre Steuerung anwenden müssen. Zu diesem Zweck stellt er eine Schnittstelle zur Verfügung, auf die andere Programme zugreifen können. Diese Schnittstelle ist unabhängig von der jeweiligen Rechnerarchitektur. Das hat zur Folge, dass die einzelnen Programme die Gestaltung der Hardware nicht berücksichtigen müssen. Sie verwenden lediglich die Schnittstelle und der Kernel kümmert sich dann um die korrekte Umsetzung. Er bestimmt außerdem die Grundlagen der Prozess- und Datenorganisation auf dem Gerät. Alle weiteren Programme, die auf dem Computer ablaufen, bauen auf den Funktionen des Systemkerns auf.

In Kapitel 1.1 wurden bereits die grundlegenden Aufgaben eines Betriebssystems besprochen. Die meisten der hier genannten Aspekte übernimmt der Kernel. Dazu zählen die Speicherverwaltung, das Prozessor-Management, die Geräteverwaltung (Device Management) und die Organisation des Dateisystems.

Neben diesen grundlegenden Aufgaben muss der Kernel eines modernen Betriebssystems jedoch noch einige weitere Anforderungen erfüllen: Von großer Bedeutung ist es beispielsweise, dass er multitaskingfähig ist. Das bedeutet, dass er mehrere Prozesse gleichzeitig verwalten kann. Bei einem Prozess handelt es sich um einen linearen Ablauf von Befehlen eines einzelnen Programms. Dabei tritt jedoch das Problem auf, dass in der Regel viele Prozesse gleichzeitig ausgeführt werden sollen. Ein einfacher Prozessor kann diese jedoch nicht gleichzeitig umsetzen, denn er kann immer nur einen einzelnen Befehl bearbeiten. Lediglich bei Mehrkern-Prozessoren ist eine parallele Bearbeitung möglich. Doch übersteigt auch hierbei die Zahl der Prozesse, die ausgeführt werden sollen, fast immer die Anzahl der verfügbaren Prozessorkerne. Nun wäre es jedoch sehr ineffizient, wenn der Kernel die einzelnen Prozesse einfach nacheinander abarbeiten würde. Aus diesem Grund verfügen moderne Betriebssysteme über ein multitaskingfähiges Prozess-Management. Diese Funktion wird auch als Scheduler bezeichnet. Dabei kommt es immer wieder zu einem Wechsel zwischen den einzelnen Anwendungen. Aufgrund der geringen Intervalle führt das dazu, dass der Eindruck entsteht, als würden sie gleichzeitig ablaufen. Diese

Organisationsform dient dazu, effiziente Strukturen und damit eine optimale Performance zu erzielen. Die Aufgabe des Kernels besteht darin, zu entscheiden, welcher Prozess zu welchem Zeitpunkt welche Ressourcen erhält.

Eine weitere wichtige Anforderung an den Kernel besteht darin, dass dieser für die Einhaltung zeitkritischer Grenzen sorgt. Wenn ein Programm bestimmte Ressourcen bis zu einem gewissen Zeitpunkt unbedingt benötigt, muss er dafür sorgen, dass es diese rechtzeitig erhält. Darüber hinaus ist er für die Virtualisierung der Ressourcen zuständig. Das bedeutet, dass er für bestimmte Funktionen ein höheres Abstraktionsniveau vorgibt und den Computer anhand dieser Strukturen organisiert. Ein Beispiel hierfür wurde bereits genannt: der Prozess. Hierbei handelt es sich um ein abstraktes Konstrukt, das eigentlich aus einer einfachen Befehlsfolge besteht. Ein weiteres Beispiel hierfür ist die Datei. Hierbei handelt es sich um eine Abfolge binärer Informationen. Erst das Betriebssystem fasst diese in sinnvolle Einheiten zusammen und gibt ihnen einen Sinn.

Wenn es sich um ein Mehrbenutzersystem handelt, fällt dem Systemkern außerdem die wichtige Aufgabe zu, die Zugriffsrechte zu überprüfen. Es muss für alle Dateien einen Besitzer festlegen und bei jedem Zugriff überprüfen, ob die erforderlichen Rechte vorhanden sind.

Eine der zentralen Aufgaben des Kernels besteht auch darin, den Startvorgang des Rechners zu organisieren. Nach dem Anschalten kommt es hierbei zunächst zu einem Hardware-Check. Danach wird der sogenannte Bootloader initialisiert. Dessen Aufgabe besteht lediglich darin, den Kernel in den Arbeitsspeicher zu laden. Das bedeutet, dass es sich hierbei um das erste Programm handelt, das der Computer nach dem Starten ausführt. Daraufhin initialisiert der Kernel alle weiteren Geräte und startet einen fest vorgegebenen Prozess. Je nach Betriebssystem kann es sich hierbei um einen Kommandozeileninterpreter oder um eine grafische Oberfläche handeln. Beide Alternativen ermöglichen es, die Funktionen des Betriebssystems zu steuern und weitere Programme aufzurufen.

Es gibt mehrere unterschiedliche Arten von Systemkernen. Von großer Bedeutung sind monolithische Kernel. Diese zeichnen sich dadurch aus, dass sie nicht nur die wesentlichen Funktionen aus den Bereichen Speicher- und Prozessverwaltung innerhalb des Kernels implementieren. Darüber hinaus kommt es dabei zu einer direkten Einbindung der Treiber für verschiedene Hardwarekomponenten. Das bringt den Vorteil mit sich, dass für diese Aufgabe keine zusätzlichen Programme erforderlich sind. Das verbessert die Performance. Allerdings bringt diese Architektur auch gewisse Probleme mit sich. Dabei ist insbesondere eine höhere Fehleranfälligkeit zu nennen. Wenn bei einem monolithischen Kernel in einem Teilbereich ein Fehler auftritt, kann sich das auf das gesamte System auswirken und sogar zu einem Absturz führen.

Der Gegenentwurf zum monolithischen Kernel wird als Mikrokern bezeichnet – oder auch als μ -Kernel. Hierbei übernimmt der Kernel nur die elementaren Funktionen: die Speicher- und Prozessverwaltung sowie die Kommunikation zwischen den verschiedenen Prozessen (Inter Process Communication, IPC). Mikrokern verwenden ein Ring-Modell, in dem Programme mit unterschiedlichen Privilegien ablaufen. Der innerste Ring mit den höchsten Privilegien ist ausschließlich dem Kernel vorbehalten. Die übrigen Funktionen können beispielsweise über ein Server-Client-Modell umgesetzt werden. Das bedeutet, dass hierfür eigene Prozesse eingerichtet werden, mit denen die Anwenderprogramme kommunizieren können. Eine weitere Möglichkeit ist es, die entsprechenden Funktionen per Programmbibliothek bereitzustellen.

Darüber hinaus gibt es Hybrid-Kernel. Diese stellen einen Kompromiss zwischen einem monolithischem Kernel und einem Mikrokern dar. Das bedeutet, dass sie im Prinzip den Ansatz eines Mikrokernels verfolgen und lediglich die elementaren Funktionen anbieten. Aus Performance-Gründen werden darüber hinaus jedoch einige Zusatzfunktionen in den Kernel eingebunden. Um welche Bereiche es sich dabei genau handelt, ist nicht einheitlich. Je nach Umsetzung kann es sich dabei beispielsweise um den Netzwerktreiber, um das Grafiksystem oder um eine ganz andere Funktion handeln.

Schließlich gibt es noch sogenannte Exokernel. Diese beschränken sich im Wesentlichen darauf, die Zugriffsrechte zu prüfen und Ressourcenkonflikte zu vermeiden. Die Verwendung von Exokerneln ist jedoch ausgesprochen selten. Sie kommen fast ausschließlich bei wissenschaftlichen Modellprojekten zum Einsatz.

2.2 Die technischen Eigenschaften des Linux-Kernels

Bei Linux handelt es sich um einen monolithischen Kernel. Das bedeutet, dass er umfassende Aufgaben im Bereich der Steuerung der Hardware übernimmt. Im vorigen Abschnitt wurde jedoch bereits darauf hingewiesen, dass monolithische Systemkerne anfällig gegenüber Fehlern sind. Um dieses Problem zu reduzieren, verwendet Linux einen modularen Aufbau. Das bedeutet, dass einzelne Module – häufig handelt es sich dabei um Hardware-Treiber – separat nachgeladen werden können. Das verhindert, dass es bei einem Fehler sofort zum Absturz des gesamten Systems kommt. Das führt dazu, dass das gesamte System deutlich robuster wird.

Darüber hinaus bietet der modulare Aufbau des Linux-Kernels noch einen weiteren Vorteil: Auf diese Weise ist es - im Gegensatz zu einem strikt monolithischen Ansatz - nicht erforderlich, stets alle Treiber in den Arbeitsspeicher zu laden. Stattdessen ist es möglich, nur die Module zu verwenden, die tatsächlich erforderlich sind. Das führt zu einer effizienteren Nutzung des Arbeitsspeichers und damit zu einer besseren Performance.

Dass es sich hierbei dennoch um ein monolithisches System handelt, wird daran deutlich, dass die meisten dieser Module innerhalb des Rings 0 ablaufen. Bei diesen Ringen handelt es sich um ein zentrales Element der Rechnerarchitektur. Sie dienen dazu, eine hohe Sicherheit des Systems zu gewährleisten. Sie definieren Bereiche mit unterschiedlichen Privilegien. Beim Ring 0 handelt es sich um die unterste Schicht, die mit allen Rechten ausgestattet ist. Nur wenige Module werden unter Linux in andere Ringe ausgelagert.

Eine Besonderheit dieses Betriebssystems ist das sogenannte User Mode Linux. Hierbei handelt es sich um eine Anwendung, die es erlaubt, den eigentlichen Linux-Kernel nochmals auf einem bereits laufenden Linux-System als Prozess zu starten. Dieser ermöglicht es dem Anwender, alle Funktionen im Root-Mode – also mit allen Rechten versehen – durchzuführen. Das ist beispielsweise beim Ausprobieren der Programme bei der Software-Entwicklung sehr hilfreich. Dennoch greift dieser aufgesetzte Kernel nicht direkt auf die Hardware zu. Er leitet alle Anforderungen lediglich an den eigentlichen Kernel weiter. Das verhindert Schäden am System.

Der größte Teil des Linux-Kernels ist in der Programmiersprache C geschrieben. Nur für wenige Bereiche kam eine Assemblersprache zum Einsatz. Dabei handelt es sich um besonders wichtige Teilroutinen sowie um zeitkritische Module.

Linux steht mittlerweile für viele verschiedene Hardware-Architekturen zur Verfügung. Ursprünglich war das Betriebssystem lediglich für den Computer vorgesehen, den Torvalds selbst verwendete. Dabei handelte es sich um ein Gerät mit der Intel Architecture 32 Bit (IA-32). Später erstellte Torvalds selbst eine Portierung für einen DEC-Alpha-Rechner mit 64 Bit. Später folgten noch unzählige weitere Portierungen – für alle gängigen PCs, für Server, Smartphones (Android) und sogar für Geräte wie Digitalkameras.

2.3 Verschiedene Kernel-Versionen

Bei der Entwicklung des Linux-Kernels handelt es sich um einen stetigen Prozess. Das ist notwendig, um Fehler zu beheben, neue Funktionen hinzuzufügen und um die Sicherheit zu gewährleisten. Daher ist es sinnvoll, einen Blick auf die wichtigsten Kernel-Versionen zu werfen und das System, das hinter den Bezeichnungen steht, zu erklären.

Die erste Version war Linux 0.01. Hierbei handelte es sich um die ursprüngliche Ausführung, mit dem Torvalds die Entwicklung des Betriebssystems begann. Diese Version war jedoch noch nicht stabil.

Daraufhin folgten einige weitere Test-Versionen – bis hin zu Linux 0.99. Die Nummerierungen erfolgten damals aber noch keinem klaren Schema.

Die Version 1.0.0 stellte dann die erste stabile Version des Linux-Kernels dar. Die Veröffentlichung erfolgte im März 1994. Hierbei wurde zunächst eine Versionsnummer mit drei Zahlen eingeführt, die jeweils durch einen Punkt voneinander getrennt waren. Die erste Nummer sollte nur bei grundlegenden Änderungen der verwendeten Systemarchitektur verändert werden. Die zweite Zahl war für Majorreleases vorgesehen. Dabei handelt es sich um größere Anpassungen des Systems, die erhebliche Auswirkungen auf die Funktionsweise haben. Die dritte Zahl war hingegen für Minorreleases vorgesehen – also für kleinere Veränderungen der Funktionsweise.

Die Version 2.0.0 erschien im Juni 1996. Diese brachte erhebliche Erweiterungen der Funktionen mit sich. Von besonderer Bedeutung war dabei die Einführung eines symmetrischen Multiprozessorsystems. Dieses erleichterte die Verwendung des Betriebssystems auf Geräten mit mehreren Prozessoren.

In den folgenden Jahren kam es zu mehreren Majorreleases. Von besonderer Bedeutung war dabei die Version 2.6.0, die ab Dezember 2003 verfügbar war. Diese brachte zahlreiche Neuerungen mit sich. Eine der wichtigsten Anpassungen stellte eine Verbesserung bei der Unterstützung von 64 Bit-Systemen.

Ab der Version 2.6.8.1 kam es zur Einführung einer vierten Zahl. Diese diente dazu, Versionen zu kennzeichnen, die der Fehlerbehebung dienten, aber keine neuen Funktionen mit sich brachten. Insbesondere nach der Veröffentlichung einer neuen Version fallen häufig Fehler auf, die im Abstand von nur wenigen Stunden korrigiert werden. Nach dem bisherigen System war es nicht möglich, diese einzelnen Versionen zu kennzeichnen. Da damit jedoch keine Änderungen bei den Funktionen einhergehen, ist diese vierte Zahl nur von geringer Bedeutung. Häufig wird sie bei der Angabe der Versionsnummer sogar weggelassen.

2011 kam es dann zu einer erneuten Umstellung der Nummerierung. Die damals aktuelle Version war Linux 2.6.39. Zum einen wurden dabei die Versionsnummern für die Minorreleases bereits sehr hoch, was zu einer wenig übersichtlichen Darstellung führte. Zum anderen wollte Torvalds aus Anlass des 20. Geburtstags des Betriebssystems die Versionsnummer 3 verwenden – als Symbol für das dritte Jahrzehnt. Daher wurde im Mai dieses Jahres Linux 3.0 vorgestellt, obwohl es sich dabei eigentlich nur um ein Majorrelease handelte. Dabei wurde die Anzahl der Ziffern wieder auf drei begrenzt. In den folgenden Jahren kamen noch zahlreiche weitere Versionen heraus. Im Moment (Stand: September 2020) ist die aktuelle Ausführung Linux 5.8.

2.4 Übungsaufgaben

Finden Sie heraus, wann die letzte Hauptversion (Linux 5.0) veröffentlicht wurde und welche Neuerungen sie mit sich brachte.

Lösung:

Die Veröffentlichung von Linux 5.0 erfolgte am 3. März 2019. Zu den wesentlichen Neuerungen zählten:

- ▶ Unterstützung für den Prozessor AMD Radeon FreeSync
- ▶ Energy-Scheduling-Funktion für eine bessere Energieverwaltung bei Prozessoren des Typs ARM big.LITTLE
- ▶ Neue Treiber für Intel Stratix 10 FPGA
- ▶ Unterstützung der neuen Verschlüsselungstechnik Adiantum
- ▶ Neue Multimedia-Treiber (z.B. ASpeed video engine)

Kapitel 3

Linux-Distributionen

Ein arbeitsfähiges Betriebssystem besteht nicht nur aus dem Kernel. Darüber hinaus sind viele weitere Bestandteile notwendig. Hierzu zählen etwa eine Benutzeroberfläche für die Steuerung und die System-Software, die es erlaubt, die verschiedenen Funktionen des Computers abzurufen.

Als Linus Torvalds das Betriebssystem Linux entwickelte, handelte es sich dabei jedoch lediglich um den Kernel. Dieser stellt für sich genommen kein vollständiges Betriebssystem dar. Um ihn zu nutzen, sind einige Anpassungen notwendig. Diese erfordern jedoch eine sehr hohe Sachkenntnis und bringen einen erheblichen Aufwand mit sich. Der größte Teil der Anwender ist daher sicherlich nicht in der Lage dazu, diese durchzuführen.

Um die Verwendung dennoch zu ermöglichen, entstanden bereits bald nach der Entwicklung des Linux-Kernels die ersten Linux-Distributionen. Diese enthalten alle weiteren Elemente, die notwendig sind, um ein arbeitsfähiges Betriebssystem zu erstellen. Daher ist es wichtig, auch einen Blick auf deren Funktion und Aufgaben zu werfen. Dieses Kapitel stellt vor, durch welche Eigenschaften sich die Distributionen auszeichnen und welche Möglichkeiten in diesem Bereich bestehen.

3.1 Was ist eine Linux-Distribution?

Bei einer Linux-Distribution – häufig auch als Distro bezeichnet – handelt es sich um eine Zusammenstellung unterschiedlicher Software-Produkte, die gemeinsam ein einsatzfähiges Betriebssystem ergeben. Die Gemeinsamkeit aller Linux-Distributionen besteht darin, dass diese auf den Linux-Kernel aufgesetzt sind – also dass sie diesen für die Steuerung der Hardware des Geräts nutzen. Allerdings gibt es hierbei

auch kleinere Abweichungen zwischen den einzelnen Angeboten. Zum einen ist es möglich, dass die verschiedenen Distributionen unterschiedliche Versionen des Linux-Kernels verwenden. Zum anderen führen zahlreiche Anbieter auch selbst Veränderungen am Kernel durch. Da es sich bei Linux um ein quellenoffenes System handelt, steht es nicht nur jedem Anwender frei, den Quellcode zu betrachten. Darüber hinaus kann er Änderungen daran vornehmen, um das System an seine Bedürfnisse anzupassen. Einige Entwickler von Linux-Distributionen nutzen diese Freiheit, um Änderungen am Linux-Kernel vorzunehmen. Das führt dazu, dass selbst der Kernel nicht bei allen Distributionen einheitlich ist.

Der Distribution kommt eine entscheidende Rolle bei der Verwendung von Linux zu. Sie legt zum einen das Erscheinungsbild des Betriebssystems fest. Verschiedene Linux-Distributionen wirken optisch sehr unterschiedlich und weisen auch verschiedene Organisationsformen auf. Zum anderen bestimmt sie die Funktionen, die das Betriebssystem anbietet und die Programme, die darauf nutzbar sind. Darüber hinaus organisieren die Distributionen in der Regel den kompletten Installationsprozess. Wenn Sie eine Linux-Distribution bei einem Anbieter herunterladen, ist diese normalerweise mit einem Installationsprogramm ausgestattet. Dieses ermöglicht eine einfache Einrichtung des Systems. Hinsichtlich der Komplexität und des Aufwands für den Installationsprozess bestehen jedoch erhebliche Unterschiede zwischen den einzelnen Distributionen.

Die Entstehung der ersten Linux-Distributionen

Als Torvalds Linux 1991 der Öffentlichkeit präsentierte, handelte es sich dabei lediglich um den Kernel. Dessen direkte Nutzung setzte weit fortgeschrittene Kenntnisse im Bereich der Informatik voraus. Allerdings war das neue Betriebssystem zu dieser Zeit ohnehin nur einem stark eingegrenzten Fachpublikum bekannt. Hierbei handelte es sich fast ausschließlich um Informatiker mit hohen Fachkenntnissen, die daher in der Lage dazu waren, den Kernel zu verwenden.

Das änderte sich jedoch zu dem Zeitpunkt, als Linux unter der GNU GPL als freie Software veröffentlicht wurde. Diese Ankündigung hierfür erfolgte bereits im Januar 1992 in einem Protokoll zur Version 0.12 – auch wenn erst die Version 0.99 am Ende jenes Jahres unter der GNU GPL veröffentlicht wurde. Damit wurde Linux nicht nur zu einer freien Software, darüber hinaus erhöhte sich die Aufmerksamkeit schlagartig. Das hatte zur Folge, dass immer mehr Menschen Linux verwenden wollten. Aus diesem Grund wurde schnell klar, dass es notwendig war, auf dem Kernel aufbauend ein vollständiges Betriebssystem zu entwickeln.

Daher erschienen bereits im Verlaufe des Jahres 1992 die ersten Linux-Distributionen. Den Anfang machte die Universität Manchester mit MCC Interim Linux. Etwas später folgte TAMU. Diese Distributionen enthielten allerdings nur den Linux-Kernel und die unverzichtbaren Basis-Applikationen. Jedoch erschien noch im gleichen Jahr Softlanding Linux System (SLS). Hierbei handelte es sich um die erste Linux-Distribution, die etwas umfangreichere Funktionen beinhaltete. Außerdem war SLS das erste Linux-System mit einer grafischen Benutzeroberfläche. All diese Distributionen standen kostenfrei zum Download zur Verfügung.

Mit Yggdrasil Linux erschien ebenfalls 1992 die erste kommerzielle Linux-Version. Diese zeichnete sich unter anderem dadurch aus, dass sie als CD-Rom erhältlich war. Darüber hinaus verfügte sie über eine automatische Hardware-Erkennung. Das erleichterte den Installationsvorgang auch für Nicht-Informatiker deutlich. Deshalb wurde diese Distribution damals auch als „Plug-and-Play-Linux“ bezeichnet.

1993 kam dann Slackware hinzu. Dabei handelt es sich um die älteste Version, die bis heute fortgeführt wird. In den folgenden Jahren stießen immer mehr Anbieter hinzu. Die Internetseite <https://lwn.net/>, die sich mit der Beobachtung der verfügbaren Linux-Distributionen beschäftigt, listet beispielsweise im Moment (Stand: September 2020) mehr als 250 aktive Distributionen auf. Das zeigt, wie vielfältig das Angebot in diesem Bereich ist.

Entwicklung und Vertrieb

Für die Entwicklung des Linux-Kernels ist eine große Gruppe von freiwilligen Programmierern verantwortlich, die diese Aufgabe unentgeltlich übernehmen. Diese kümmern sich jedoch nur um den Kernel, nicht jedoch um die einzelnen Distributionen. Für deren Entwicklung sind jeweils eigene Entwickler-Teams verantwortlich. Diese sind allerdings ganz unterschiedlich zusammengesetzt.

Häufig handelt es sich hierbei um Freiwillige, die unentgeltlich einen Beitrag zur Entwicklung freier Software leisten. Darüber hinaus gibt es aber auch viele Distributionen, mit deren Erstellung sich die Entwickler hauptberuflich beschäftigen. Für die Finanzierung kommen dabei jedoch ganz unterschiedliche Modelle zum Einsatz.

Manche Linux-Distributionen basieren auf Spenden. Sie stehen zwar kostenfrei zur Verfügung. Allerdings bitten die Entwickler alle Anwender, die dazu finanziell in der Lage sind, um einen freiwilligen Beitrag. Darüber hinaus gibt es auch Stiftungen, die für die Finanzierung einer Linux-Distribution aufkommen. Ein anderes Geschäftsmodell besteht darin, die Linux-Distribution zwar kostenfrei zur Verfügung zu stellen, dafür allerdings einen kostenpflichtigen Support anzubieten. Insbesondere gewerbliche Anwender benötigen häufig professionelle Hilfe bei der Einrichtung und der Wartung der Systeme. Die Einnahmen, die hierbei entstehen, dienen der Finanzierung der Distribution. Darüber hinaus gibt es auch einige kommerzielle Linux-Distributionen, für deren Verwendung eine Lizenzgebühr anfällt. Hierbei handelt es sich in der Regel um Server-Systeme und um andere kommerzielle Anwendungen. Für Privatanwender spielen kommerzielle Linux-Distributionen daher kaum eine Rolle. Häufig bieten die Entwickler kommerzieller Distributionen auch eine reduzierte Gratis-Version für private Nutzer an.

Der Vertrieb erfolgt in der Regel über das Internet. Meistens ist es möglich, die gewünschte Distribution direkt auf der Homepage des Anbieters herunterzuladen. Der Verkauf über den Einzelhandel, der bei kommerziellen Betriebssystemen wie Windows üblich ist, stellt bei Linux die Ausnahme dar.

Die Bestandteile einer Linux-Distribution

Bei einer Linux-Distribution handelt es sich um eine Zusammenstellung verschiedener Software-Produkte. Im Mittelpunkt steht dabei der Linux-Kernel, der bei jeder Linux-Distribution mitgeliefert wird. Die einzelnen Distributionen werden dann in der Regel um ein Programm für die Paketverwaltung herum aufgebaut. Dabei handelt es sich um eine Software, die es ermöglicht, weitere Programme zu installieren, zu warten und zu löschen.

Darüber hinaus sind hier alle erforderlichen Systemprogramme enthalten. Dabei handelt es sich um die Software, die zur Steuerung des Computers erforderlich ist (beispielsweise für Hardware-Treiber, falls diese noch nicht im Kernel integriert sind) und um eine Oberfläche für die Steuerung durch den Anwender. Bei der verwendeten Oberfläche handelt es sich um einen zentralen Aspekt, der die verschiedenen Linux-Distributionen auszeichnet. Die meisten Angebote verwenden hierfür eine grafische Oberfläche. Manche Distributionen sind jedoch nur mit einem *Command Line Interface* (CLI) ausgestattet, das ausschließlich schriftliche Eingaben erlaubt.

Für die Systemprogramme kommt in erster Linie GNU-Software zum Einsatz. Von besonderer Bedeutung ist hierbei etwa der GCC-Compiler, der es ermöglicht, Programme für verschiedene Computer-Architekturen zu kompilieren. Auch verschiedene GNU-Programmbibliotheken – allen voran glibc – sind für das Funktionieren des Systems von großer Bedeutung. Hinzu kommt die Shell, die die Steuerung des gesamten Systems erlaubt. Die Verwendung der GNU-Software führt dazu, dass die meisten Linux-Distributionen eine erhebliche Ähnlichkeit zueinander aufweisen. Außerdem sorgt sie für eine intensive Verbindung zum UNIX-Betriebssystem.

Die Verwendung der GNU-Software ist jedoch nicht zwingend notwendig. Es gibt auch Distributionen, die ohne diese auskommen. Ein bekanntes Beispiel hierfür ist das Smartphone-Betriebssystem Android. Dieses verwendet zwar den Linux-Kernel, aber keine GNU-Software. Das führt zu deutlichen Unterschieden bei der Anwendung, weshalb

diese Betriebssysteme kaum als Linux-Distribution wahrgenommen werden. Es gibt sogar eine kontroverse Diskussion darüber, ob es sich bei Android überhaupt um eine Linux-Distribution handelt.

Neben der Systemsoftware bieten die meisten Linux-Distributionen auch zahlreiche Anwenderprogramme an. Dabei handelt es sich beispielsweise häufig um Office- oder E-Mail-Programme, um Multimedia-Anwendungen, Spiele oder Webbrowser. Welche Anwenderprogramme enthalten sind, hängt stark vom Einsatzzweck der entsprechenden Distribution ab.

Unterschiede zwischen den verschiedenen Distributionen

Die Unterschiede zwischen den verschiedenen Linux-Distributionen sind enorm. Das beginnt bereits damit, ob sie als freie Software oder kostenpflichtig zur Verfügung stehen. Ein weiterer wichtiger Aspekt besteht darin, für welche Geräte sie vorgesehen sind. Darüber hinaus gibt es Unterschiede hinsichtlich der unterstützten Software. Manche Distributionen erlauben nur die Verwendung von Open-Source-Software. Andere Alternativen ermöglichen hingegen auch die Verwendung proprietärer Programme.

Im Zentrum der meisten Linux-Distributionen steht - wie bereits erwähnt - ein Paket-Management-System. Dieses organisiert die Installation der Software auf dem System. Die verwendeten Systeme sind hierbei jedoch nicht kompatibel zueinander.

Diese Inkompatibilität stellt auch eines der wesentlichen Probleme von Linux dar. Wenn ein Entwickler eine Software herstellt, wäre es mit einem erheblichen Aufwand verbunden, für jede einzelne Distribution eine passende Version zu erstellen. Nur wenige Hersteller verfügen über die hierfür notwendigen Ressourcen. Open-Source-Projekte beschränken sich daher häufig darauf, den Quellcode zu veröffentlichen. Daraus ein lauffähiges Programm zu erstellen, erfordert jedoch eine hohe Fachkenntnis und ist anfällig für Fehler. Anwender mit geringen Informatik-Kenntnissen bleiben daher in der Regel auf die Software be-

schränkt, die mit der entsprechenden Distribution mitgeliefert wird. Für kommerzielle Software-Entwickler scheidet die Veröffentlichung des Quellcodes grundsätzlich aus. Daher gestalten diese in der Regel nur einzelne Versionen für die gebräuchlichsten Distributionen. Das führt wiederum dazu, dass weniger verbreitete Alternativen insbesondere in einem gewerblichen Umfeld kaum als Arbeitsplattform infrage kommen, da die hierfür benötigte Software nicht zur Verfügung steht.

3.2 Verschiedene Arten von Distributionen

3

Es gibt mehrere verschiedene Arten von Linux-Distributionen. Um eine passende Software auszuwählen, ist es daher zunächst erforderlich, sich mit diesen Unterschieden zu befassen. Die verschiedenen Distros unterscheiden sich zum einen darin, ob es sich dabei um ein *Live-System* oder um ein fest installiertes Betriebssystem handelt. Zum anderen bestehen erhebliche Unterschiede hinsichtlich der Geräte, für die die Software vorgesehen ist.

Live-Systeme vs. fest installierte Betriebssysteme

Das Betriebssystem stellt die wesentliche Grundlage aller Funktionen des Computers dar. Daher gehen die meisten Anwender davon aus, dass es notwendig ist, dieses fest auf dem Gerät zu installieren. Das ist jedoch nicht zwingend erforderlich. Es gibt auch sogenannte Live-Systeme. Ein Beispiel hierfür haben wir bereits im ersten Kapitel kennengelernt. Dabei haben wir gesehen, dass sich das Betriebssystem dabei auf einem externen Datenträger abspeichern lässt – beispielsweise auf einer DVD oder wie im vorgestellten Beispiel auf einem USB-Stick. Beim Einschalten des Computers ist es dann möglich, das Gerät mit diesem System zu booten.

Dafür ist keine Installation notwendig. Außerdem ist es nicht erforderlich, Daten auf die Festplatte zu schreiben. Das komplette Betriebssystem wird lediglich vom externen Datenträger in den Arbeitsspeicher geladen und läuft hier ab. Live-Systeme kommen zum einen häufig

zum Einsatz, um die entsprechende Linux-Distribution kennenzulernen und auszuprobieren. Zum anderen sind sie sehr beliebt für die Datenrettung. In Kapitel 1 wurde bereits gezeigt, dass es damit auch möglich ist, auf die Daten des eigentlichen Betriebssystems zuzugreifen. Wenn dieses defekt ist und sich der Rechner deshalb nicht mehr starten lässt, ist es mit einem Live-System möglich, die Informationen auf ein anderes Speichermedium zu übertragen und sie auf diese Weise zu retten.

Distributionen, die ausschließlich als Live-System erhältlich sind, sind sehr selten. Selbst bei Knoppix, das im ersten Kapitel als Beispiel für ein Live-System vorgestellt wurde und das fast ausschließlich zu diesem Zweck zum Einsatz kommt, ist mittlerweile auch eine feste Installation möglich. Viele Distributionen sind in zwei Ausführungen erhältlich – eine als Live-System und die andere als fest installiertes Betriebssystem. Die meisten Anbieter wollen es den Anwendern auf diese Weise ermöglichen, zunächst die Funktionen kennenzulernen. Darüber hinaus ist es meistens möglich, den Installationsprozess direkt aus dem Live-System zu starten. Allerdings ist es wichtig, zu beachten, dass nicht alle Distributionen als Live-System zur Verfügung stehen.

Linux-Distributionen für unterschiedliche Gerätetypen

Die einzelnen Linux-Distributionen sind immer für ganz bestimmte Arten von Geräten vorgesehen. Manchmal ist es zwar möglich, sie auf einem anderen Typ zu installieren, doch führt das zu Problemen bei der Steuerung und bei der Performance. Daher ist es bei der Auswahl der Distribution sehr wichtig, auch diesen Aspekt zu berücksichtigen.

Viele Distributionen eignen sich für gewöhnliche Computer. Das stellt den ursprünglichen Verwendungszweck des Betriebssystems dar. Dieses Buch konzentriert sich auf diesen Aspekt. Es bestehen jedoch noch viele weitere Möglichkeiten, die teilweise in den vorherigen Abschnitten bereits zur Sprache kamen.

Zahlreiche Distributionen sind speziell auf die Anwendung auf einem Server optimiert. Sie halten alle hierfür erforderlichen Funktionen

bereit und bieten eine optimale Unterstützung für die umfangreiche Hardware-Ausstattung, die bei diesen Geräten üblich ist.

Bei Smartphone-Distributionen handelt es sich um sehr spezielle Systeme, die erhebliche Unterschiede zu anderen Alternativen aufweisen. Sie zeichnen sich beispielsweise durch eine Unterstützung der Touchscreen-Steuerung aus und sie bieten zahlreiche zusätzliche Funktionen – etwa für die Telefonie und den SMS-Versand.

Darüber hinaus gibt es eingebettete Linux-Systeme. Diese eignen sich für verschiedene Haushaltsgeräte und Maschinen, bei denen es sich nicht um einen Computer handelt. Eingebettete Systeme weisen große Unterschiede zu den Distributionen auf, die auf einem gewöhnlichen Computer zum Einsatz kommen. Das beginnt bereits bei der Steuerung. Eine Waschmaschine, ein Smart-TV-Gerät oder eine Digitalkamera zeichnet sich durch ganz andere Steuerungselemente als ein Computer aus. Auch die Funktionen sind dabei genau auf die jeweiligen Geräte ausgelegt. Das führt dazu, dass hierbei in der Regel sehr umfangreiche Anpassungen notwendig sind und dass häufig auch der Linux-Kernel abgewandelt wird.

3.3 Die wichtigsten Linux-Distributionen im Überblick

Es wurde bereits gesagt, dass die Liste der verschiedenen Linux-Distributionen ausgesprochen lang ist. Hierbei bestehen mehrere Hundert Möglichkeiten. Diese hier alle aufzuführen, ist nicht möglich. Allerdings gibt es dabei einige größere Gruppen. Im Laufe der Jahre entstanden verschiedene Distributionen, die einen großen Einfluss gewannen. Das führte dazu, dass andere Programmierer diese weiterentwickelten und abänderten. Dennoch blieben dabei einige Gemeinsamkeiten zur ursprünglichen Version erhalten. Das gemeinsame Erbe führte dazu, dass auch die einzelnen Ableitungen untereinander zahlreiche Ähnlichkeiten aufweisen. Auf diese Weise entstanden die Gruppen, die in den folgenden Abschnitten vorgestellt werden.

Debian

Eine der wichtigsten Linux-Distributionen ist ohne Zweifel Debian. Diese entstand bereits 1993 – also recht kurz nach der Entwicklung des Linux-Kernels. Sie trug erheblich zur Verbreitung von Linux bei. Debian entstand aus einer Initiative des US-amerikanischen Informatikers Ian Murdock. Mittlerweile arbeiten jedoch mehr als 1.000 Programmierer an diesem Projekt mit. Debian zeichnet sich zum einen durch die Unterstützung von zwölf verschiedenen Prozessor-Architekturen aus. Das ist mehr als bei jeder anderen binären Linux-Distribution. Zum anderen enthält Debian ausgesprochen viele Anwenderprogramme. Insgesamt sind hier mehr als 57.000 Programmpakete verfügbar. Das stellt sehr vielfältige Anwendungsmöglichkeiten sicher. Diese Vorteile führten dazu, dass sich viele Anwender für Debian entschieden. Außerdem diente die Software als Vorbild zahlreicher weiterer Distributionen und hatte dadurch einen enormen Einfluss.

Debian stellte den Ursprung vieler weiterer Distributionen dar. Beispiele hierfür sind DebianEdu, Kanotix und Parrot OS. Ein weiteres bekanntes Debian-Derivat ist Knoppix. Hierbei handelte es sich um eine der ersten Live-Versionen dieses Betriebssystems.

Von besonderer Bedeutung ist außerdem Ubuntu. Auch hierbei handelt es sich um ein Debian-Derivat. Ubuntu zählt heutzutage zu den am häufigsten genutzten Linux-Distributionen. Dieses Projekt entstand 2004 auf Initiative des südafrikanischen Unternehmers Mark Shuttleworth. Das Ziel dieser Distribution bestand darin, die Verwendung von Linux genauso einfach zu machen - wie bei anderen Betriebssystemen auch. Auf diese Weise sollte Menschen in aller Welt – insbesondere auch in den Entwicklungsländern – ein kostenfreies Betriebssystem zur Verfügung gestellt werden, für dessen Verwendung keine fortgeschrittenen Informatik-Kenntnisse erforderlich sind. Außerdem legt Ubuntu hohen Wert auf die Barrierefreiheit. Ein Beispiel hierfür ist die Übersetzung in zahlreiche, auch wenig gebräuchliche Sprachen. Das soll den Menschen in den entsprechenden Ländern einen einfachen Zugang zu einem Computer ermöglichen. Auf diese Weise gelang es Ubuntu, Linux massentauglich zu machen. Daher kam es durch diese Distribu-

tion zu einer deutlichen Erhöhung der Nutzerzahlen. Neben der eigentlichen Ubuntu-Distribution sind noch einige Abwandlungen davon erhältlich – beispielsweise Kubuntu, Xubuntu oder Ubuntu MATE.

Von Ubuntu und damit in zweiter Instanz auch von Debian abgeleitet ist eine weitere beliebte Linux-Distribution: Linux Mint. Das Ziel bestand hierbei darin, auch die Verwendung proprietärer Software zu erlauben, was bei Ubuntu ursprünglich nicht möglich war. Später entstand davon noch eine weitere Abwandlung: Linux Mint Debian Edition (LMDE). Diese wurde direkt von Debian abgeleitet – ohne den Umweg über Ubuntu.

Arch

Arch Linux stellt eine weitere Linux-Distribution mit großer Bedeutung dar. Die Entwicklung begann 2001. Arch Linux zeichnet sich dadurch aus, dass hierbei Rolling Releases zum Einsatz kommen. Das bedeutet, dass es hierbei keine klar abgegrenzten Versionen gibt. Stattdessen kommt es zu einer kontinuierlichen Entwicklung. Eine weitere Besonderheit dieser Distribution stellt es dar, dass sie sehr streng das KISS-Prinzip („Keep it simple, stupid“) befolgt. Das bedeutet, dass alle Funktionen des Betriebssystems so einfach wie möglich gehalten sein sollen. Das hat unter anderem den vollständigen Verzicht auf grafische Oberflächen bei der Installation und Konfiguration zur Folge. Diese würden nach Ansicht der Entwickler die eigentlichen Funktionen des Betriebssystems verdecken. Das führt jedoch dazu, dass für die Einrichtung von Arch Linux erhebliche Computer-Kenntnisse erforderlich sind. Daher eignet sich diese Distribution ausschließlich für fortgeschrittene Anwender.

Allerdings gibt es auch Ableitungen von Arch, bei denen Anfänger ebenfalls zur Zielgruppe gehören. Zunehmender Beliebtheit erfreut sich beispielsweise Manjaro Linux. Hierbei handelt es sich zwar um ein Arch-Derivat. Dennoch sind dabei grafische Oberflächen für die Installation enthalten, sodass auch unerfahrene Anwender dieses System nutzen können. Für die Entwicklung ist ein internationales Team mit

Programmierern aus Deutschland, Frankreich und Österreich verantwortlich.

Red Hat Linux

Eine sehr einflussreiche Distribution war Red Hat Linux. Diese wurde 1994 präsentiert. Sie verwendete den grafischen Installer Anaconda, der den Installationsprozess deutlich erleichterte. Das stellte einen der wesentlichen Grundsteine für den Erfolg der Distribution dar. Red Hat Linux zeichnete sich dadurch aus, dass die Entwicklung komplett innerhalb des Unternehmens Red Hat stattfand. Für die meisten anderen Distributionen waren damals hingegen freie Entwickler-Teams verantwortlich. Red Hat war zwar frei erhältlich, seinen Gewinn erzielte das Unternehmen jedoch aus kostenpflichtigen Support-Angeboten. Mittlerweile wurde die Entwicklung von Red Hat Linux allerdings eingestellt.

Dennoch ist diese Distribution auch heute noch von großer Bedeutung. Das liegt daran, dass hierbei zwei Nachfolger existieren, die nach wie vor zu den beliebtesten Linux-Distributionen gehören. 2003 beschloss Red Hat, nur noch den Unternehmensmarkt zu bedienen. Aus diesem Grund stellte das Unternehmen die Entwicklung von Red Hat Linux ein. Diese Distribution ging daher in das Fedora-Projekt über. Hierbei handelt es sich um eine freie Community. Allerdings hat das Unternehmen Red Hat dabei nach wie vor einen erheblichen Einfluss. Fedora stellt eine einflussreiche Linux-Distribution für Privatanwender dar.

Außerdem entstand zu dieser Zeit Red Hat Enterprise Linux (RHEL). Diese Distribution ist ausschließlich für Unternehmen vorgesehen und nimmt in diesem Bereich eine marktbeherrschende Stellung ein.

Gentoo

Gentoo ist eine Distribution, die sich an fortgeschrittene Anwender richtet. Der Installationsprozess ist hierbei recht schwierig. Er bietet

jedoch sehr umfassende Individualisierungsmöglichkeiten. Das macht es möglich, das Betriebssystem auf viele verschiedene Anforderungen auszurichten. Voraussetzung für die Installation ist jedoch eine umfassende Kenntnis der Abläufe des Linux-Betriebssystems. Außerdem ist es ratsam, sich intensiv mit der Dokumentation der Distribution auseinanderzusetzen. Da diese sehr umfangreich ist, ist hierfür sehr viel Zeit erforderlich.

Diese Eigenschaften führen dazu, dass es sich bei Gentoo um ein Nischenprodukt handelt, das nur für relativ wenige Anwender infrage kommt. Dennoch gewann diese Distribution einige Bedeutung. Das liegt daran, dass der Konzern Google sie seit 2010 für die Entwicklung seines Betriebssystems Chromium OS – und damit abgeleitet auch für Chrome OS – verwendet. Diese Produkte entstanden zwar bereits etwas früher, allerdings basierten die ersten Versionen auf Ubuntu. 2010 erfolgte dann die Umstellung auf Gentoo als Basis-Version.

Slackware

Slackware entstand bereits 1992 und stellt damit eine der ersten Linux-Distributionen dar. Dabei handelt es sich außerdem um die älteste Linux-Distribution, die heute noch existiert. Slackware zeichnet sich dadurch aus, dass es die UNIX-Prinzipien sehr streng befolgt. Anwender, die Wert auf eine große Ähnlichkeit zu UNIX legen, sind daher mit Slackware gut beraten. Zu den Anfangszeiten von Linux war Slackware ausgesprochen beliebt. Mittlerweile hat die Bedeutung im Vergleich zu anderen Distributionen jedoch stark abgenommen.

Allerdings gibt es hierbei ein Derivat, das auch heute noch sehr beliebt ist: SUSE-Linux. Für die Entwicklung ist die SUSE GmbH verantwortlich, ein deutsches Unternehmen mit Sitz in Nürnberg. Hierbei gibt es zwei unterschiedliche Alternativen. Das Betriebssystem openSUSE ist quelloffen und richtet sich an private Anwender. SUSE Linux Enterprise ist hingegen für geschäftliche Anwendungen vorgesehen. Insbesondere im deutschsprachigen Raum hat SUSE Linux eine wichtige Bedeutung.

3.4 Eine passende Linux-Distribution auswählen

Der letzte Abschnitt hat gezeigt, dass es nahezu unzählig viele verschiedene Linux-Distributionen gibt. Das gibt dem Anwender zwar eine hervorragende Auswahl. Es stellt ihn aber auch vor das Problem, dass es nicht einfach ist, eine passende Version auszuwählen. Aus diesem Grund stellt der letzte Abschnitt dieses Kapitels vor, welche Aspekte bei der Auswahl von Bedeutung sind. Auf diese Weise wird es wesentlich einfacher, sich für eine passende Distribution zu entscheiden.

Server oder Desktop?

Eine wesentliche Frage, die Sie sich vor der Auswahl einer Linux-Distribution stellen sollten, besteht darin, ob das Betriebssystem auf einem Desktop-PC oder auf einem Server installiert werden soll. Hierfür gibt es jeweils unterschiedliche Versionen. Zwar wäre es grundsätzlich auch möglich, die einzelnen Distributionen an eine unterschiedliche Verwendung anzupassen, doch würde dies einen enormen Aufwand mit sich bringen und sehr umfangreiche Computerkenntnisse erfordern. Deshalb ist es sinnvoll, von Beginn an eine passende Version auszuwählen.

Die Anforderungen in diesen beiden Bereichen weisen erhebliche Unterschiede auf. Das betrifft in erster Linie die verfügbaren Funktionen. Ein Server muss ganz andere Dienste bereitstellen als ein Desktop-System. Hinzu kommt, dass bei einem Server die Stabilität des Systems eine herausragende Rolle spielt. Daher sind hierbei gut erprobte und bewährte Versionen sehr wichtig, auch wenn dies mit längeren Update-Zyklen verbunden ist. Bei Desktop-Betriebssystemen wünschen sich die meisten Anwender hingegen einen schnellen Zugang zu aktueller Software, selbst wenn die neueste Version eventuell zu Beginn etwas instabil ist.

Es gibt zahlreiche Distributionen, die ausschließlich für Server vorgesehen sind. Ein bekanntes Beispiel hierfür ist Red Hat Enterprise Linux. Viele Distributionen bieten aber auch zwei unterschiedliche Ausführungen an – eine für Desktop-Computer und eine für Server. Ein Beispiel hierfür ist Ubuntu, das in verschiedenen Versionen erhältlich ist.

Obwohl Linux insbesondere beim Betrieb von Servern von enormer Bedeutung ist, beschränken wir uns in diesem Buch auf Desktop-Systeme. Zum einen ist hierbei die Komplexität etwas geringer, sodass die Einrichtung für Anfänger leichter ist. Zum anderen ist es auf diese Weise möglich, alle Beispiele auf dem eigenen Computer nachzustellen.

Welcher Anwendungszweck ist vorgesehen?

Die nächste Frage, die für die Auswahl einer passenden Linux-Distribution von Bedeutung ist, ist der vorgesehene Anwendungszweck. Es gibt Linux-Distributionen, die sich in erster Linie an private Anwender ohne spezifische Anforderungen richten. Diese bieten allgemeine Funktionen an – vom Webbrowser bis hin zu Multimedia-Anwendungen. Auch hierfür ist Ubuntu wieder ein gutes Beispiel. Hierbei handelt es sich um eine alltagstaugliche Distribution, die die Wünsche der meisten Privatanwender erfüllt.

Darüber hinaus gibt es zahlreiche Linux-Distributionen, die für einen ganz speziellen Anwendungszweck vorgesehen sind. Von großer Bedeutung sind hierbei beispielsweise wissenschaftliche Distributionen. Wenn Sie wissenschaftliche Forschungsarbeiten durchführen, bietet beispielsweise eine Version wie Fedora Scientific oder Poseidon Linux viele Vorteile. Es gibt sogar Distributionen für ganz spezielle Wissenschaftszweige – Bio Linux für Bioinformatiker, Fedora Astronomy Suite für Astronomen oder CAELinux für Ingenieure. Dabei handelt es sich jedoch nur um eine kleine Auswahl an wissenschaftlichen Linux-Distributionen. Es gibt noch weitere Möglichkeiten für die unterschiedlichsten Fachbereiche. Die Spezialisierung der Distribution muss außerdem nicht immer im wissenschaftlichen Bereich liegen. Es gibt beispielsweise auch spezielle Gaming-Versionen. Beispiele hierfür sind Steam OS und batocera.linux.

Etwas allgemeiner gesprochen ist es wichtig, darauf zu achten, welche Programme die jeweilige Distribution beinhaltet. Jede Distribution enthält ein sogenanntes Repository mit verschiedenen Anwenderprogrammen. Die hier enthaltene Software lässt sich schnell und ein-

fach auf dem Rechner installieren. Andere Programme hinzuzufügen, ist zwar nicht gänzlich unmöglich. Allerdings stellt dies keine einfache Aufgabe dar. Die enthaltenen Anwendungen geben daher den Verwendungszweck der jeweiligen Distribution vor. Daher lohnt es sich immer, einen Blick in das *Repository* zu werfen und sich darüber zu informieren, welche Angebote dieses enthält.

Die grafische Oberfläche

Ein weiterer wichtiger Aspekt betrifft die grafische Oberfläche, die das Betriebssystem dem Anwender bereitstellt. Hierfür gibt es zahlreiche verschiedene Möglichkeiten, die einen wichtigen Einfluss auf die Verwendung des Betriebssystems haben.

Die frühesten Linux-Distributionen enthielten überhaupt keine grafische Oberfläche. Die Steuerung des Systems war lediglich mit schriftlichen Befehlen möglich. Auch heute gibt es noch einige Distributionen, die für diese Art der Anwendung vorgesehen sind. Das stellt jedoch insbesondere für Anfänger eine erhebliche Herausforderung dar. Daher entscheiden sich die meisten Anwender, die Linux zum ersten Mal verwenden, für eine Distribution, die auf einer grafischen Oberfläche beruht.

Doch auch hierbei gibt es einige Unterschiede. Von großer Bedeutung ist hierbei beispielsweise die GNOME Desktop-Umgebung. Deren grafische Benutzeroberfläche findet in vielen beliebten Distributionen Verwendung – beispielsweise bei Ubuntu oder Fedora. Diese ermöglicht zwar eine einfache und effiziente Steuerung. Allerdings weisen die Strukturen hierbei erhebliche Unterschiede zum Betriebssystem Windows auf. Das sorgt bei vielen Anwendern, die zuvor einen Windows-Rechner verwendeten, für eine erhebliche Umstellung.

Eine Alternative dazu stellt die Cinnamon Desktop-Umgebung dar. Diese ist in ihrem Aufbau deutlich ähnlicher zur grafischen Oberfläche von Windows. Das stellt einen wichtigen Grund für ihre Beliebtheit dar. Das bekannteste Beispiel für Distributionen mit Cinnamon

Desktop-Umgebung ist Linux Mint. Das liegt daran, dass die Mint-Entwickler auch für die Gestaltung von Cinnamon verantwortlich sind. Mittlerweile verwenden jedoch noch viele weitere Distributionen diese Umgebung.

Darüber hinaus gibt es noch einige weitere Möglichkeiten, die jedoch seltener zum Einsatz kommen. Beispiele hierfür sind MATE und Xfce. Dennoch tragen auch diese Desktop-Umgebungen dazu bei, dass in dieser Hinsicht zahlreiche Auswahlmöglichkeiten bestehen. Um eine Entscheidung zu treffen, ist es sinnvoll, die verschiedenen Möglichkeiten genau zu betrachten und zu überprüfen, welche Ihnen am stärksten zusagt.

Betriebssystem für Anfänger oder Fortgeschrittene?

Eine weitere wichtige Frage besteht darin, ob Sie ein System für Anfänger oder für fortgeschrittene Anwender suchen. Zahlreiche Distributionen sind darauf ausgelegt, dem Anwender die Nutzung so leicht wie möglich zu machen. Die Installation lässt sich dabei auch ohne Vorkenntnisse durchführen und die Nutzung stellt ebenfalls kein Problem dar.

Es gibt aber auch Distributionen, bei denen dies nicht ganz so einfach ist. Dabei muss der Anwender bei der Installation und der Konfiguration des Systems viele Aufgaben selbst erledigen. Auch für die Wartung des Systems ist immer wieder ein Eingriff des Nutzers notwendig. Das bringt einen deutlich höheren Zeitaufwand mit sich und außerdem sind die erforderlichen Kenntnisse hierbei ungleich höher. Der Vorteil besteht jedoch darin, dass es auf diese Weise möglich ist, das System zu individualisieren.

Passende Systeme für Anfänger sind beispielsweise Ubuntu, Open SUSE, Fedora oder Linux Mint. Distributionen, die hingegen nur für fortgeschrittene Anwender zu empfehlen sind, sind Debian, Arch Linux oder Gentoo.

Update-Intervalle beachten

Ein weiteres Detail, das Einfluss auf die Auswahl haben kann, sind die Update-Intervalle der entsprechenden Version. Es wurde bereits gesagt, dass die Programme, die sich auf dem System installieren lassen, stets von der verwendeten Distribution abhängen. Darüber hinaus spielt jedoch auch die Version eine wichtige Rolle. Um eine neue Software verfügbar zu machen, ist es notwendig, die Linux-Distribution entsprechend anzupassen.

Manche Distributionen werden nur sehr selten aktualisiert. Das hat zur Folge, dass es nicht möglich ist, die neueste Software zu nutzen. Andere Versionen werden praktisch ständig überarbeitet – so häufig, dass es hierbei überhaupt keine klar definierten Versionen gibt. Das wäre im Prinzip zwar positiv, allerdings führt diese Form der Aktualisierung auch zu Problemen. Sie ist sehr fehlerbelastet und kann zu ernsthaften Stabilitätsproblemen führen. Die Distribution Arch Linux geht beispielsweise nach diesem Modell vor: Obwohl sie stets die aktuelle Software anbietet, hat sie sich dennoch einen zweifelhaften Ruf erarbeitet, weil das System häufig sehr instabil ist.

Daher ist es wichtig, einen Kompromiss zu finden. Jeder Anwender muss selbst entscheiden, ob er mehr Wert auf die Aktualität oder auf die Stabilität legt. Distributionen, die hierbei eine gute Mischung gefunden haben und daher bei vielen Nutzern sehr beliebt sind, sind beispielsweise Ubuntu, Linux Mint und OpenSUSE.

3.5 Übungsaufgaben

Finden Sie heraus, wo Sie sich über die Häufigkeit der Nutzung der einzelnen Distributionen informieren können. Ermitteln Sie, welche Distribution im Moment die beliebteste ist und finden Sie heraus, was diese auszeichnet.

Lösung:

Um die Beliebtheit der einzelnen Distributionen zu ermitteln, ist die Seite distrowatch.com sehr hilfreich. Im Moment (Stand: Oktober 2020) wird hier MX Linux als die am häufigsten verwendete Distribution angegeben. Dabei handelt es sich um ein Debian-Derivat. MX Linux nutzt standardmäßig die Xfce-Desktopumgebung, wobei es auch möglich ist, weitere Alternativen hinzuzufügen.

Kapitel 4

Vorüberlegungen und Vorbereitungsmaßnahmen

Bevor wir im nächsten Kapitel damit beginnen, verschiedene Linux-Distributionen zu installieren, ist es wichtig, noch einige Vorüberlegungen durchzuführen. Die Entscheidungen, die hierbei getroffen werden, haben später erhebliche Auswirkungen auf die Nutzung des Computers. Daher ist es sehr wichtig, sich die entsprechenden Aspekte genau zu überlegen.

Darüber hinaus werden wir bereits mit den ersten praktischen Vorbereitungsmaßnahmen beginnen. Das bedeutet, dass Sie hierbei selbst Hand anlegen müssen, um Ihren Rechner entsprechend einzurichten. Daher ist es wichtig, dass bei der Bearbeitung dieses Kapitels ein Computer zur Verfügung steht, an dem Sie die entsprechenden Maßnahmen durchführen können.

4.1 Ausschließlich Linux verwenden oder ein Multi-Boot-System implementieren?

Es bestehen unterschiedliche Formen, um Linux zu nutzen. Zahlreiche Anwender entscheiden sich dazu, ausschließlich dieses Betriebssystem zu verwenden. Eine weitere Alternative besteht darin, das bisherige Betriebssystem auch weiterhin zu nutzen. Es ist möglich, Linux neben einer anderen OS-Software zu installieren. In diesem Fall spricht man von einem Multi-Boot-System oder – in dem spezifischen Anwendungsfall, in dem es sich genau um zwei verschiedene Betriebssysteme handelt – von einem Dual-Boot-System. Hinzu kommen noch weitere Möglichkeiten, bei denen es überhaupt nicht notwendig ist, Linux zu installieren. Dabei handelt es sich um das Windows Subsystem for Linux (WSL) und um die Einrichtung einer virtuellen Maschine (VM). Daher ist es wichtig, sich zu entscheiden, welche dieser Optionen Sie umsetzen, um

Linux zu nutzen. Dieser Abschnitt stellt zunächst die Möglichkeiten für die Umsetzung ohne eine Installation vor. Anschließend geht er darauf ein, was ein Multi-Boot-System auszeichnet und welche Vor- und Nachteile dieses bietet.

Windows Subsystem for Linux

Das Betriebssystem Linux wird für professionelle Programmierer immer wichtiger. Um dieses Klientel nicht zu verlieren, hat Windows seine Funktionen in gewissem Umfang auch an Linux angepasst. Dabei ist in erster Linie das Windows Subsystem for Linux (WSL) zu nennen. Dabei handelt es sich um ein Programm, das die System Calls eines Linux-Programms in System Calls für das Windows-Betriebssystem umwandelt. Diese Software verwendet keine Bestandteile des Linux-Kernels. Sie ermöglicht lediglich die Ausführung von Linux-Programmen unter Windows. Wenn Sie selbst programmieren und dabei Programme für Linux erstellen, bietet diese Software Ihnen die Möglichkeit, diese unter Windows auszuprobieren. Allerdings ist dies nur für gelegentliche Anwendungen zu empfehlen. Zum einen sind hierbei nicht alle Funktionen verfügbar. Insbesondere die Ausführung von Programmen mit grafischer Oberfläche ist nur unter erheblichen Anpassungen möglich. Zum anderen ist die Performance des Systems nicht allzu gut. Wenn Sie regelmäßig mit Linux-Programmen arbeiten, ist es daher empfehlenswert, das System fest auf dem Rechner zu installieren.

Eine virtuelle Maschine einrichten

Eine ähnliche Option stellt es dar, eine virtuelle Maschine zu verwenden. Auch hierbei handelt es sich um eine Software, die es möglich macht, Linux zu nutzen, ohne das Betriebssystem zu installieren. Sie können eine virtuelle Maschine sowohl auf Windows- als auch auf macOS-Rechnern erzeugen. Diese Software simuliert ein komplettes Linux-System mit allen Funktionen. Das bedeutet, dass Sie auf diese Weise Linux in vollem Umfang nutzen können, ohne das System zu installieren. Doch auch hierbei sein angemerkt, dass diese Lösung nur für

gelegentliche Anwender zu empfehlen ist. Der Grund dafür besteht darin, dass die Performance der virtuellen Maschine deutlich schlechter als bei einem fest installierten System ist.

Linux fest installieren

Wenn Sie einen Computer, der mit nur einem Betriebssystem arbeitet, einschalten, startet dieser direkt die entsprechende Software. Hierfür ist ein Bootloader verantwortlich, der nach dem Starten den entsprechenden Prozess initialisiert. Es ist jedoch auch möglich, ein weiteres System auf dem Rechner zu installieren. In diesem Fall startet der Bootloader nicht automatisch nach dem Anschalten das Betriebssystem. Stattdessen fragt er den Anwender, welche der verfügbaren Alternativen er starten will. Das bedeutet, dass er bei jedem Startvorgang zwischen den installierten Betriebssystemen auswählen kann. Normalerweise ist der Bootloader so konfiguriert, dass er ohne den Eingriff des Anwenders nach einiger Zeit ein Default-System auswählt. Dieses startet er nach einer kurzen Wartezeit dann automatisch.

Viele Anwender, die bereits einen Computer mit einem anderen Betriebssystem nutzen und ihre ersten Erfahrungen mit Linux sammeln, entscheiden sich für ein Dual-Boot-System. Es ist beispielsweise möglich, Linux neben Windows zu installieren. Doch Sie können das Betriebssystem auch gemeinsam mit macOS verwenden. Auf diese Weise ist es möglich, das gewohnte System bei Bedarf auch weiterhin zu nutzen. Außerdem bleiben alle bereits installierten Programme erhalten. Dennoch ist es möglich, auf dem gleichen Rechner auch Linux zu verwenden. Das erweitert die Anwendungsmöglichkeiten deutlich.

Wichtig bei der Einrichtung eines Multi-Boot-Systems ist es, eine klare Abgrenzung zwischen den verwendeten Speicherbereichen zu erzeugen. Das bezieht sich in diesem Fall nicht auf den Arbeitsspeicher. Das Betriebssystem, das zum entsprechenden Zeitpunkt aktiviert ist, hat stets Zugriff auf den gesamten Arbeitsspeicher. Die Trennung bezieht sich auf den dauerhaften Datenspeicher – also auf die Festplatte oder auf ein SSD-Laufwerk. Hierbei ist es wichtig, dass jedes Betriebssystem

tem nur auf seinen eigenen Speicherbereich zugreifen kann. Das verhindert Konflikte bei der Nutzung. Sollte der Computer über mehrere Festplatten oder SSD-Laufwerke verfügen, stellt die Trennung kein Problem dar. In diesem Fall ist es möglich, jedem System ein eigenes Laufwerk zuzuweisen. Sollte – wie bei den meisten Computern üblich – nur eine Festplatte vorhanden sein, stellt das allerdings auch kein Problem dar. In diesem Fall ist es notwendig, die Festplatte zu partitionieren. Das bedeutet, dass man sie in feste Bereiche für die einzelnen Betriebssysteme einteilt. Wie Sie hierbei vorgehen, erfahren Sie in Kapitel 4.2.

Um die Entscheidung zu treffen, ob Sie ausschließlich Linux verwenden oder ein Multi-Boot-System einrichten, ist es sinnvoll, sich mit den Vor- und Nachteilen dieser beiden Alternativen zu befassen. Einer der wesentlichen Vorteile wurde bereits angesprochen. Die Verwendung eines Multi-Boot-Systems erlaubt es, alle Programme, die bisher auf dem Rechner installiert sind, auch weiterhin zu nutzen. Das stellt insbesondere dann einen großen Vorteil dar, wenn der entsprechende Computer bereits in Verwendung ist. Auf diese Weise wird der Aufwand für die Umstellung reduziert. Doch auch bei neuen Systemen kann dies hilfreich sein. Es gibt einige Programme, die nur in Versionen für Windows oder macOS verfügbar sind, nicht jedoch für Linux. Wenn Sie eine solche Software beispielsweise für Ihre Arbeit benötigen, wäre es nicht sinnvoll, ausschließlich Linux zu verwenden, da Sie sie sonst nicht mehr nutzen können. Ein weiterer Vorteil besteht darin, dass Sie auf diese Weise die vertraute Umgebung auch weiterhin nutzen können. Sollten zu Beginn Probleme bei der Verwendung von Linux auftreten, müssen Sie lediglich den Rechner neu starten, um wieder zum bisherigen System zu wechseln.

Dennoch bietet es auch einige Vorteile, ausschließlich mit Linux zu arbeiten – insbesondere wenn es sich dabei um einen neuen Computer handelt, auf dem noch keine Programme installiert sind. Beispielsweise können Sie auf diese Weise die Lizenzgebühren für ein proprietäres Betriebssystem sparen. Außerdem reduzieren Sie dadurch den Speicherbedarf auf der Festplatte. Das Betriebssystem hat einen hohen Speicherbedarf. Sollten Sie zwei Systeme installieren,

kann es insbesondere bei kleineren Festplatten zu Platzproblemen kommen. Darüber hinaus wird auf diese Weise das Betriebssystem stets direkt geladen. Manche Anwender empfinden es als lästig, wenn sie bei jedem Startvorgang zunächst das Betriebssystem auswählen müssen.

4.2 Die Partitionen der Festplatte vorbereiten

Kapitel 4.1 hat aufgezeigt, dass es möglich ist, entweder Linux als ausschließliches Betriebssystem zu installieren oder ein Dual-Boot-System einzurichten. Die folgenden Abschnitte stellen vor, wie Sie die Festplatte auf die Partitionierung für ein Dual-Boot-System vorbereiten. Das bedeutet, dass Sie sie in mehrere unabhängige Abschnitte einteilen. Auf diese Weise erhält jedes Betriebssystem seinen eigenen Bereich.

Diese Aufgabe ist allerdings nur dann notwendig, wenn Sie sich dazu entschieden haben, mehrere Betriebssysteme zu verwenden. Sollten Sie ausschließlich mit Linux arbeiten wollen, ist dies nicht notwendig. In diesem Fall kann Linux die komplette Festplatte beanspruchen und es ist keine weitere Unterteilung notwendig.

In diesem Kapitel erstellen wir noch keine Partitionen. Dieser Schritt erfolgt erst bei der Installation des neuen Betriebssystems. Allerdings müssen wir Platz dafür schaffen. Die Vorgehensweise ist dabei genau die gleiche – unabhängig davon, ob Sie ein Dual-Boot-System mit genau zwei Systemen oder ein Multi-Boot-System mit noch mehr Alternativen einrichten möchten.

Was ist eine Partition?

Als Partition wird eine bestimmte Sektion eines physischen Datenträgers bezeichnet, der klar von den übrigen Bereichen abgetrennt ist. Der Computer behandelt sie wie ein eigenes Laufwerk – obwohl sich alle Partitionen auf dem gleichen physischen Datenträger befinden.

Die Verwendung von Partitionen macht es möglich, jedem Betriebssystem einen separaten Bereich zuzuordnen und stellt daher eine wichtige Voraussetzung für die Einrichtung eines Multi-Boot-Systems dar. Es ist sogar möglich, innerhalb eines Betriebssystems mehrere Partitionen zu erzeugen. Das ist beispielsweise bei sehr umfangreichen Festplatten sinnvoll, um die Organisation der Daten zu erleichtern. Darüber hinaus trägt dies zu einer höheren Datensicherheit bei. Jede *Partition* verwendet ihr eigenes Dateisystem. Sollte es hierbei zu einem Schaden kommen, ist es nur notwendig, das Dateisystem der betroffenen Partition zu formatieren. Die übrigen Bereiche sind daher nicht betroffen.

Bestehende Partition unter Windows 10 verkleinern

Da es sich bei Windows 10 um das am weitesten verbreitete PC-Betriebssystem handelt, stellen wir zunächst vor, auf welche Weise es hierbei möglich ist, eine weitere Partition für Linux hinzuzufügen. Bevor wir damit beginnen, sei jedoch darauf hingewiesen, dass es unbedingt ratsam ist, vor diesem Eingriff eine Sicherungskopie des gesamten Systems zu erstellen. Außerdem ist es empfehlenswert, eine Installationsdatei für Windows zu erstellen. Der Grund dafür besteht darin, dass es bei einem Fehler in diesem Bereich zu schwerwiegenden Folgen kommen kann. Nicht nur ein kompletter Datenverlust ist dabei möglich. Darüber hinaus kann dabei das gesamte Betriebssystem beschädigt werden, sodass es sich nicht mehr starten lässt. Dabei handelt es sich jedoch nur um ein Worst-Case-Szenario. Wenn Sie alle Anweisungen in diesem Kapitel genau befolgen, sollten keine Probleme auftreten.

Für diese Aufgabe ist es notwendig, die Systemsteuerung aufzurufen. Diese ist nicht zu verwechseln mit dem Menü für die Einstellungen, das Sie im Start-Menü über das Zahnradsymbol öffnen können. Dieses enthält nur die grundlegenden Möglichkeiten, um die Funktionen des Betriebssystems festzulegen. Die Systemsteuerung geht hierbei deutlich weiter. Im Gegensatz zu früheren Windows-Versionen ist hierfür jedoch keine Schaltfläche mehr vorhanden, zu der man sich

durchklicken könnte. Daher ist es notwendig, diesen Begriff in das Suchfeld unten in der Taskleiste einzugeben, wie dies die folgende Abbildung zeigt.

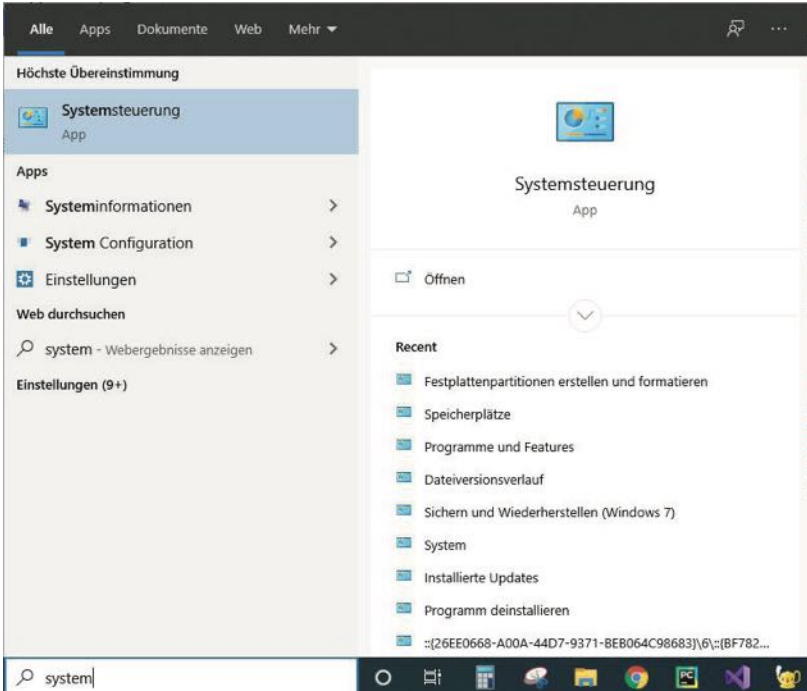


Abb. 4.1 Der Aufruf der Systemsteuerung

Bereits nach der Eingabe der ersten Buchstaben erscheint im oberen Bereich des Fensters die entsprechende Auswahlmöglichkeit. Nach einem Klick auf den entsprechenden Begriff sollte sich das Fenster öffnen, das in der folgenden Abbildung zu sehen ist.

4 Vorüberlegungen und Vorbereitungsmaßnahmen

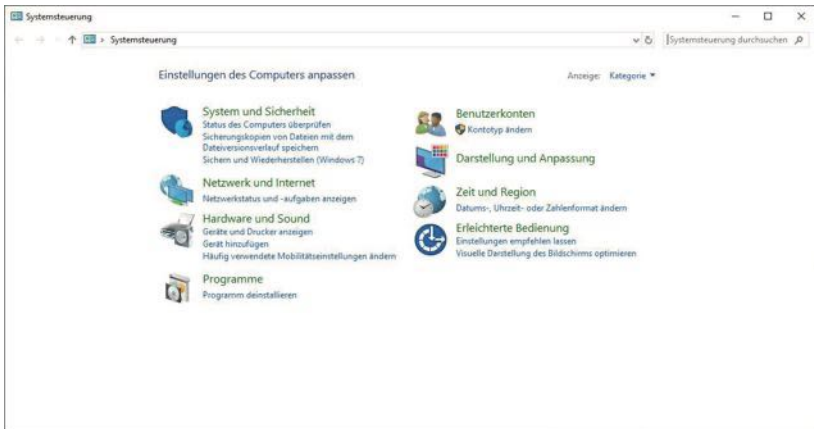


Abb. 4.2 Die Auswahlmöglichkeiten in der Systemsteuerung

Hier ist es nun notwendig, auf „System und Sicherheit“ zu klicken. Danach müssen Sie etwas nach unten scrollen und unter der Überschrift „Verwaltung“ den Begriff „Festplattenpartitionen erstellen und formatieren“ auswählen. Das zeigt die nächste Abbildung.



Abb. 4.3 Die Auswahl für die Verwaltung der Partitionen

Daraufhin öffnet sich ein neues Fenster. Hier werden die Partitionen angezeigt, die bereits auf der Festplatte vorhanden sind. Deren Zusammenstellung kann sich jedoch - je nach System - leicht unterscheiden. In

der folgenden Abbildung ist zu sehen, dass in diesem Beispiel vier Blöcke vorhanden sind. Die ersten beiden Partitionen sind für den Startvorgang erforderlich. Bei der letzten der beiden Partitionen handelt es sich um eine sogenannte Wiederherstellungspartition. Diese dient dazu, das System bei einem Fehler wiederherzustellen. Darüber hinaus ist das Laufwerk C vorhanden, das eine eigene Partition darstellt. Dieses Laufwerk kennen Sie wahrscheinlich von der Nutzung des PCs. Hierbei handelt es sich um den Ursprung des Dateisystems auf der Festplatte, das Sie unter Windows nutzen können. Alle Dateien und Programme – darunter auch das komplette Betriebssystem – sind darin enthalten.

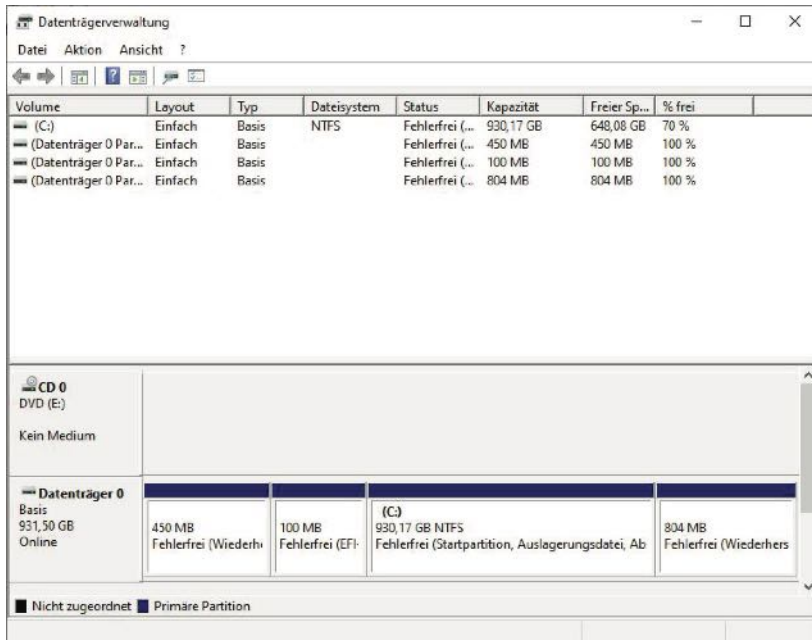


Abb. 4.4 Die standardmäßig vorhandenen Partitionen

An dieser Aufstellung wird deutlich, dass die bereits vorhandenen Partitionen in der Regel bereits die komplette Festplatte einnehmen. Darüber hinaus zeigt die Abbildung, dass das Laufwerk C hierbei mit großem Abstand den größten Bereich belegt. Die anderen Partitionen beanspruchen jeweils nur einige Hundert MB.

Da kein freier Raum mehr auf der Festplatte vorhanden ist, der keiner Partition zugeordnet ist, ist es notwendig, eine der vorhandenen Partitionen zu verkleinern. Hierfür bietet sich das Laufwerk C an, da die übrigen Partitionen zum einen wichtig für die Funktionsweise des Computers sind und zum anderen nicht den ausreichenden Platz bieten. Voraussetzung hierfür ist es selbstverständlich, dass im Laufwerk C noch ausreichend Speicherplatz verfügbar ist.

Hierbei ist es nun notwendig, eine wichtige Entscheidung zu treffen: Wie viel Platz soll das Linux-Betriebssystem einnehmen? Das hängt von vielen Faktoren ab. Zum einen ist es wichtig, auf die Mindestanforderungen der gewünschten Distribution zu achten. Hierbei können erhebliche Unterschiede auftreten. Darüber hinaus spielt die Rechnerarchitektur eine wichtige Rolle. Bei Ubuntu 20.04 werden etwa für die AMD64-Architektur, die bei den meisten PCs mit 64 Bit zum Einsatz kommt, 10 GB als Minimum angegeben. Wenn Sie das System ohne grafische Desktopoberfläche installieren möchten, sind sogar nur 2 GB erforderlich. Dennoch ist es nicht empfehlenswert, sich an diesen Mindestwerten zu orientieren. Als empfohlener Speicherplatz sind beispielsweise 25 GB angegeben. Das ist sinnvoll, um eine reibungslose Funktionsweise sicherzustellen. Wenn Sie eine andere Distribution installieren möchten, sollten Sie hierbei ebenfalls einen kurzen Blick auf die jeweiligen Mindestanforderungen werfen, um diesen Wert auf keinen Fall zu unterschreiten.

Da mittlerweile die meisten Festplatten sehr groß sind, ist es jedoch normalerweise nicht notwendig, sich an diesen Mindestwerten zu orientieren. Viel wichtiger ist die Frage, welches Betriebssystem Sie vorwiegend nutzen möchten. Wenn Sie am Computer arbeiten, fallen im Laufe der Zeit große Mengen an Daten an. Wenn Sie auch weiterhin in erster Linie Ihr bisheriges System verwenden möchten, ist es sinnvoll, diesem auch weiterhin den größten Teil der Festplatte zuzuweisen. Wollen Sie hingegen fortan hauptsächlich Linux nutzen, ist es empfehlenswert, einen entsprechend großen Speicherbereich dafür vorzuhalten.

Nachdem diese Entscheidung getroffen ist, ist es erforderlich, das Laufwerk C um den entsprechenden Wert zu verkleinern. Dazu ist es not-

wendig, es mit der rechten Maustaste anzuklicken und im Kontextmenü daraufhin „Volume verkleinern“ auszuwählen. Das ist in der folgenden Abbildung zu sehen.

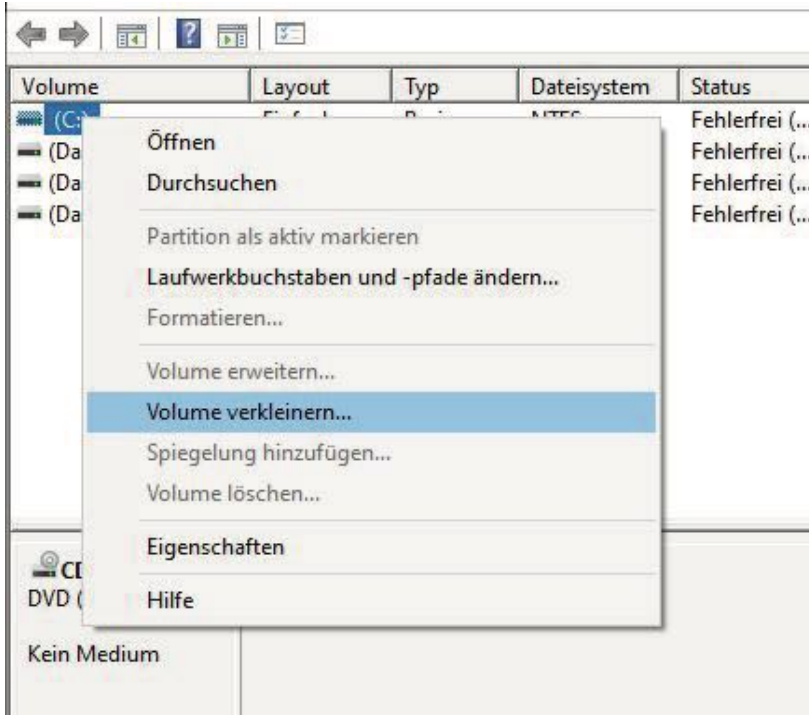


Abb. 4.5 Eine vorhandene Partition verkleinern

Nach einer gewissen Wartezeit erscheint das Fenster, das in Abbildung 6 zu sehen ist. Hier ist es nun möglich vorzugeben, um welchen Bereich die bestehende Partition verkleinert werden soll. An dieser Stelle müssen Sie den gewünschten Wert eingeben.

Allerdings ist hierbei in der zweiten Zeile ein Höchstwert für die Verkleinerung angegeben. Dabei tritt häufig das Problem auf, dass dieser recht klein ist, obwohl eigentlich noch viel freier Speicherplatz zur Verfügung steht. Das liegt daran, dass häufig leere Bereiche zwischen den verschiedenen Inhalten auftreten. Um die Partition zu verkleinern,

kann jedoch nur der freie Platz entfernt werden, der sich am Ende befindet. Sollte sich beispielsweise eine Datei ganz am Ende der bestehenden Partition befinden, lässt sich diese überhaupt nicht verkleinern – selbst wenn diese sonst noch fast leer ist. Eine Möglichkeit, um dieses Problem zu beheben, besteht darin, über die Windows-Suchfunktion die folgende Anwendung aufzurufen: „Laufwerke defragmentieren und optimieren“. Durch eine Defragmentierung der Festplatte lässt sich das Problem häufig beheben. Eine andere Alternative besteht darin, ein externes Partitionierungs-Tool zu verwenden. Eine Möglichkeit hierfür ist der MiniTool Partition Wizard, der unter folgendem Link zum Download bereitsteht: <https://www.partitionwizard.com/>. Hierbei ist auch eine Gratis-Version verfügbar. Mit solchen Tools ist es häufig einfacher, den benötigten Platz freizugeben. Sollten Sie auch damit nicht ans Ziel kommen, ist es leider notwendig, die Festplatte durchzugehen und alle nicht mehr benötigten Dateien zu löschen.

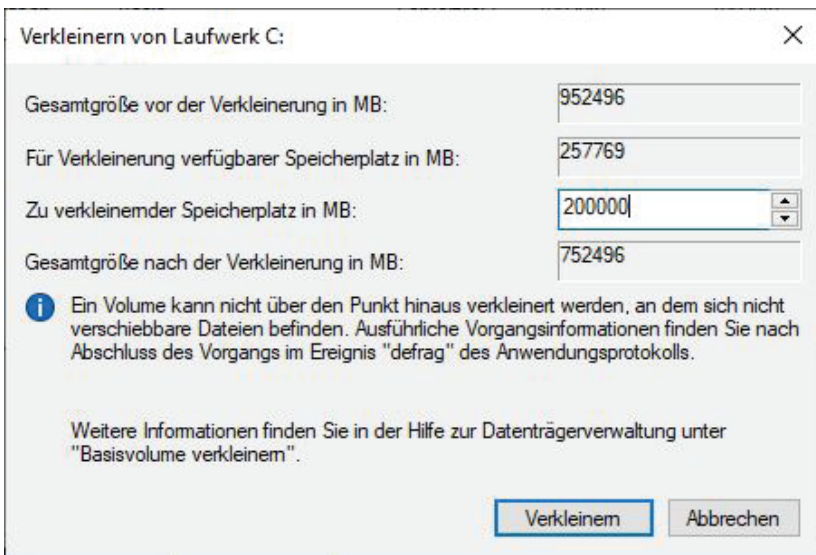


Abb. 4.6 In diesem Beispiel wird das Laufwerk C um 200.000 MB verkleinert

Sobald der benötigte Platz verfügbar ist, können Sie den Button „Verkleinern“ anklicken. Daraufhin beginnt der Rechner damit, die Fest-

platte neu aufzuteilen. Abbildung 4.7 zeigt, dass nun ein neuer Bereich erscheint. Dieser wird bislang als „Nicht zugeordnet“ beschrieben. Wir können ihn in diesem Zustand belassen. Die Zuordnung wird dann bei der Installation des Linux-Betriebssystems erfolgen.

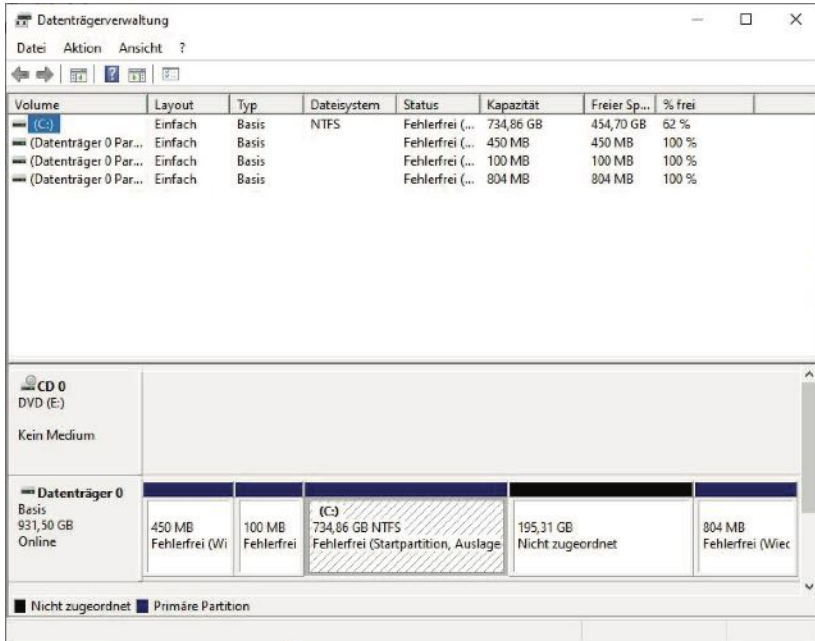


Abb. 4.7 Die Aufteilung der Festplatte nach dem Erstellen einer neuen Partition

Partitionen in anderen Betriebssystemen einrichten

Bei Windows 10 handelt es sich im Moment zwar um das am weitesten verbreitete Betriebssystem. Einige Leser verwenden aber sicherlich auch eine andere Alternative. Daher soll das Vorgehen hierbei ebenfalls kurz skizziert werden. Sollten diese Anweisungen nicht ausreichen, um die Partition einzurichten, ist es empfehlenswert, eine kurze Internetrecherche durchzuführen. Dabei sollte es problemlos möglich sein, eine detaillierte Anweisung für alle gängigen Betriebssysteme zu finden.

Wenn Sie eine der früheren Windows-Versionen verwenden, ist die Vorgehensweise recht ähnlich. Hier ist es sogar in der Regel etwas einfacher, die Systemsteuerung zu finden, da hierfür bei den früheren Versionen eine eigene Schaltfläche vorhanden war. Alternativ dazu ist es immer auch möglich, die Windows-Suchfunktion zu verwenden – auch direkt mit dem Suchbegriff „Festplattenpartitionen erstellen“. Das Fenster, das daraufhin erscheint, ist sehr ähnlich wie in Windows 10. Beim Erstellen der Partition sollten nur minimale Unterschiede auftreten.

Auf einem Mac ist es hingegen notwendig, das Festplattendienstprogramm aufzurufen. Hier ist es dann möglich, eine bestehende Partition auszuwählen und durch einen Klick auf „Partitionieren“ neu aufzuteilen. Eine etwas detailliertere Anweisung ist direkt beim Apple-Support verfügbar:

<https://support.apple.com/de-de/guide/disk-utility/dskutl14027/mac>

4.3 Weitere Vorbereitungsmaßnahmen

In Kapitel 1 wurde bereits dargestellt, dass es möglich ist, dass ein Betriebssystem im BIOS- oder im UEFI-Modus arbeitet. Beim Booten des Computers kann dieser jedoch zwischen diesen beiden Alternativen auswählen. Das hat einen wichtigen Einfluss auf den Installationsprozess. Die folgenden Schritte sind jedoch nur für Windows-Rechner von Bedeutung. Apple-Computer verwenden – von sehr alten Modellen abgesehen – immer den UEFI-Modus.

Zwar sind Windows 8 und Windows 10 in der Regel im UEFI-Modus installiert, doch gibt es hierbei auch Ausnahmen. Daher ist es wichtig, dies nachzuprüfen. Wenn Sie bereits in Kapitel 1.5 die erste Live-Version von Linux getestet haben, haben sie wahrscheinlich bereits festgestellt, ob Ihr System im UEFI- oder im BIOS-Modus abläuft. Sollte dies noch nicht bekannt sein, ist es nun wichtig, dies herauszufinden. Hierzu müssen Sie in die Windows-Suchleiste „Systeminformationen“ eingeben und daraufhin die entsprechende App auswählen. Unter dem Eintrag „BIOS Mode“ sehen Sie nun, welchen Modus Ihr System verwendet. Abbildung 14 zeigt den entsprechenden Eintrag, wenn Windows im UEFI-Mode abläuft.

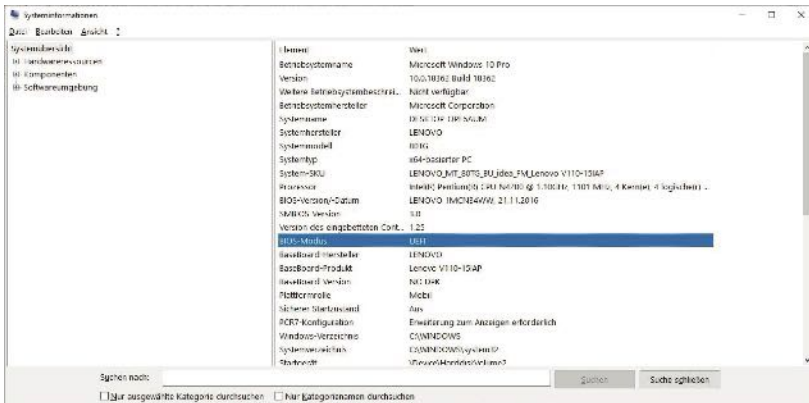


Abb. 4.8 Die Überprüfung, ob Windows im UEFI-Modus abläuft.

Wenn Sie ein Dual-Boot-System installieren möchten, ist es wichtig, dass Sie die Live-Version hierfür im gleichen Modus starten wie Ihr Windows-System. Sonst kann es zu Problemen beim Booten kommen. Aus diesem Grund ist es wichtig, die entsprechenden Modus beim Booten vorzugeben. Dazu müssen Sie wieder das Boot-Menü aufrufen – genau auf die gleiche Weise wie in Kapitel 1.5 beschrieben. Am Besten klicken Sie hierfür bei gedrückter Umschalttaste auf „Neu starten“. Klicken Sie dann auf auf „Problembehandlung“, „Erweiterte Optionen“, „UEFI Firmwareeinstellungen“ und abschließend auf „Neu starten“. So gelangen Sie wieder zum Boot-Menü. Hier müssen Sie nun in der Zeile „Boot Priority“ „EFI first“ auswählen. Falls Ihr Betriebssystem hingegen den BIOS-Modus installiert ist, sollte hier „Legacy first“ ausgewählt sein. Nachdem Sie die richtige Einstellung vorgenommen haben, können Sie wieder zu Windows zurückkehren. Auch hierbei ist zu beachten, dass die genauen Bezeichnungen - je nach Hersteller der Hardware - etwas abweichen können.

4.4 Übungsaufgaben

Jede Partition verwendet ein bestimmtes Dateisystem. Dieses sorgt für die Organisation der Daten. Finden Sie heraus, welche Dateisysteme Windows und Linux verwenden und worin die Unterschiede liegen.

Lösung:

Windows verwendet das Dateisystem NTFS. Linux nutzt hingegen EXT4 – zumindest in den aktuellen Versionen. Dabei gibt es folgende Unterschiede:

- ▶ NTFS ist ein proprietäres Format, während EXT4 frei ist.
- ▶ Windows kann nur mit dem NTFS-Format arbeiten. Linux kann hingegen beide Dateisysteme verwenden. Daher ist unter Linux immer auch ein Zugriff auf die Dateien möglich, die Sie unter Windows angelegt haben – vorausgesetzt, dass diese nicht verschlüsselt sind. Umgekehrt ist jedoch kein Zugriff möglich.
- ▶ EXT4 hat in zahlreichen Tests bewiesen, dass dieses Format zu einer höheren Effizienz bei Lese- und Schreibvorgängen beiträgt.

Kapitel 5

Installationsbeispiele für verschiedene Linux-Distributionen

Nachdem die Grundlagen für das Verständnis des Linux-Betriebssystems vorgestellt wurden, können wir nun mit der praktischen Nutzung beginnen. Eine Möglichkeit hierfür haben wir bereits kennengelernt: die Verwendung eines Live-Systems. Wenn man jedoch vorhat, Linux regelmäßig zu verwenden, ist es empfehlenswert, dieses System fest zu installieren. Auf diese Weise bleiben alle Daten und Einstellungen auch nach einem Neustart des Rechners verfügbar. Diese Aufgabe stellt daher stets den ersten Schritt für die dauerhafte Verwendung von Linux dar. Allerdings erfolgt bei zahlreichen Distributionen die Installation ebenfalls über ein Live-System. Das bedeutet, dass Sie zunächst eine Version des Betriebssystems erstellen, die sich auf einem Wechseldatenträger ausführen lässt. Diese können Sie dann auf Ihrem Rechner starten. Hier ist dann ein Programm verfügbar, mit dem sich das Betriebssystem fest installieren lässt.

Wie bereits in Kapitel 3 ausgeführt, ist es für die Verwendung von Linux notwendig, sich für eine Distribution zu entscheiden. Welche Sie dabei auswählen, bleibt Ihnen überlassen. In diesem Kapitel stellen wir fünf verschiedene Möglichkeiten vor: Linux Mint, OpenSUSE, Debian, Manjaro und Ubuntu. Dabei erklären wir nicht nur den jeweiligen Installationsprozess. Darüber hinaus dient dieses Kapitel nämlich dazu, sich einen besseren Überblick über die unterschiedlichen Distros zu verschaffen. Dabei ist es selbstverständlich nicht notwendig, dass Sie alle Alternativen auf Ihrem Rechner installieren. Es wäre zwar möglich, diese über ein Multi-Boot-System parallel zu betreiben. Das würde jedoch sehr viel Speicherplatz erforderlich machen. Eine weitere Option besteht darin, die einzelnen Distributionen nach der Installation wieder zu löschen und daraufhin eine andere Alternative zu installieren. Hierbei sei jedoch darauf hingewiesen, dass das zu er-

heblichen Problemen führen kann. Beim Löschen einer Distribution kommt es häufig vor, dass hierbei Dateien entfernt werden, die für den Start des Computers sehr wichtig sind. Die Folge ist, dass er nicht mehr bootet. Zwar lassen sich die defekten Dateien mit relativ geringem Aufwand wieder reparieren. Allerdings sind hierfür umfassende Kenntnisse notwendig. Deshalb sollten Sie als Anfänger darauf besser verzichten. Daher ist es empfehlenswert, sich zunächst das gesamte Kapitel durchzulesen und daraufhin eine oder maximal zwei Distributionen zu installieren. Wenn Sie etwas Zeit zur Verfügung haben, können Sie auch die entsprechenden Live-Versionen ausprobieren. Das ist empfehlenswert, um sich selbst ein Bild von den verschiedenen Distros zu verschaffen. Das hilft nicht nur bei der Entscheidungsfindung. Darüber hinaus werden Sie auf diese Weise auch mit dem Installationsprozess vertraut.

Welche der hier vorgestellten Distributionen Sie letztendlich auswählen, bleibt Ihnen überlassen. Bei der Auswahl sind Sie außerdem nicht auf die hier vorgestellten Distributionen beschränkt. Wenn Sie es wünschen, können Sie selbstverständlich auch eine ganz andere Alternative auswählen. Die Aufgaben und Übungen in diesem Buch sollten sich mit allen gängigen Distributionen umsetzen lassen.

Für die Darstellungen werden wir im weiteren Verlauf jedoch Ubuntu verwenden. Wenn Sie Wert darauf legen, dass die Anweisungen und Abbildungen genau mit Ihrem System übereinstimmen, ist es daher empfehlenswert, ebenfalls Ubuntu zu installieren.

5.1 Linux Mint

Die erste Distribution, die wir ausprobieren werden, ist Linux Mint. Diese basiert auf Ubuntu, sodass hierbei gewisse Ähnlichkeiten vorhanden sind, wie wir später noch feststellen werden. Der entscheidende Grund für die Entwicklung von Linux Mint bestand darin, dass Ubuntu die Verwendung kostenpflichtiger Software strikt von den Open-Source-Angeboten trennte. Das erschwerte die Anwendung. Häufig verwendete Programme (z. B. Adobe Flash) und Multi-Media-

Codecs (z. B. MP3) ließen sich daher auf Ubuntu nur mit fortgeschrittenen Kenntnissen nutzen. Da solche Programme und Formate jedoch sehr beliebt waren, hielt dies zahlreiche Nutzer von der Verwendung von Ubuntu ab. Daher beschloss im Jahre 2006 ein Teil der Ubuntu-Community, der mit dieser Vorgehensweise nicht einverstanden war, Linux Mint zu entwickeln. Zwar unterstützt Ubuntu mittlerweile auch proprietäre Programme deutlich besser. Dennoch konnte sich Linux Mint am Markt halten und stellt bis heute eine der beliebtesten Linux-Distributionen dar.

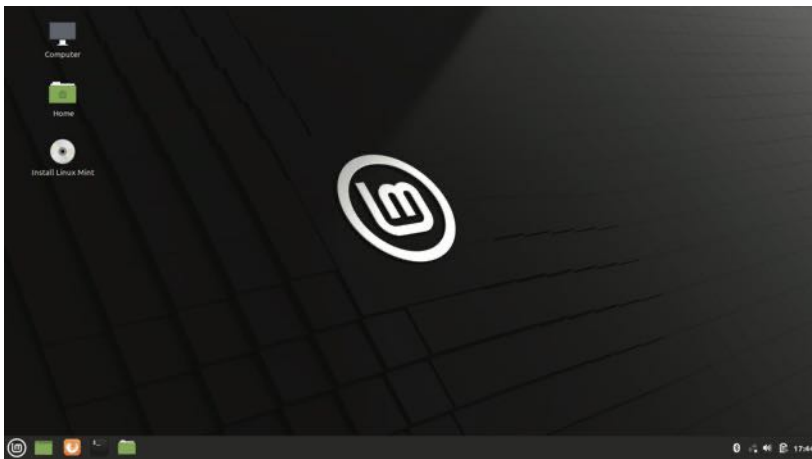
Genau wie Ubuntu ist auch Linux Mint mit verschiedenen Desktop-Umgebungen verfügbar. Das hat erhebliche Auswirkungen auf die grafische Gestaltung und die Steuerung des Systems. Wir entscheiden uns dafür, Linux Mint mit der Cinnamon-Desktop-Umgebung zu installieren. Das stellt nicht nur die häufigste Wahl bei den Anwendern dar. Darüber hinaus wurde diese speziell für Linux Mint entwickelt. Mittlerweile findet sie jedoch auch in anderen Distributionen Verwendung. Auf diese Weise lernen Sie verschiedene Möglichkeiten für die Desktop-Umgebung kennen und können die Cinnamon-Oberfläche mit anderen Alternativen vergleichen, die wir für die Installation der weiteren Distributionen nutzen werden.

Um mit der Installation zu beginnen, müssen Sie zunächst wieder die erforderliche iso-Datei herunterladen. Dafür können Sie den folgenden Link nutzen: <https://linuxmint.com/download.php>. Hier sind wiederum Links zu allen verfügbaren Desktop-Oberflächen angegeben. Wenn Sie sich ebenfalls für Cinnamon entscheiden, müssen Sie die entsprechende Schaltfläche anklicken. Sie können aber auch gerne die anderen Alternativen ausprobieren. Daraufhin gelangen Sie zu einer weiteren Seite, auf der zahlreiche Download-Links zu Servern in aller Welt angegeben sind. Dabei können Sie ein beliebiges Angebot auswählen.

Der nächste Schritt besteht dann wieder darin, einen bootfähigen USB-Stick zu gestalten. Zu diesem Zweck kommt - wie bei der Erstellung des Live-Systems in Kapitel 1.5 - der Universal USB Installer zum Einsatz. Dieser sollte daher bereits installiert sein. Selbstverständlich müssen

Sie hierbei nun Linux Mint als Distribution auswählen und die soeben heruntergeladene Datei verwenden. Danach können Sie Ihren Rechner neu starten und mit dem USB-Stick booten.

Nach dem Neustart öffnet sich der Linux-Mint-Desktop der Live-Version. Dieser ist in Abbildung 5.1 zu erkennen. Bevor wir mit der Installation beginnen, ist es empfehlenswert, sich hier eine Weile umzusehen, um sich mit dieser Distribution vertraut zu machen.



5

Abb. 5.1 Die Desktop-Oberfläche von Linux-Mint

Nun können Sie einmal auf das Mint-Symbol in der linken unteren Ecke klicken. Daraufhin öffnet sich das Menü, das in Abbildung 5.2 zu sehen ist. Dieses erinnert stark an das Start-Menü unter Windows. Das stellt einen der wesentlichen Gründe dafür dar, dass sich Anwender, die bislang mit diesem Betriebssystem gearbeitet haben, unter Linux Mint schnell heimisch fühlen.

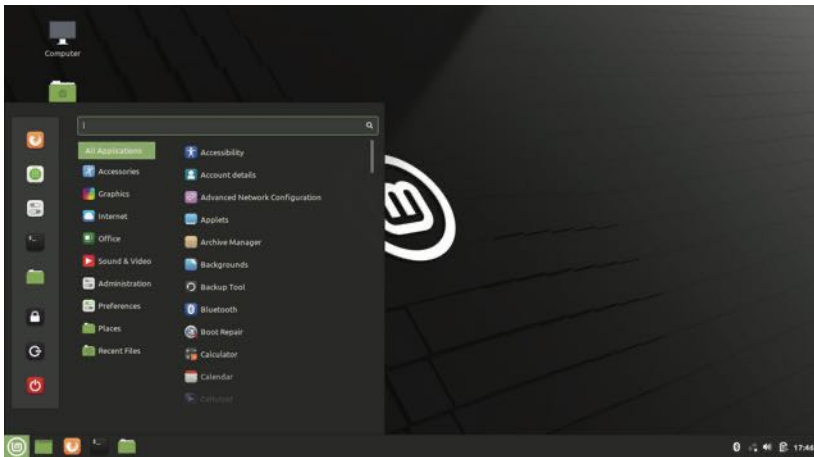


Abb. 5.2 Das „Start-Menü“ unter Linux Mint

Nun können Sie selbst die verschiedenen Anwendungen unter Linux Mint ausprobieren und schauen, ob diese Ihren Anforderungen entsprechen. Wenn Sie sich dazu entschließen, das Betriebssystem fest zu installieren, müssen Sie auf das CD-Symbol mit der Aufschrift „Install Linux Mint“ am linken Rand des Desktops-Bereichs klicken. Dieses ist in Abbildung 5.1 zu erkennen.

Daraufhin wird der Installations-Assistent gestartet. Dieser führt Sie schnell und einfach durch den Installationsprozess. Zunächst fragt er die Sprache ab, in der das Betriebssystem installiert werden soll. Danach müssen Sie die gewünschte Tastaturbelegung vorgeben. Im nächsten Schritt ist es notwendig, zu entscheiden, ob Sie die Multimedia-Codescs installieren möchten. Das führt zwar dazu, dass das Betriebssystem etwas mehr Raum einnimmt. Allerdings ist es auf diese Weise möglich, Multimedia-Inhalte in vielen verschiedenen Formaten zu nutzen. Abbildung 5.3 stellt dieses Fenster dar. Welche Auswahl Sie dabei treffen, bleibt jedoch Ihnen überlassen.

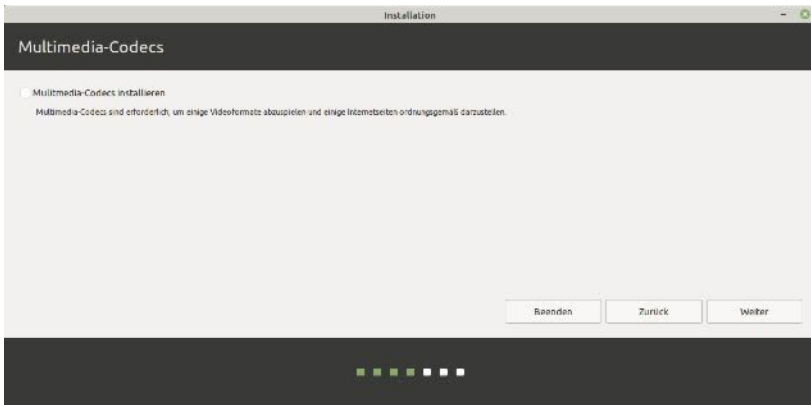


Abb. 5.3 Die Abfrage für die Installation der Multimedia-Codecs

Von großer Bedeutung ist das folgende Fenster. Dieses zeigt an, welche Betriebssysteme bereits auf dem Rechner installiert sind. Hier sollte auf jeden Fall das System erscheinen, das Sie bisher nutzen. Gelegentlich kommt es jedoch auch zu der Nachricht, die in Abbildung 5.4 zu sehen ist. In diesem Fall wurde das bisherige Betriebssystem nicht erkannt.

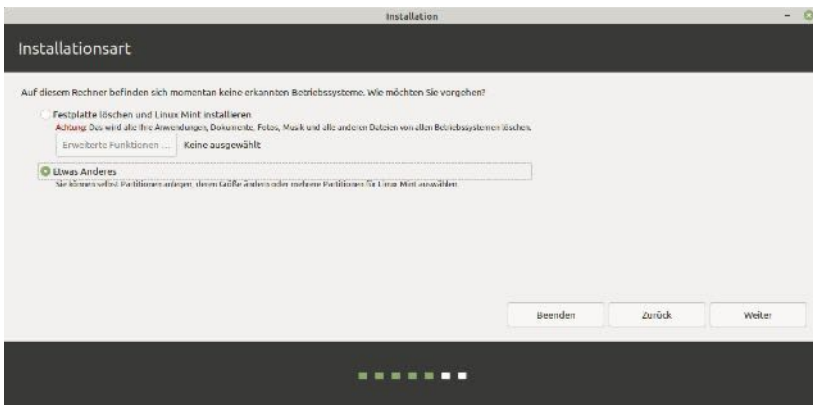


Abb. 5.4 Das bisherige Betriebssystem wurde von Linux nicht erkannt

Vorausgesetzt, dass bereits ein anderes Betriebssystem auf dem Rechner installiert ist, deutet diese Nachricht auf einen Fehler hin. Wollen

Sie ausschließlich mit Linux Mint arbeiten und Ihr bisheriges Betriebssystem dafür löschen, stellt dies kein Problem dar. In diesem Fall können Sie die erste Option auswählen. Dabei müssen Sie jedoch berücksichtigen, dass hierbei die komplette Festplatte formatiert wird. Das löscht nicht nur alle persönlichen Dateien und Anwendungen, sondern auch das komplette bisherige Betriebssystem. Wenn Sie ein Dual-Boot-System einrichten möchten, müssen Sie hingegen nochmals zu Windows zurückkehren, wenn die Nachricht aus Abbildung 18 erscheint. Der Grund hierfür besteht darin, dass der Boot-Modus der Live-Version nicht mit Ihrer Windows-Version übereinstimmt. In diesem Fall müssen Sie nochmals zu Kapitel 4.3 zurückblättern. Führen Sie die dort zusammengestellten Anweisungen nochmals aus und befolgen Sie diese genau.

Wenn alle Vorbereitungsmaßnahmen korrekt ausgeführt wurden, sollte das Fenster so aussehen, wie dies in Abbildung 5.5 zu sehen ist. Das zeigt, dass ein anderes Betriebssystem – in diesem Fall Windows – erkannt wurde.

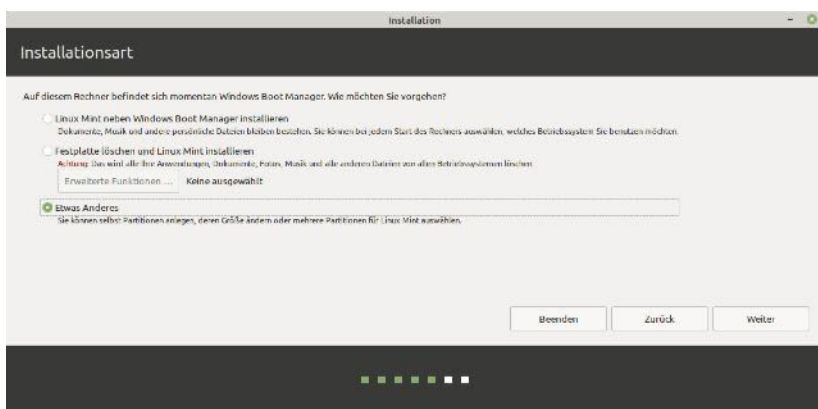


Abb. 5.5 Linux hat das bestehende Betriebssystem erkannt und bietet verschiedene Möglichkeiten für die Installation an

Nun ist es wieder notwendig, eine Auswahl zu treffen. Wenn Sie das bisherige Betriebssystem löschen und nur mit Linux arbeiten möchten, müssen Sie die zweite Option auswählen. Um ein Dual-Boot-System zu

installieren, können Sie entweder die erste oder die dritte Möglichkeit verwenden. Wählen Sie die erste Option aus, teilt der Installationsassistent den verfügbaren Platz auf der Festplatte automatisch ein. Wählen Sie hingegen die dritte Option aus, können Sie dies selbst übernehmen. Wenn Sie Wert auf eine schnelle und einfache Installation legen, ist es ratsam, den ersten Eintrag auszuwählen. In unserem Beispiel entscheiden wir uns jedoch für „Etwas Anderes“. Das gibt uns nicht nur die Möglichkeit, die Aufteilung der Festplatte selbst festzulegen. Darüber hinaus lernen Sie auf diese Weise auch die benötigten Partitionen kennen.

5

Im nächsten Schritt sollte dann ein Fenster erscheinen, das ähnlich wie in Abbildung 5.6 aussieht. Hier sind die bestehenden Partitionen der Festplatte aufgelistet. Den Vorbereitungsmaßnahmen aus Kapitel 4.2 entsprechend sollte hier nun der Bereich, den wir dabei freigegeben haben, als „Freier Speicherplatz“ erscheinen. Diesen müssen wir nun so einteilen, dass er sich für die Installation von Linux eignet.

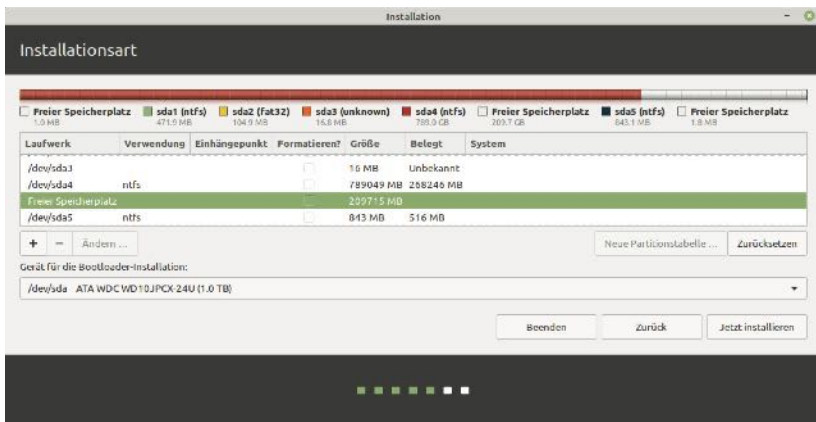


Abb. 5.6 Die bisherigen Partitionen und der freie Speicherplatz

Hierfür wählen wir drei Partitionen aus: root, swap und home. Während die Partitionen für root und swap unbedingt notwendig sind, ist die home-Partition optional. Diese erstellen wir jedoch ebenfalls. Das macht es möglich, unsere persönlichen Dateien von den Betriebssystem-Dateien zu trennen. Das führt zu einer besseren Übersichtlichkeit.

Wenn Sie auf die home-Partition verzichten, werden auch die persönlichen Daten in der root-Partition abgespeichert. Wenn Sie sich weiter vorne für die automatische Installation von Linux Mint entschieden haben, wird hierbei ebenfalls keine eigene Partition für das home-Verzeichnis erstellt. Die freie Gestaltungsmöglichkeit für die Aufteilung der Partitionen ist jedoch einer der wesentlichen Gründe dafür, dass viele Anwender hierbei „Etwas Anderes“ auswählen, statt den automatischen Installationsprozess zu nutzen.

Wir erstellen nun zunächst die root-Partition, in der das Betriebssystem erstellt wird. Dazu wählen wir zunächst den freien Speicherplatz aus und klicken daraufhin auf das Pluszeichen, das unter der Auflistung der Partitionen zu finden ist. Anschließend öffnet sich das Fenster, das in Abbildung 5.7 zu sehen ist. Hier müssen wir nun den gewünschten Speicherplatz vorgeben. Die Mindestanforderung liegt bei etwa 15 GB. Für einen reibungslosen Betrieb ist es aber besser, mindestens 25 GB vorzugeben. Da unser freier Speicherplatz ausreichend groß ist, wählen wir jedoch mit 50 GB einen noch etwas großzügigeren Bereich.

Partition erstellen

Größe: - + MB

Typ der neuen Partition: ☐ Primär ☒ Logisch

Position der neuen Partition: ☒ Anfang dieses Bereichs ☐ Ende dieses Bereichs

Benutzen als:

Einbindungspunkt:

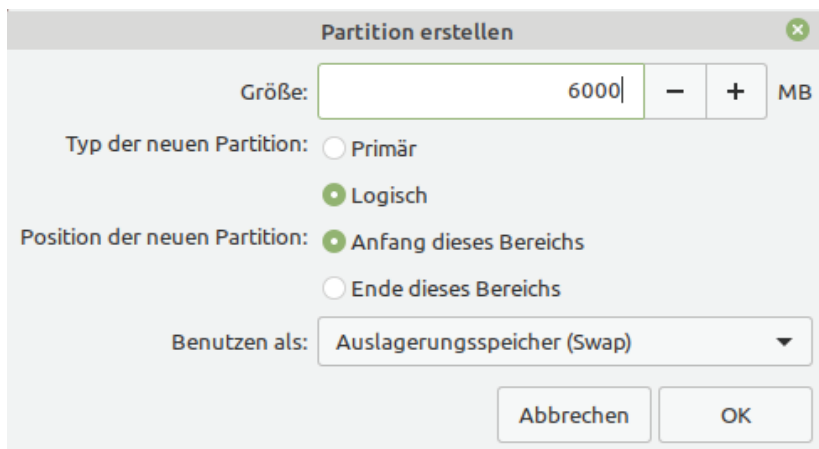
Abbrechen OK

Abb. 5.7 Das erstellen der root-Partition

Bei „Typ der neuen Partition“ ist es notwendig, „Logisch“ auszuwählen. Der Unterschied besteht hierbei darin, dass primäre Partitionen einen etwas höheren Stellenwert haben. Zum Starten des Systems ist beispielsweise eine primäre Partition erforderlich. Allerdings unterstützen gewöhnliche Computersysteme maximal vier primäre Partitionen. Da wir insgesamt drei neue Partitionen erstellen werden, ist es sehr wahrscheinlich, dass dieser Wert überschritten wird. Wenn auf dem Rechner jedoch nur drei primäre Partitionen vorhanden sind, können wir in den übrigen Bereich beliebig viele logische Partitionen einsetzen. Daher wählen wir diese Alternative.

Wir setzen die neue Partition an den Anfang des freien Bereichs. Es wäre jedoch auch problemlos möglich, sie am Ende einzufügen. Die root-Partition muss dem Typ „Ext4-Journaling-Dateisystem“ entsprechen. Wichtig ist es, den Einbindungspunkt festzulegen. Das Root-Verzeichnis steht ganz am Anfang des Dateisystems. Deshalb wählen wir den Eintrag aus, der nur aus einem Schrägstrich besteht.

Die zweite notwendige Partition ist der swap-Bereich. Dieser kommt zum Einsatz, wenn der Arbeitsspeicher voll ist. In diesem Fall lädt das System die Daten auf die Festplatte, damit sie zu einem späteren Zeitpunkt wieder zur Verfügung stehen. Hierfür müssen Sie wieder auf das Pluszeichen unter der Auflistung der Partitionen klicken. Je kleiner der Arbeitsspeicher ist, desto größer sollte der swap-Bereich sein. Bei Systemen mit einem Arbeitsspeicher von 2 GB oder weniger ist es ratsam, mindestens die doppelte RAM-Größe zu verwenden. Bei größerem Arbeitsspeicher ist der einfache oder der eineinhalbfache Wert ausreichend. In unserem Beispiel entscheiden wir uns für einen swap-Bereich von 6 GB. Das ist ideal für ein System mit 4 GB RAM. Nach der Einstellung der Größe müssen Sie in der Zeile mit der Beschriftung „Benutzen als“ die Option „Auslagerungsspeicher (swap)“ auswählen.

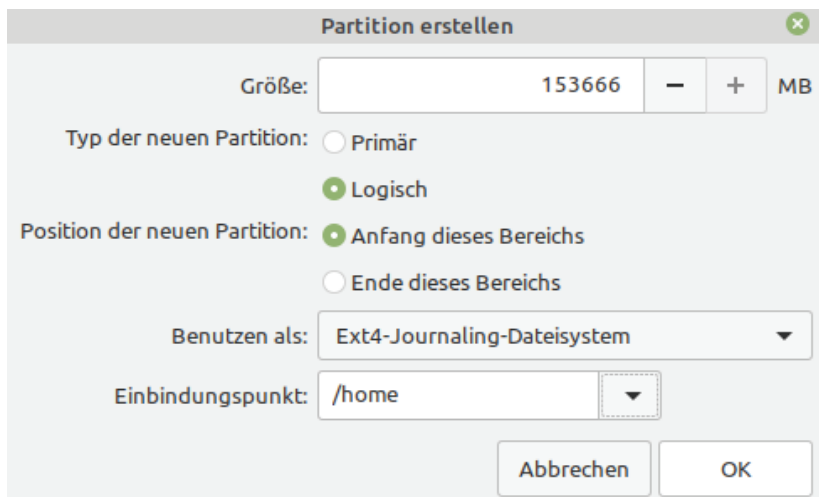


The dialog box 'Partition erstellen' has a title bar with a close button. It contains the following fields and options:

- Größe:** A text input field containing '6000', followed by minus and plus buttons, and the unit 'MB'.
- Typ der neuen Partition:** Two radio buttons: 'Primär' (unselected) and 'Logisch' (selected).
- Position der neuen Partition:** Two radio buttons: 'Anfang dieses Bereichs' (selected) and 'Ende dieses Bereichs' (unselected).
- Benutzen als:** A dropdown menu showing 'Auslagerungsspeicher (Swap)'.
- Buttons:** 'Abbrechen' and 'OK' at the bottom right.

Abb. 5.8 Das Erstellen der swap-Partition

Schließlich erstellen wir die home-Partition für die persönlichen Daten. Dabei können wir den kompletten übrigen Speicherplatz verwenden – vorausgesetzt, dass wir nicht noch ein weiteres Betriebssystem installieren möchten. Hierbei ist wieder ein „Ext4-Journaling-Dateisystem“ notwendig. Als Einbindungspunkt wählen wir „/home“ aus. Abbildung 5.9 zeigt die Einstellungen.



The dialog box 'Partition erstellen' has a title bar with a close button. It contains the following fields and options:

- Größe:** A text input field containing '153666', followed by minus and plus buttons, and the unit 'MB'.
- Typ der neuen Partition:** Two radio buttons: 'Primär' (unselected) and 'Logisch' (selected).
- Position der neuen Partition:** Two radio buttons: 'Anfang dieses Bereichs' (selected) and 'Ende dieses Bereichs' (unselected).
- Benutzen als:** A dropdown menu showing 'Ext4-Journaling-Dateisystem'.
- Einbindungspunkt:** A text input field containing '/home', followed by a dropdown arrow button.
- Buttons:** 'Abbrechen' and 'OK' at the bottom right.

Abb. 5.9 Das Erstellen der home-Partition

Nun sind alle gewünschten Partitionen erstellt. Die Auflistung der neuen Bereiche sollte daraufhin etwa wie in Abbildung 5.10 aussehen. Den Eintrag unter „Gerät für die Bootloader-Installation“ lassen wir unverändert bei „/dev/sda“.

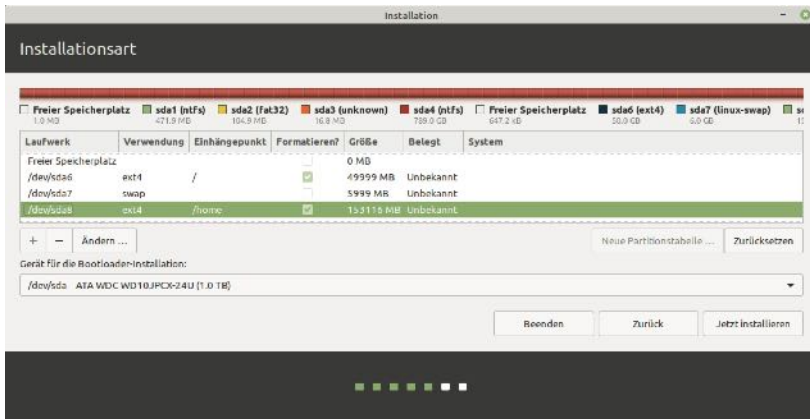


Abb. 5.10 Die neu erstellten Partitionen

Wenn Sie nun auf „Jetzt installieren“ klicken, beginnt der Installationsprozess. Dieser nimmt einige Minuten in Anspruch. Wenn er abgeschlossen ist, können Sie das Live-System weiterhin nutzen. Wenn Sie das Gerät abschalten und neu starten, steht jedoch das fest installierte Linux-Betriebssystem zur Verfügung. Fall Sie ein Dual-Boot-System installiert haben, können Sie dann bei jedem Startvorgang auswählen, welches Betriebssystem gestartet werden soll.

5.2 OpenSUSE

Die nächste Linux-Distribution, die wir in diesem Buch vorstellen, ist openSUSE. Hierbei handelt es sich um eine Entwicklung der SUSE GmbH mit Sitz in Nürnberg. Diese Abkürzung steht für Software- und Systementwicklung. Das Unternehmen begann bereits in den 90-er Jahren – also bereits kurz nach der Entwicklung des Linux-Kernels – mit dem Vertrieb von Linux-Distributionen. Allerdings hatte es zu diesem Zeitpunkt noch keine eigene Distro entwickelt. Stattdessen vertrieb es

Slackware. Diese Distribution ergänzte SUSE jedoch bereits um eine Eigenentwicklung. Dabei handelte es sich um YaST – eine Software für die Konfiguration der Linux-Distribution. 1996 stellte das Unternehmen dann seine erste eigene Version dar. Diese trug den Namen SUSE Linux. Sie basierte auf *jurix* – einer Distribution, die ihrerseits von Slackware abgeleitet war. Daher handelt es sich auch bei SUSE Linux um ein Slackware-Derivat. Das führt dazu, dass openSUSE einige erhebliche Unterschiede zu den beiden bisher vorgestellten Distributionen Knoppix und Linux Mint aufweist, die beide von Debian abstammen. Bei den ersten Versionen der Software handelte es sich um proprietäre Distributionen.

1997 kam dann eine Version hinzu, die für die Verwendung in großen Unternehmen vorgesehen war: SUSE Linux Professional. Diese wird ebenfalls bis heute weitergeführt, wobei sie mittlerweile den Namen SUSE Linux Enterprise Server (SLES) trägt. 2003 kam es schließlich zur Übernahme durch den US-amerikanischen Softwarekonzern Novell. Dieser führte in der Folgezeit einige wichtige Umstrukturierungen durch. Von besonderer Bedeutung war dabei, dass die Version für private Desktop-PCs fortan als Open-Source-Software fortgeführt werden sollte. So kam es 2006 zur Umbenennung in openSUSE.

openSUSE ist in zwei unterschiedlichen Versionen erhältlich: Tumbleweed und Leap. Bei Tumbleweed handelt es sich um ein sogenanntes Rolling Release. Das bedeutet, dass es hierbei ständig zu kleineren oder größeren Updates kommt – ohne feste Versionen. Das führt zwar dazu, dass die Nutzer stets die aktuellsten Entwicklungen umsetzen können. Allerdings sind die Systeme nicht immer stabil. Bei Leap handelt es sich hingegen um feste Versionen mit größeren Update-Zyklen. Das führt zu einer höheren Stabilität. Wenn Sie openSUSE installieren möchten, müssen Sie sich daher zunächst entscheiden, ob Sie Tumbleweed oder Leap verwenden möchten. In diesem Buch stellen wir die Verwendung von Leap dar. Die Unterschiede sind dabei jedoch nur minimal.

Wenn Sie openSUSE installieren möchten, müssen Sie zunächst den folgenden Link öffnen: <https://software.opensuse.org/distributions>. Hier können Sie dann zwischen Tumbleweed und Leap auswählen. Wenn sie

sich für Leap entscheiden und den entsprechenden Bereich anklicken, gelangen Sie zu der Seite, die in Abbildung 5.11 zu sehen ist. Hier sehen Sie vier verschiedene Bereiche: Installation, JeOS, Live und Portierungen. Die Bereiche JeOS und Portierungen sind für uns nicht von Interesse. Bei JeOS handelt es sich um eine Version für Virtualisierungs- und Cloud-Anwendungen. Unter Portierungen sind Versionen für andere Rechner-Architekturen verfügbar. Dass hierbei zwei unterschiedliche Bereiche für die Installation und für die Live-Version verfügbar sind, zeigt jedoch bereits einen großen Unterschied zur Installation von Linux Mint auf. Bei openSUSE ist es nicht möglich, die Installation aus der Live-Version zu veranlassen. Wenn Sie diese Distribution zunächst ausprobieren möchten, ohne sie zu installieren, müssen Sie daher die Live-Version herunterladen und damit einen bootfähigen USB-Stick erstellen. Wenn Sie das System dann fest installieren möchten, ist es notwendig, diesen Vorgang mit der Installations-Version zu wiederholen.



Abb. 5.11 Die Download-Seite von openSUSE

In diesem Buch verzichten wir jedoch auf die Vorstellung der Live-Version und beginnen direkt mit der Installation. Auch hierfür stehen wieder zwei Versionen zur Auswahl: ein DVD-Image und ein Netzwerk-Image. Es ist ratsam, das DVD-Image zu verwenden. Das Netzwerk-Image muss während der Installation umfangreiche Daten herunterladen, was manchmal zu Problemen führen kann.

Nachdem Sie die iso-Datei heruntergeladen haben, ist es wieder notwendig, einen bootfähigen USB-Stick zu erstellen. Theoretisch ist es möglich, auch hierfür wieder den Universal USB Installer zu verwenden. Allerdings kommt es bei dieser Software häufig zu Fehlern, wenn Sie ein Abbild für openSUSE erstellen möchten. Daher bietet es sich an, eine andere Software dafür zu verwenden – beispielsweise Rufus (<https://rufus.ie/>). Wenn Sie Linux Mint oder eine andere Linux-Distribution installiert haben oder als Live-System verwenden, ist diese Aufgabe ebenfalls ganz einfach. In diesem Fall müssen Sie die heruntergeladene Datei einfach mit der rechten Maustaste anklicken und im Kontextmenü daraufhin „Startfähigen USB-Stick erstellen“ auswählen – so wie dies in Abbildung 5.12 zu sehen ist.

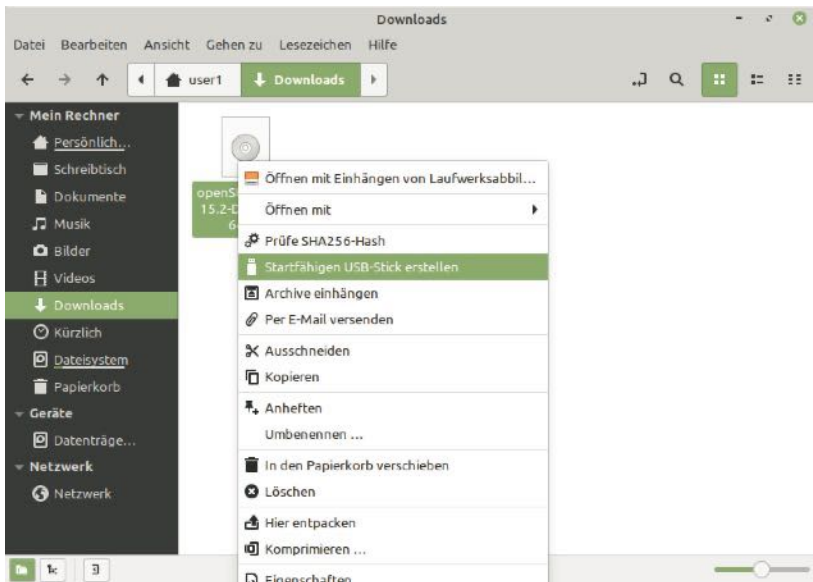


Abb. 5.12 Einen startfähigen USB-Stick mit Linux Mint erstellen

Nun können wir mit der Installation von openSUSE beginnen. Dafür ist es notwendig, den Computer bei eingeglegtem USB-Stick neu zu starten. Daraufhin sollte eine Menüauswahl erscheinen. Hier müssen Sie „Install“ auswählen. Im nächsten Schritt müssen Sie die Sprache und die Tastaturbelegung vorgeben. Darüber hinaus können Sie auswählen,

welche Desktop-Oberfläche Sie verwenden möchten. Wir entscheiden uns hierbei für KDE, um auch diese Option vorzustellen. Doch können Sie auch gerne eine der übrigen Alternativen auswählen.

Der Installer partitioniert die Festplatte außerdem automatisch nach seinen Anforderungen. Wenn ein leerer Bereich vorhanden ist, nutzt openSUSE diesen für die Installation. Falls dies nicht der Fall ist, greift der Installer auf die leeren Bereiche einer anderen Partition zu. Sie müssen die Auswahl lediglich bestätigen. Dabei ist es sinnvoll, die entsprechenden Partitions-Nummern genau zu überprüfen, um zu verhindern, dass Sie Bereiche mit wichtigen Daten versehentlich löschen. Sobald Sie alle Vorbereitungsmaßnahmen abgeschlossen haben, beginnt der Installationsprozess, der einige Zeit in Anspruch nehmen kann.

Damit openSUSE startet, ist zunächst ein Neustart notwendig. Wenn Sie dann im Boot-Menü dieses Betriebssystem auswählen, sollte die Oberfläche erscheinen, die in Abbildung 5.13 zu sehen ist.

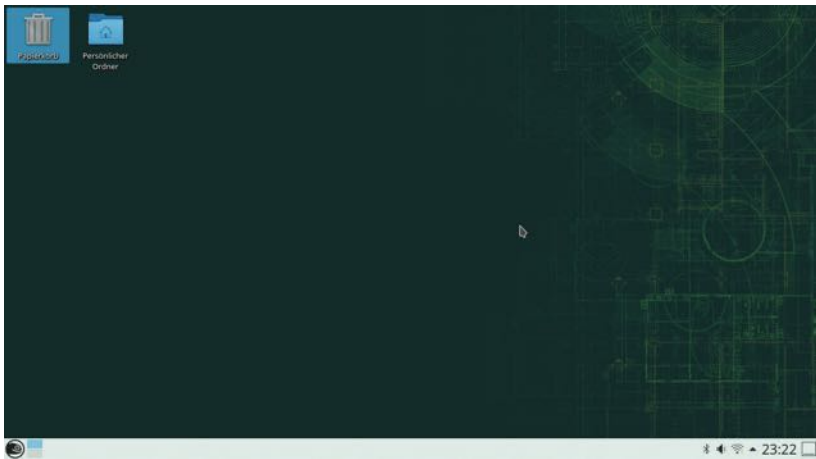


Abb. 5.13 Die Oberfläche von openSUSE

Auf den ersten Blick erscheint die Oberfläche recht ähnlich wie bei den übrigen Alternativen, die wir bisher vorgestellt haben. Wenn Sie jedoch

auf das openSUSE-Symbol in der linken unteren Ecke klicken, werden die ersten Unterschiede bemerkbar. Das ist in Abbildung 5.14 zu sehen.

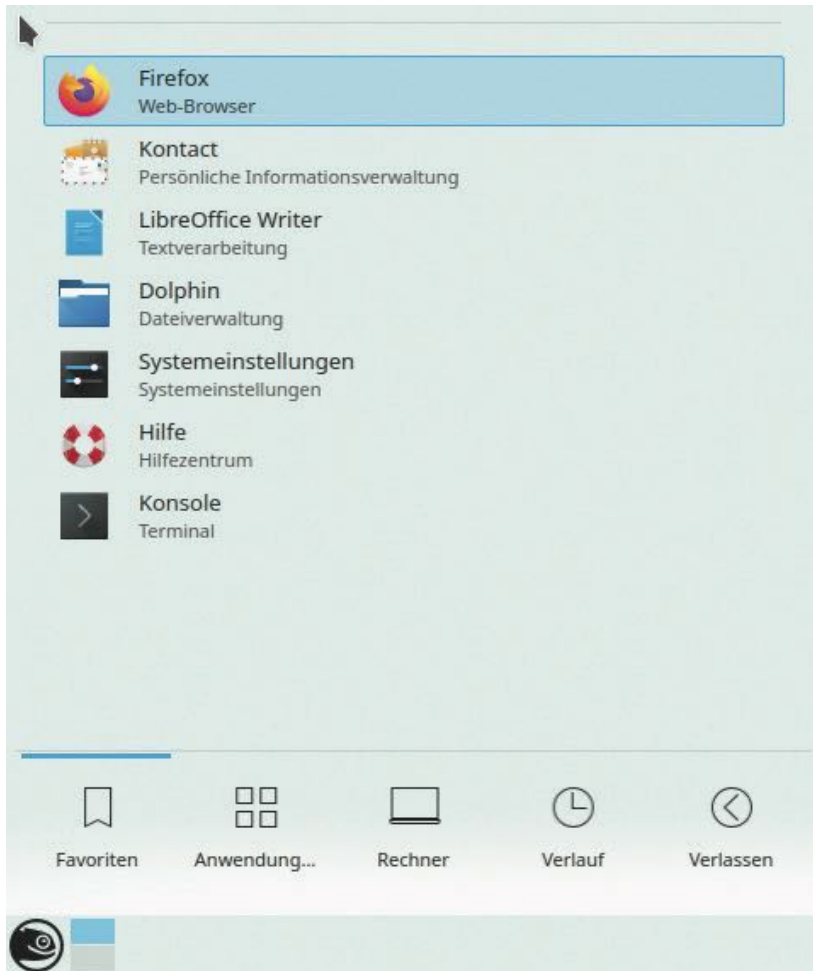


Abb. 5.14 Das Menü von openSUSE

Das zeigt bereits, dass für die Steuerung des Systems eine gewisse Umgewöhnung notwendig ist, wenn Sie bisher Windows verwendet ha-

ben. Doch stehen die wichtigsten Funktionen schnell zur Verfügung. Beispielsweise haben Sie hier einen direkten Zugang zur Konsole, die für die Steuerung von Linux von großer Bedeutung ist. Wenn Sie in der unteren Zeile auf „Anwendungen“ klicken, gelangen Sie auch zu allen übrigen Programmen. Abb. 5.15 zeigt, dass auch openSUSE vielfältige Möglichkeiten bietet.

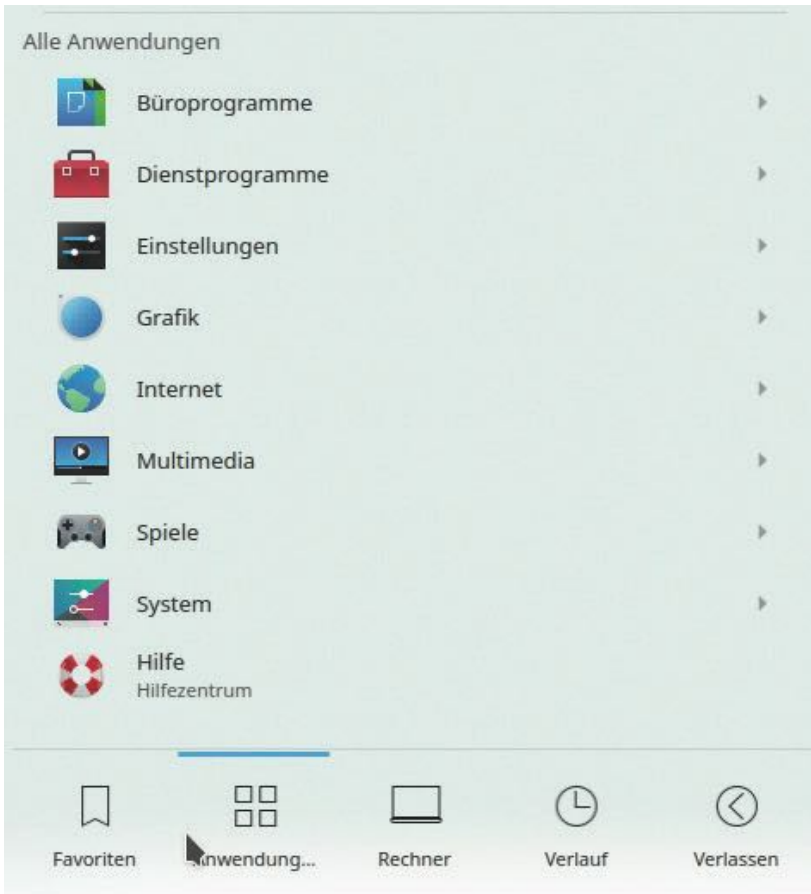


Abb. 5.15 Die Anwendungen unter openSUSE

Damit ist die Vorstellung von openSUSE abgeschlossen. Da diese Oberfläche jedoch zahlreiche Unterschiede sowohl zu Windows als auch zu den bisher vorgestellten Linux-Distributionen aufweist, ist es empfehlenswert, hier eine Weile zu verweilen, um sich mit den Funktionen vertraut zu machen.

5.3 Debian

Als Nächstes lernen wir die Distribution Debian kennen. Hierbei handelt es sich um eine der ältesten und einflussreichsten Linux-Distributionen. Sie entstand bereits 1993 – also kurz nach der Vorstellung des Linux-Kernels. Bis heute zählt Debian zu den am häufigsten genutzten Linux-Distributionen. Insbesondere im Bereich der Webserver nimmt diese Distro eine Führungsposition ein. Doch auch für Desktop-PCs ist sie beliebt. Dabei ist außerdem hervorzuheben, dass sie den Ursprung sowohl von Ubuntu als auch von Linux Mint darstellt. Hierbei handelt es sich um Debian-Derivate, die bis heute auf diese Distribution angewiesen sind. Das zeigt, dass sie auch über die eigentliche Verwendung hinausgehend eine enorme Bedeutung hat.

Die Verwendung von Debian bietet viele Vorteile. Beispielsweise ist diese Distribution für sehr viele verschiedene Rechnerarchitekturen verfügbar. Außerdem umfasst sie besonders viele Programme. Hierbei sind rund 57.000 Programmpakete inbegriffen. Das stellt sehr vielfältige Anwendungsmöglichkeiten sicher. Darüber hinaus ist Debian sehr stabil und bietet zahlreiche Möglichkeiten für die Personalisierung der Funktionen.

Allerdings gibt es auch einige Nachteile. Einer der wesentlichen Gründe dafür, dass Derivate wie Ubuntu oder Linux Mint der ursprünglichen Version im Desktop-Bereich den Rang abgelaufen haben, besteht darin, dass diese deutlich einfacher zu installieren waren. Während für die Verwendung von Debian gute Computerkenntnisse notwendig waren, eigneten sich die anderen beiden Distributionen auch gut für Anfänger. Zwar hat sich in dieser Hinsicht in der Zwischenzeit viel getan. Wie wir später sehen werden, ist für Debian mittlerweile ebenfalls ein Ins-

taller verfügbar, der die Installation sehr einfach gestaltet. Allerdings ist dieser Prozess nach wie vor etwas komplizierter. Auch die Anwendung ist nicht immer ganz so intuitiv wie bei anderen Distributionen.

Hinzu kommt, dass die Unterstützung für Anwendungen aus den Bereichen Unterhaltung und Spiele hierbei nicht allzu gut ist. Viele Privatanwender legen jedoch gerade auf diesen Aspekt viel Wert. Hierbei bieten Linux Mint, Ubuntu und einige weitere beliebte Distros deutlich mehr Möglichkeiten. Darüber hinaus sind die Entwicklungszyklen bei Debian vergleichsweise lang. Das sorgt zwar für eine hohe Stabilität. Allerdings ist deshalb nicht immer die neueste Software verfügbar.

5

Eine erste Schwierigkeit bei der Verwendung von Debian besteht darin, dass hierfür unzählige verschiedene Versionen verfügbar sind. Vielen Anfängern fällt es daher schwer, herauszufinden, welche sich für sie eignet. Hierbei ist es in erster Linie wichtig, auf die Rechnerarchitektur zu achten. Wie bereits erwähnt, bietet Debian in dieser Hinsicht vielfältige Möglichkeiten. Für gewöhnliche PCs ist es wichtig, stets darauf zu achten, eine Version für die Architektur AMD64 auszuwählen. Darüber hinaus bestehen einige Unterschiede hinsichtlich der Installationsform. Es gibt Angebote, die sich komplett herunterladen und per DVD installieren lassen. Darüber hinaus bestehen Versionen, die nur eine CD mit geringem Datenumfang benötigen. Deshalb müssen jedoch während der Installation sehr umfangreiche Daten heruntergeladen werden. Schließlich besteht die Möglichkeit, eine Live-DVD beziehungsweise einen Live-USB-Stick zu gestalten. Dabei wird zunächst ein Live-System gestartet, von dem aus Sie dann die Installation veranlassen können – genau wie dies bei Linux Mint der Fall war. Für dieses Buch wählen wir diese Alternative aus, da sie besonders einfach und praktisch ist. Außerdem können Sie auf diese Weise die Distribution auch kennenlernen, ohne sie gleich zu installieren. Die entsprechende Version steht unter dem folgenden Link zum Download bereit: <https://cdimage.debian.org/debian-cd/current-live/amd64/iso-hybrid/>.

Hier sehen Sie, dass Sie Debian mit verschiedenen Desktop-Oberflächen verwenden können: mit GNOME, Cinnamon, KDE, MATE und einigen weiteren. Hierbei können Sie die Version auswählen, die Sie am

stärksten interessiert. In diesem Buch wird die Installation anhand der Cinnamon-Oberfläche demonstriert. Die Unterschiede zu den übrigen Möglichkeiten sind jedoch nur gering. Wichtig ist es, hierbei stets ein Systemabbild mit der Endung .iso herunterzuladen.

Nach dem Download der entsprechenden Datei ist es wieder notwendig, mit dem Universal USB Installers einen bootfähigen USB-Stick zu erstellen. Danach müssen Sie das System neu starten.

Beim Booten können Sie - im Gegensatz zu den bisher vorgestellten Distributionen - auswählen, ob Sie das System direkt installieren oder zunächst die Live-Version ausprobieren möchten. Wir entscheiden uns dafür, zu Beginn die Live-Version zu starten. Dabei gelangen Sie zu dem Bildschirm, der in Abbildung 5.16 zu sehen ist.

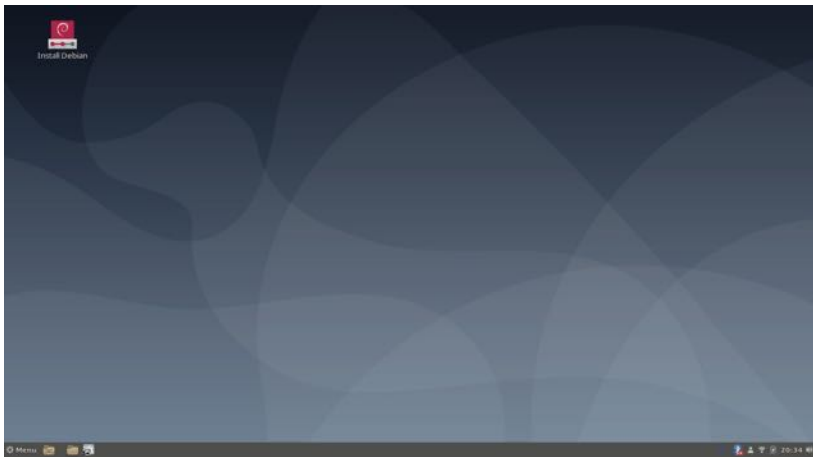


Abb. 5.16 Die Oberfläche der Debian-Live-Version

Nun können Sie wieder die Funktionen des Systems ausprobieren. Da wir die Version mit Cinnamon-Oberfläche ausgewählt haben, ist die Darstellung dabei recht ähnlich wie bei Linux Mint in Kapitel 5.1. Dennoch werden beim Ausprobieren der Distribution bereits einige Unterschiede deutlich. Beispielsweise wird Debian mit sehr vielen verschiedenen Programmen ausgeliefert, sodass die Anwendungsmöglichkeiten sehr vielfältig sind.

Für die Installation klicken wir nun auf das Feld „Install Debian“ links oben auf dem Desktop. Damit öffnen wir den Installations-Assistenten. Dieser führt Sie automatisch durch die Installation. Dabei können Sie wieder die Sprache, die Tastaturbelegung und Ihren Standort eingeben.

Wie bei allen Installationsvorgängen stellt auch hierbei die Zuordnung der Partitionen den wichtigsten Schritt dar. Für unser Beispiel haben wir zuvor die übrigen Linux-Systeme gelöscht, sodass ein freier Bereich auf der Festplatte verfügbar ist. Daher können wir den Punkt „Ersetze eine Partition“ auswählen und daraufhin den freien Bereich für Debian verwenden. Wie immer ist es auch hierbei möglich, die komplette Festplatte zu formatieren und Linux als alleiniges Betriebssystem zu verwenden. Wenn kein geeigneter Freiraum vorhanden ist, können Sie auch eine bestehende Partition verkleinern. Es ist jedoch empfehlenswert, den benötigten Platz vor der Installation manuell freizugeben.

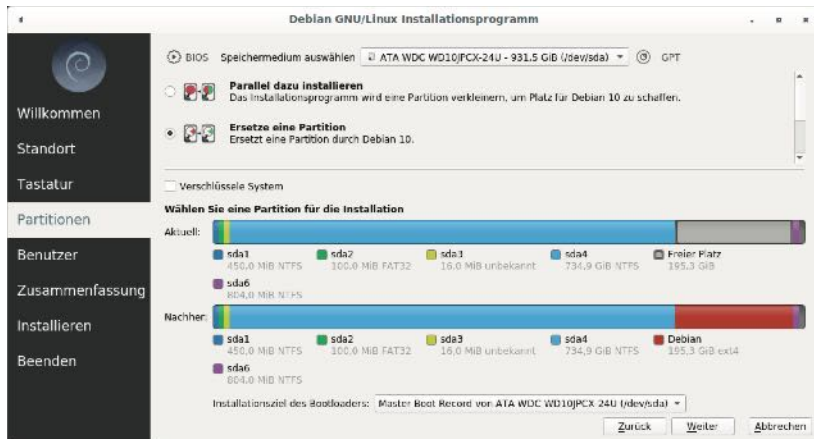


Abb. 5.17 Die Belegung der Festplatte für die Installation von Debian

Abschließend ist es noch notwendig, sich als Benutzer einzutragen und ein Passwort vorzugeben. Danach erhalten Sie noch eine Zusammenfassung aller getätigten Einstellungen. Wenn Sie diese bestätigen, beginnt der eigentliche Installationsprozess. Nach einer gewissen Wartezeit steht Debian dann auf der Festplatte zur Verfügung.

5.4 Manjaro Linux

Als nächstes Beispiel soll die Distribution Manjaro Linux vorgestellt werden. Hierbei handelt es sich um eine recht neue Entwicklung. Die erste Version der Distro erschien erst 2011. Allerdings wurde Manjaro Linux nicht von Grund auf neu gestaltet. Hierbei handelt es sich um ein Arch-Derivat. Manjaro Linux ist demnach von Arch Linux abgeleitet und ist von dessen Entwicklung nach wie vor abhängig. Mit diesem Arch-Derivat lernen Sie noch eine weitere Hauptlinie unter den Linux-Distributionen kennen. Trotz des geringen Alters hat sich Manjaro Linux schnell zu einer der beliebtesten Distributionen entwickelt. Die Seite DistroWatch, die die Nutzung der verschiedenen Distributionen dokumentiert, hat Manjaro Linux 2018 als beliebtes Linux-Betriebssystem ermittelt. 2019 belegte Manjaro immerhin noch den zweiten Platz.

Arch Linux verwendet ein Rolling-Release-Modell. Wie bereits dargestellt, bedeutet das, dass es keine klar abgegrenzten Versionen gibt. Aktualisierungen werden umgehend veröffentlicht, um das Betriebssystem stets auf dem aktuellen Stand zu halten. Diese Vorgehensweise geht jedoch häufig zulasten der Stabilität. Arch-Updates sind daher oftmals fehleranfällig.

Manjaro Linux befolgt als Arch-Derivat im Prinzip ebenfalls das Rolling-Release-Modell. Allerdings gibt diese Distribution dem Anwender auch die Möglichkeit, die Stabilität zu erhöhen. Dabei kann er aus drei verschiedenen Repositories auswählen. Dabei bestehen die folgenden Optionen:

- ▶ **Unstable:** Hier werden die neuesten Updates von Arch Linux übernommen. Die Verzögerung bei der Veröffentlichung beträgt in der Regel nur rund drei Tage.
- ▶ **Testing:** Hier werden die Pakete aus dem Unstable-Repository übernommen, die einen einführenden Test bestanden haben.
- ▶ **Stable:** Hier finden ausschließlich Pakete Einzug, die das Entwickler-Team als stabil bewertet hat. Dadurch kommt es jedoch zumindest bei größeren Updates zu einer Verzögerung von mehreren Wochen.

Manjaro Linux legt viel Wert auf die Benutzerfreundlichkeit. Das stellt einen der wesentlichen Unterschiede zu Arch Linux dar. Während für die Verwendung der Ursprungs-Distribution fortgeschrittene Kenntnisse notwendig sind, richtet sich Manjaro Linux auch an Anfänger. Hierbei ist insbesondere das eigens entwickelte Installationsprogramm zu nennen. Dieses führt den Anwender über eine grafische Benutzeroberfläche durch den Installationsprozess und gestaltet diese Aufgabe dadurch einfach und intuitiv.

Auch diese Distribution ist mit mehreren Desktop-Oberflächen verfügbar. Als Standard zählen hierbei Xfce, KDE und GNOME. Mittlerweile gibt es aber auch noch einige weitere Alternativen: beispielsweise Cinnamon, MATE, LXDE und Openbox. Hier stellen wir die Verwendung von Xfce vor – einer Desktop-Oberfläche, die wir bislang noch nicht behandelt haben. Auf diese Weise lernen Sie eine weitere beliebte Möglichkeit kennen.

Um Manjaro Linux zu nutzen, müssen Sie zunächst wieder die .iso-Datei herunterladen. Dafür können Sie den folgenden Link nutzen: <https://manjaro.org/download/>. Hier sind alle Standard-Desktop-Oberflächen aufgeführt. Sehr interessant ist hierbei auch, dass eine kurze Beschreibung der verschiedenen Alternativen aufgeführt ist. Das erleichtert die Auswahl. Wenn Sie dem Beispiel in diesem Buch entsprechend Xfce nutzen möchten, müssen Sie den ersten Link anklicken. Sie können aber auch gerne eine der übrigen Alternativen ausprobieren. Der Installationsprozess läuft dabei sehr ähnlich ab.

Wie üblich ist es auch bei Manjaro Linux notwendig, zunächst einen bootfähigen USB-Stick zu erstellen. Theoretisch ist es hierbei wieder möglich, den Universal USB Installer zu verwenden. Allerdings kann es bei der Zusammenstellung dieser Distribution gelegentlich zu Problemen kommen. Sollte sich der bootfähige USB-Stick damit nicht erstellen lassen, ist es sinnvoll, zur bereits vorgestellten Alternative Rufus zu greifen. Der Vorgang läuft dabei fast genau gleich ab. Lediglich die grafische Darstellung weist leichte Unterschiede auf. Abbildung 5.18 zeigt, wie Sie den bootfähigen USB-Stick mit diesem Werkzeug erstellen.

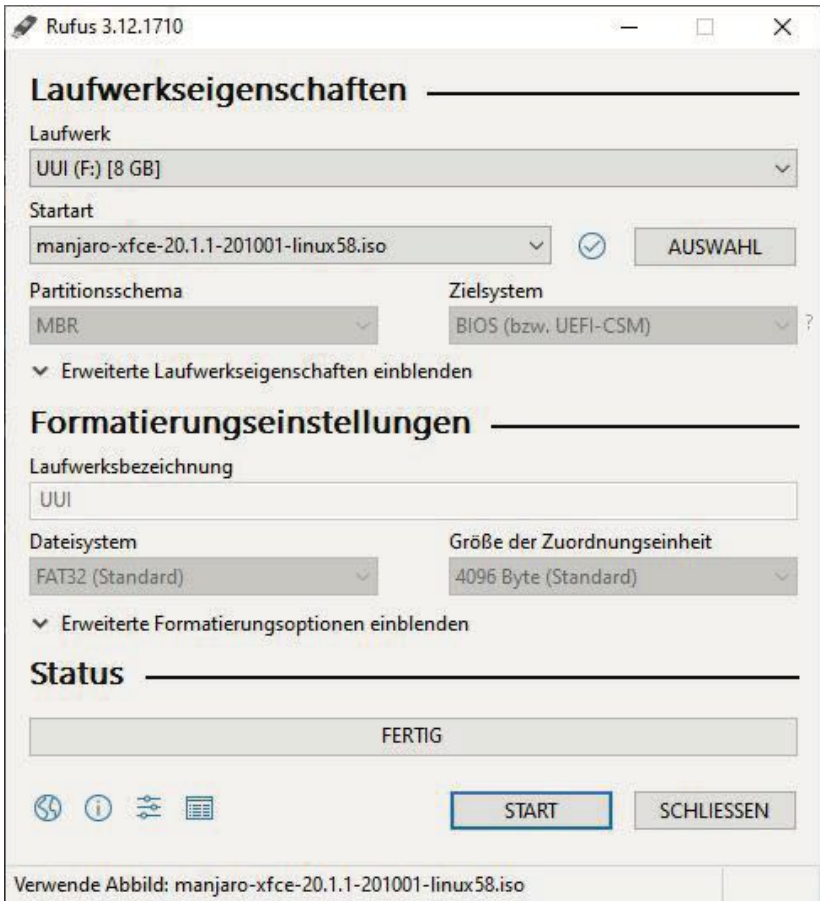


Abb. 5.18 Einen bootfähigen USB-Stick mit Rufus erstellen

Manjaro Linux wird als Live-System ausgeliefert, das es nach dem Booten ermöglicht, das System fest zu installieren. Daher starten wir den Computer nun mit eingelegtem USB-Stick neu. Die Live-Version gibt uns auch die Möglichkeit, die Distribution kennenzulernen. Bereits direkt nach dem Hochfahren des Systems wird deutlich, dass Manjaro Linux viel Wert auf die Benutzerfreundlichkeit legt und die Verwendung auch für Anfänger so einfach wie möglich gestalten soll. Sie werden hierbei mit einem Willkommens-Fenster begrüßt, das Ihnen die wichtigsten Eigen-

schaften des Betriebssystems vorstellt und Ihnen zeigt, wo Sie Informationen und Hilfe finden. Abbildung 5.19 stellt dieses Fenster dar.



Abb. 5.19 Das Willkommens-Fenster von Manjaro Linux

Wenn Sie dieses Fenster schließen, sehen Sie bereits, dass die grafische Oberfläche hierbei etwas anders als bei den bisher vorgestellten Distributionen gestaltet ist. Das liegt daran, dass wir uns hierbei für die Xfce-Desktopumgebung entschieden haben. Die Gestaltung ist in Abbildung 5.20 zu erkennen. Dabei sehen Sie auch, dass am linken Bildrand das Programm „Install Manjaro“ vorhanden ist. Dabei handelt es sich um den Installer, mit dem wir anschließend das Betriebssystem fest installieren werden.

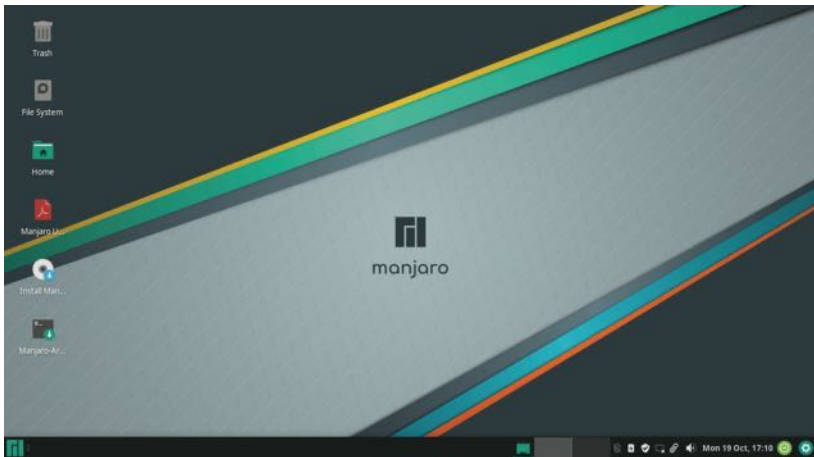


Abb. 5.20 Die Desktop-Oberfläche von Manjaro

Nun betrachten wir auch das Start-Menü, das Sie über das Manjaro-Symbol im linken unteren Bildschirmck erreichen. Genau wie bei den übrigen Distributionen gelangen Sie auf diesem Wege zu den verschiedenen Programmen des Systems. Allerdings weisen die grafische Gestaltung und die Menüführung hierbei ebenfalls einige Unterschiede zu den bisherigen Beispielen auf. Das ist in Abbildung 5.21 zu sehen. Um Manjaro Linux kennenzulernen, können Sie nun die verschiedenen Programme betrachten, die diese Distribution mit sich bringt.



Abb. 5.21 Die Schaltfläche für die Steuerung des Systems

Nachdem wir die Funktionen ausprobiert haben, fahren wir mit der Installation fort. Hierfür kommt das Programm „Install Manjaro“ zum

Einsatz, das auf dem Desktop verfügbar ist. Nachdem wir dieses angeklickt haben, öffnet sich wieder ein motivierendes Willkommens-Fenster, das vermitteln soll, dass dieser Prozess einfach zu bewältigen ist und auch für Anfänger kein Problem darstellt. Das zeigt nochmals, dass diese Distribution viel Wert auf diesen Aspekt legt. Das entsprechende Fenster ist in Abbildung 5.22 zu sehen. Dabei erkennen Sie auch, dass Sie für den Installations-Assistenten aus vielen verschiedenen Sprachen auswählen können.



Abb. 5.22 Das Willkommens-Fenster für die Installation

Anschließend werden Sie durch den Installationsprozess geleitet. Der Ablauf ist hierbei sehr ähnlich wie bei den übrigen Distributionen: Sie müssen eine Sprache auswählen, die Tastaturbelegung vorgeben und einen Benutzer anlegen. Eine kleine Besonderheit besteht hierbei lediglich darin, dass Sie bereits bei der Installation vorgeben können, welches Office-Paket Sie nutzen möchten. Wie immer ist es hierbei sehr wichtig, die Partitionen richtig vorzugeben. Für dieses Beispiel haben wir bereits vor der Installation einen freien Bereich geschaffen. Diesen verwenden wir dann für die Installation, indem wir „Ersetze eine Partition“ auswählen – so wie dies in Abbildung 5.23 zu sehen ist.



Abb. 5.23 Die Festlegung der Partitionen

Nachdem Sie alle Vorgaben gemacht haben, müssen Sie die Installation endgültig bestätigen. Dieser Vorgang nimmt wie immer einige Minuten in Anspruch. Nachdem er abgeschlossen ist, können Sie das System neu starten. Daraufhin steht Manjaro Linux beim Booten als Option zur Auswahl.

5.5 Ubuntu

Als letzte Linux-Distribution soll noch Ubuntu vorgestellt werden. Wie bereits geschrieben, werden wir diese auch für die weiteren Beispiele und Aufgaben in diesem Buch verwenden. Diese lassen sich zwar auch mit den übrigen Distributionen auf ähnliche Weise erledigen. Allerdings kann es dabei zu Verschiedenheiten in der grafischen Darstellung und in der Menüführung kommen. Manchmal sind sogar unterschiedliche Programme erforderlich. Für ein einfacheres Verständnis ist es daher ratsam, ebenfalls Ubuntu zu installieren. Wenn Sie jedoch in den vorherigen Abschnitten eine Distribution entdeckt haben, die Ihren Bedürfnissen besser entspricht, können Sie auch diese verwenden.

Wenn Sie sich an dieser Stelle fragen, weshalb wir diese Distribution für dieses Buch ausgewählt haben, sei gesagt, dass es sich hierbei weitest-

gehend um eine willkürliche Entscheidung handelt. Es gibt zahlreiche weitere Alternativen, die sich auch für Anfänger bestens eignen. Insbesondere die hier ebenfalls bereits vorgestellten Distributionen Linux Mint und Manjaro Linux bieten sehr ähnliche Voraussetzungen und würden sich zu diesem Zweck ebenfalls bestens anbieten. Allerdings ist es erforderlich, sich für eine Distribution zu entscheiden. Die Wahl fiel auf Ubuntu, da es sich hierbei seit dem Erscheinen im Jahr 2004 bis heute um eine der beliebtesten Distributionen handelte. Über viele Jahre hinweg stand Ubuntu auf Platz 1 der am häufigsten verwendeten Distros – laut DistroWatch zwischen 2005 und 2010. Mittlerweile hat die Verwendung zwar etwas abgenommen – beispielsweise konnten in den letzten Jahren Linux Mint und Manjaro Linux mehrmals den ersten Platz einnehmen. Dennoch schaffte es Ubuntu bis heute stets in die Top 5. Das zeigt, dass diese Distribution über die letzten zwei Jahrzehnte gesehen sicherlich das am häufigsten verwendete Linux-System darstellt.

Hinzu kommt, dass Ubuntu einen enormen Einfluss auf die Verbreitung von Linux hatte. Die bis dahin verfügbaren Distributionen richteten sich vorwiegend an ein Fachpublikum mit guten Computer-Kenntnissen. Ubuntu machte Linux hingegen massentauglich. Hierbei handelte es sich um die erste Version, die auch für Anfänger leicht zu installieren und zu nutzen war. Daher steigerte das Erscheinen von Ubuntu die Nutzung von Linux deutlich. Darüber hinaus hatte Ubuntu auch einen großen Einfluss auf weitere Distributionen. Bei Linux Mint – das es nach 2010 mehrfach auf Platz 1 der beliebtesten Distributionen schaffte – handelt es sich beispielsweise um ein Ubuntu-Derivat. Die Ähnlichkeiten sind dabei sehr groß, sodass der Einfluss von Ubuntu unverkennbar ist.

Um Ubuntu zu installieren, ist es genau wie bei den bisher vorgestellten Versionen zunächst notwendig, die Installationsdatei herunterzuladen. Diese stellt der Hersteller direkt auf seiner Homepage zur Verfügung: <https://ubuntu.com/download/desktop>. Hierbei handelt es sich um die Standard-Ausführung mit GNOME-Desktopoberfläche. Auch Ubuntu bietet hierbei noch weitere Alternativen. Diese haben jedoch jeweils einen leicht abgeänderten Namen – beispielsweise Xubuntu mit Xfce,

Kubuntu mit KDE und Ubuntu MATE mit MATE. Diese werden jeweils auf einer separaten Domain angeboten. Sollten Sie eine dieser Versionen vorziehen, sollte es kein Problem darstellen, die entsprechenden Download-Seiten mit einer Suchmaschine Ihrer Wahl zu finden.

Die Installation von Ubuntu erfolgt über eine Live-Version. Hierfür erstellen wir wieder mit dem Universal USB Installer einen bootfähigen USB-Stick. Hierfür ist es notwendig, die Distribution Ubuntu vorzugeben und anschließend die soeben heruntergeladene Datei auszuwählen. Wenn Sie den USB-Stick eingelegt lassen und daraufhin den Computer neu starten, wird die Live-Version geladen.

Ganz zu Beginn erscheint hier ein Fenster, in dem Sie gefragt werden, ob Sie Ubuntu zunächst einmal ausprobieren oder direkt installieren möchten. Wir entscheiden uns dafür, das System auszuprobieren. Die Installation lässt sich auch aus der Live-Version heraus veranlassen. Abbildung 38 stellt die Ubuntu-Desktopoberfläche dar, die daraufhin erscheint. Hierbei handelt es sich um die aktuelle Version 20.04. Da hierbei jedoch halbjährlich neue Versionen erscheinen, die jeweils einige Unterschiede bei der grafischen Gestaltung mit sich bringen, ist es gut möglich, dass Sie hier eine etwas andere Oberfläche vor sich sehen.

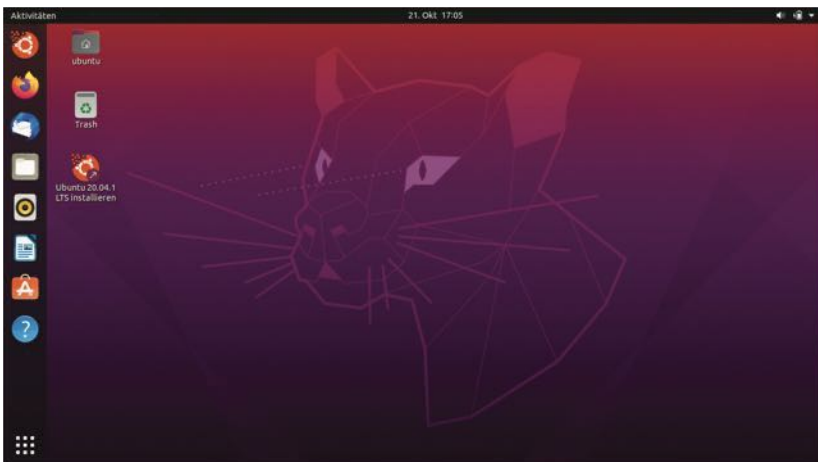


Abb. 5.24 Der Ubuntu-Desktop mit GNOME

An dieser Abbildung wird bereits deutlich, dass die Desktop-Oberfläche hierbei etwas anders als bei Windows und den meisten bisher vorgestellten Linux-Distributionen aufgebaut ist. Auffällig ist, dass die wichtigsten Funktionen zur Steuerung des Systems hierbei am oberen und nicht am unteren Bildrand angebracht sind. Auch bei vielen Fenstern sind die Buttons für verschiedene Aktionen im oberen Bereich positioniert. Für Windows-Nutzer mag dies zunächst etwas ungewöhnlich sein – Sie gewöhnen sich jedoch sicherlich schnell daran.

Ein praktisches Detail stellt die Leiste am linken Bildrand dar. Hier werden automatisch die beliebtesten Anwendungen aufgeführt. Auf diese Weise erhalten Sie einen schnellen und direkten Zugriff darauf. Alle übrigen Programme erreichen Sie über die Schaltfläche mit den neun kleinen Quadraten in der linken unteren Ecke. Abbildung 5.25 zeigt, dass Sie auf diese Weise Zugriff zu vielen verschiedenen Funktionen erhalten.



Abb. 5.25 Der Zugang zu den Anwendungen unter Ubuntu

Nachdem wir uns mit dem System vertraut gemacht haben, können wir mit der Installation beginnen. Der Installations-Assistent ist entweder über die Leiste am linken Bildrand oder über den Desktop erreichbar. Wenn Sie diesen starten, müssen Sie zunächst die Sprache

und die Tastaturbelegung auswählen. Abbildung 5.26 zeigt, dass Sie anschließend zwischen der normalen und einer minimalen Installation auswählen können. Es ist ratsam, sich für die normale Installation zu entscheiden, da die minimale Installation mit deutlich weniger Programmen umgesetzt wird. Dies ist nur sinnvoll, wenn nur sehr wenig Platz auf der Festplatte zur Verfügung steht. Darüber hinaus bietet Ihnen Ubuntu die Möglichkeiten, zu entscheiden, ob Sie auch proprietäre Software von Drittanbietern installieren möchten. Das ist insbesondere dann zu empfehlen, wenn Sie Wert auf Multimedia-Inhalte legen.

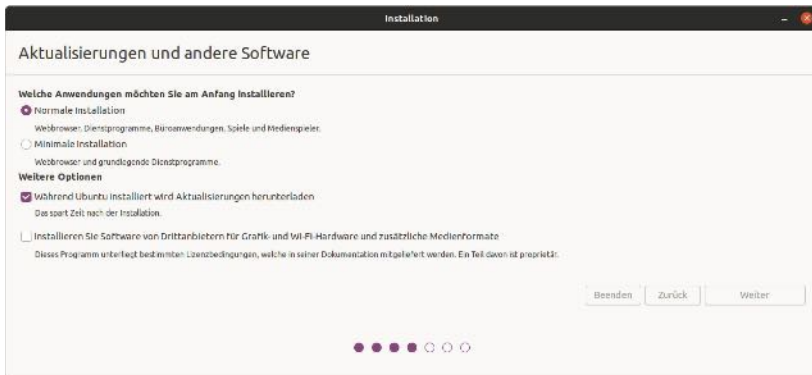


Abb. 5.26 Den Umfang der installierten Programme vorgeben

Im nächsten Schritt haben Sie wieder die Auswahl aus verschiedenen Installationsweisen. Wenn Sie ein Dual-Boot-System einrichten möchten, müssen Sie entweder die erste oder die dritte Option auswählen. Entscheiden Sie sich hingegen für die zweite Alternative, wird die komplette Festplatte – und damit alle vorhandenen Betriebssysteme – gelöscht. Wenn Sie Wert auf eine schnelle und einfache Installation legen, können Sie die erste Option auswählen. Wir stellen hier jedoch wie bei Linux Mint vor, wie Sie die Partitionen selbst vorgeben. Das gibt Ihnen besonders vielfältige Gestaltungsmöglichkeiten. Deshalb wählen wir - wie in Abbildung 5.27 zu sehen - „Etwas Anderes“ aus.

5 Installationsbeispiele für verschiedene Linux-Distributionen

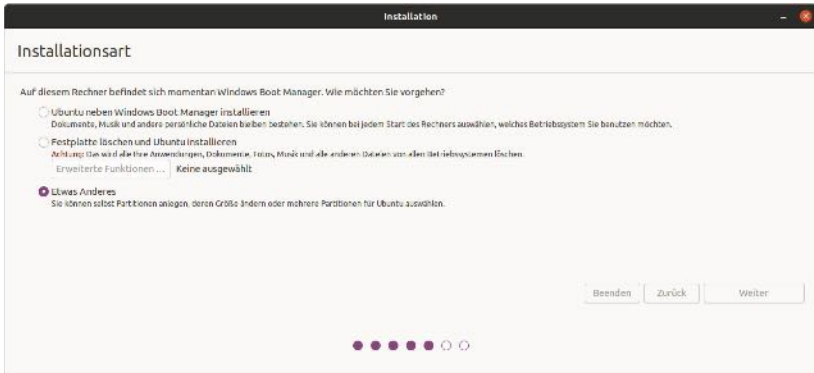


Abb. 5.27 Die Installationsart auswählen

Die Partitionierung läuft genau gleich ab wie bei Linux Mint. Daher geben wir hier nur die Kurzversion wieder. Sollten Sie weitere Details benötigen, können Sie nochmals in Kapitel 5.1 nachschlagen.

Wir gehen davon aus, dass im Rahmen der Vorbereitungsmaßnahmen eine bestehende Partition verkleinert wurde, sodass ein ausreichend großer freier Speicherbereich zur Verfügung steht. Diesen müssen Sie nun in der Liste mit den Partitionen auswählen und auf das Pluszeichen darunter klicken. Daraufhin wird ein neues Fenster geöffnet. Dort können Sie die Partition vorgeben. Zunächst erstellen wir die root-Partition. Dabei sollten Sie eine Größe von mindestens 20 GB vorgeben, eine logische Partition auswählen und das Ext4-Journaling-Dateisystem auswählen. Als Einbindungspunkt für das root-Verzeichnis dient der Schrägstrich. Abbildung 5.28 stellt dies dar.

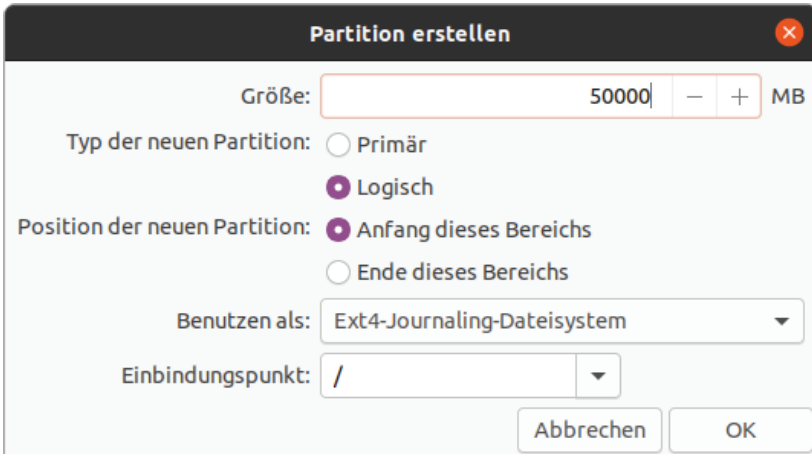


Abb. 5.28 Die root-Partition erstellen

Danach erstellen wir noch die swap-Partition mit einer Größe von 6 GB und den übrigen freien Speicherbereich nutzen wir für die home-Partition – genau nach dem gleichen Muster wie bei der Installation von Linux Mint. Danach sollte der entsprechende Festplattenbereich ungefähr so aussehen, wie dies in Abbildung 5.29 zu sehen ist.

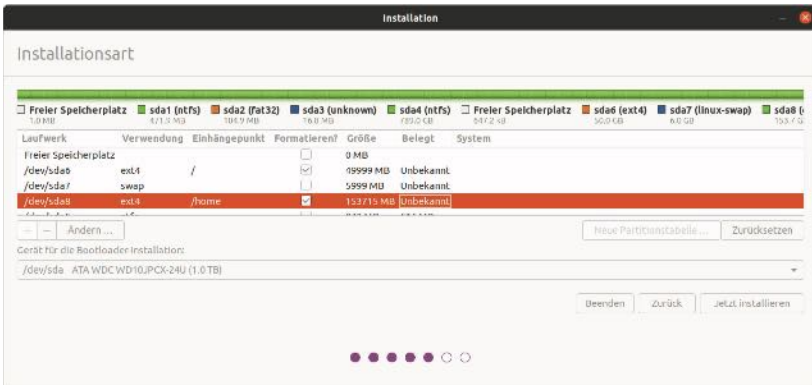


Abb. 5.29 Die Partitionen für die Installation von Ubuntu

Im Feld mit der Aufschrift „Gerät für die Bootloader-Installation“ nehmen wir keine Änderungen vor. Nun können wir auf die Schaltfläche „Jetzt installieren“ klicken. Daraufhin müssen wir nur noch den Standort auswählen und einen Benutzer anlegen. anschließend beginnt der Installationsprozess, der wie immer einige Zeit dauert.

Wenn dieser abgeschlossen ist, steht Ubuntu zur Nutzung bereit. Falls Sie ein Dual-Boot-System installiert haben, können Sie bei jedem Start auswählen, welches Betriebssystem Sie starten. Für die Bearbeitung der folgenden Kapitel ist es sinnvoll, hierbei stets Ubuntu auszuwählen, um alle Beispiele und Aufgaben selbst am Computern ausführen zu können.

5.6 Übungsaufgaben

Suchen Sie im Internet nach der Installationsdatei für MX Linux, der aktuell beliebtesten Distribution. Informieren Sie sich über den Installationsprozess und probieren Sie das System aus.

Lösung:

MX Linux steht unter dem folgenden Link zum Download bereit: <https://mxlinux.org/download-links/>. Die Installation erfolgt hierbei über einen bootfähigen USB-Stick. Um diesen zu erstellen, bietet MX Linux ein eigenes Tool an: den MX Live-USB-Maker. Doch ist auch die Verwendung der bisher vorgestellten Werkzeuge möglich. MX Linux bietet eine Live-Version an, die auch die Installation erlaubt. Auf diese Weise können Sie auch dieses System ausprobieren.

Kapitel 6

Die grundlegenden Funktionen von Linux

Nachdem die theoretischen Vorüberlegungen zu den Aufgaben eines Betriebssystems, die Vorstellung der grundlegenden Eigenschaften von Linux und die Installation der Software abgeschlossen sind, können wir uns nun der praktischen Arbeit zuwenden. Die grundlegenden Oberflächen der einzelnen Distributionen haben wir dabei bereits vorgestellt. Nun ist es an der Zeit, verschiedene Anwendungen und praktische Funktionen kennenzulernen. Der Schwerpunkt liegt hierbei bislang auf Funktionen, die für die private Nutzung sowie für die Arbeit in einem Büro erforderlich sind. Darüber hinaus stellen wir einige grundlegende Ordnungsstrukturen sowie den Ablauf von Programmen und Prozessen in Linux vor.

6.1 Das Dateisystem in Linux

Im ersten Abschnitt dieses Kapitels werfen wir einen Blick auf das Dateisystem in Linux. Hierbei handelt es sich um die Struktur, in der die Dateien auf dem Rechner angeordnet sind. Als Windows-Anwender haben Sie sich wahrscheinlich noch nie tiefer mit dem Dateisystem des Rechners beschäftigt. Das liegt daran, dass dort lediglich die persönlichen Dateien sowie die Programmdateien für Ihre Anwendungen abgelegt sind. Die Strukturen sind hierbei recht einfach verständlich.

Linux folgt jedoch dem UNIX-Prinzip „Everything is a file!“ – auf Deutsch: „Alles ist eine Datei!“. Das bedeutet, dass hierbei alle Ressourcen des Computers wie eine Datei behandelt werden. Daher können wir über das Dateisystem auf sie zugreifen. Das soll nun an einem kleinen Beispiel aufgezeigt werden. Dazu klicken wir zunächst auf den Dateiordner, der links in der Favoritenliste erscheint. Abbildung 6.1 stellt dieses Symbol dar.



Abb. 6.1 Das Symbol für das Dateisystem

Das Fenster, das sich daraufhin öffnet, ist in Abbildung 6.2 zu sehen. Von der grafischen Darstellung abgesehen, wirkt dieses zunächst sehr ähnlich wie bei einem Windows-System. Hier sind die verschiedenen Ordner für die persönlichen Dateien aufgelistet – beispielsweise für Bilder, Dokumente und Downloads.

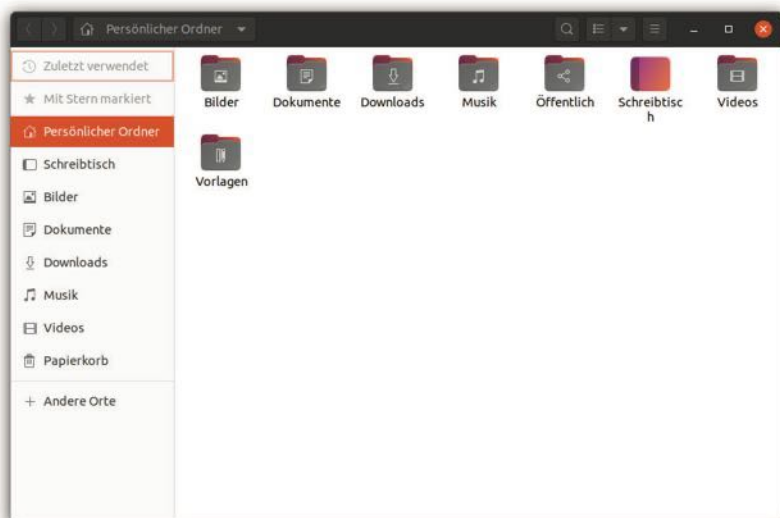


Abb. 6.2 Das Fenster für die Dateiverwaltung

Nun klicken wir jedoch in der Leiste am linken Rand auf das Pluszeichen mit der Beschriftung „Andere Orte“. Abbildung 6.3 zeigt, dass hier nun weitere Ordner erscheinen. Einer davon trägt die Bezeichnung „Rechner“. Hierbei handelt es sich um das root-Verzeichnis von Linux. Der andere Ordner mit der Bezeichnung „Datenträger“ bietet Zugriff auf die

gesamte Festplatte. Wenn Sie ein Dual-Boot-System installiert haben, erreichen Sie auf diese Weise die Dateien, die Sie unter Windows oder macOS verwenden. Auf diesen Punkt werden wir an dieser Stelle nicht näher eingehen. Sie können sich dieses Detail aber dennoch merken, da es sehr praktisch ist, wenn Sie an Dateien weiterarbeiten möchten, die Sie bereits unter Windows oder macOS begonnen haben. Auf diese Weise ist ein einfacher Zugriff möglich – vorausgesetzt, dass Sie die Daten nicht verschlüsselt haben.

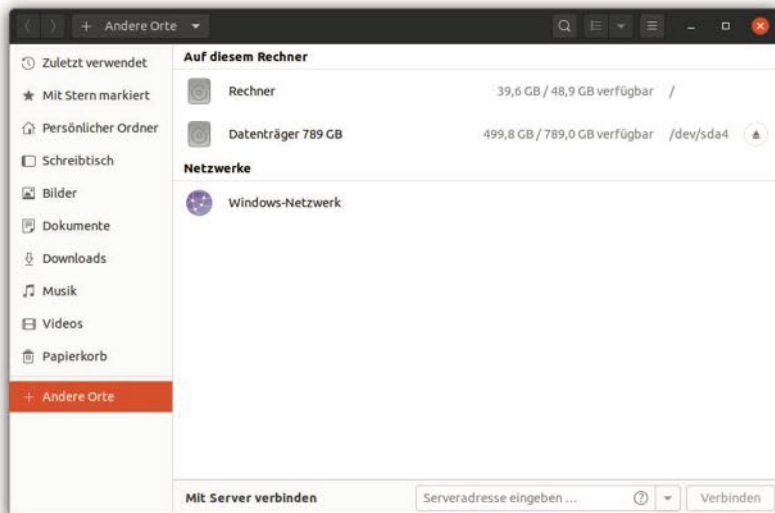


Abb. 6.3 Der Zugriff auf das root-Verzeichnis

Das root-Verzeichnis stellt die Grundlage des Linux-Dateisystems dar. Alle weiteren Verzeichnisse und Dateien sind darin abgelegt. Abbildung 6.4 zeigt beispielsweise, dass wir hier auch das home-Verzeichnis entdecken, in dem wir alle unsere persönlichen Dateien ablegen können. Da es hierbei jedoch keine großen Unterschiede zum gewöhnlichen Windows-Dateisystem gibt, sind hierfür keine weiteren Erklärungen notwendig. Viele weitere Ordner, die hier aufgelistet sind, bieten jedoch einen direkten Zugriff auf Hardware-Elemente oder Systemfunktionen.

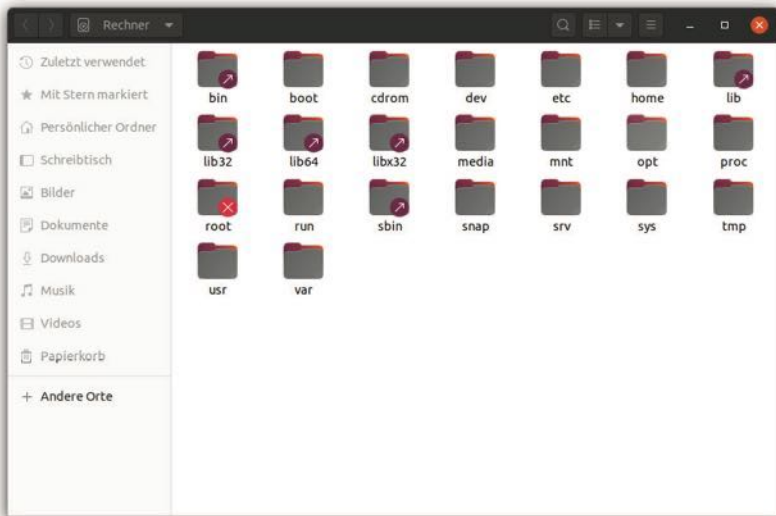


Abb. 6.4 Die Inhalte des root-Verzeichnisses

Beispieldatei aus dem Verzeichnis proc

Für unser erstes Beispiel öffnen wir nun den Ordner `proc`. Diese Abkürzung steht für „process & kernel information files“. Hier erhalten wir Zugang zu Prozessen und Elementen des Betriebssystems. Dass sich diese Informationen im Dateisystem befinden, ist eine typische Eigenschaft von Linux und anderen unixoiden Betriebssystemen.

Nachdem wir den Ordner `proc` aufgerufen haben, werden zunächst viele weitere Ordner angezeigt. Wir scrollen nun ganz nach unten. Hier sind verschiedene Dateien angegeben, die Informationen zu den Prozessen und den beteiligten Geräten enthalten. Um dies zu zeigen, klicken wir auf die Datei mit der Bezeichnung `meminfo`. Abbildung 6.5 zeigt, dass sich daraufhin in einem neuen Fenster ein Texteditor öffnet. In der Textdatei sind vielfältige Angaben zur Nutzung des Arbeitsspeichers festgehalten. Auf diese Weise ist es möglich, die Funktionen des Systems genau zu überprüfen.

```

1 MemTotal: 7716856 kB
2 MemFree: 1699232 kB
3 MemAvailable: 4432540 kB
4 Buffers: 168332 kB
5 Cached: 3027296 kB
6 SwapCached: 0 kB
7 Active: 3569000 kB
8 Inactive: 1900552 kB
9 Active(anon): 2439248 kB
10 Inactive(anon): 155332 kB
11 Active(file): 1129752 kB
12 Inactive(file): 1745220 kB
13 Unevictable: 163248 kB
14 Mlocked: 0 kB
15 SwapTotal: 5859324 kB
16 SwapFree: 5859324 kB
17 Dirty: 1036 kB
18 Writeback: 0 kB
19 AnonPages: 2437444 kB
20 Mapped: 604340 kB
21 Shmem: 320664 kB
22 KReclaimable: 167748 kB
23 Slab: 297232 kB
24 SReclaimable: 167748 kB
25 SUnreclaim: 129484 kB
26 KernelStack: 13376 kB
27 PageTables: 25420 kB
28 NFS_Unstable: 0 kB
29 Bounce: 0 kB
30 WritebackTmp: 0 kB
31 CommitLimit: 9717352 kB
32 Committed_AS: 7123320 kB
33 VmallocTotal: 34359738367 kB
34 VmallocUsed: 29236 kB
35 VmallocChunk: 0 kB
36 Percpu: 3712 kB
37 HardwareCorrupted: 0 kB

```

Abb. 6.5 Der Inhalt der Datei meminfo

Auf die gleiche Weise können wir Informationen zu weiteren Teilen des Betriebssystems abrufen. Die Datei `cpuinfo` beinhaltet beispielsweise Informationen zum verwendeten Prozessor und die Datei `devices` listet die verfügbaren Geräte auf. Allerdings ist es wichtig, darauf zu achten, dass es sich hierbei nicht um eine Textdatei handelt. Die entsprechende Datei symbolisiert dabei jeweils die zugehörige Ressource. Linux stellt diese für uns lediglich als Text dar, sodass es möglich ist, ihre Werte zu kontrollieren.

Nun kehren wir nochmals zum `root`-Verzeichnis zurück. Die Ordner `proc` und `home` haben wir bereits angesprochen. Darüber hinaus enthält dieser Ordner jedoch noch viele weitere Unterverzeichnisse, die ebenfalls von großer Bedeutung sind.

dev

Das Verzeichnis dev (für devices) enthält Dateien für alle Geräte, die auf dem Rechner zur Verfügung stehen. Auf diese Weise ist beispielsweise ein Zugriff auf das DVD-Laufwerk und auf viele weitere Hardware-Bestandteile des Computers möglich. Darüber hinaus befinden sich hier einige Dateien, die kein physisches Gerät darstellen. Ein Beispiel hierfür ist die Datei random. Hierbei handelt es sich um einen Zufallszahlen-Generator. Wenn ein Programm eine Zufallszahl benötigt, kann es diese Datei nutzen.

bin

Das Verzeichnis bin steht für „essential user command binaries“. Hier sind ausführbare Programme abgelegt, die die grundlegenden Funktionen des Systems sicherstellen. Beispielsweise ist hier für die einzelnen Shell-Befehle, die wir später noch kennenlernen werden, jeweils eine Datei vorhanden.

sbin

Das Verzeichnis sbin enthält genau wie das Verzeichnis bin grundlegende Programme für die Ausführung des Systems. Der Unterschied besteht lediglich darin, dass hier die Befehle abgelegt sind, die für die Systemadministration zum Einsatz kommen und die gewöhnliche Anwender nicht ausführen.

etc

Im Ordner etc befinden sich die Dateien für die Konfiguration des Systems. Auf diese Weise ist es möglich, die bisherigen Einstellungen zu verändern. Dafür ist es lediglich notwendig, die entsprechenden Inhalte als Textdatei aufzurufen und sie anzupassen.

lib

Das Verzeichnis lib ist ebenfalls von großer Bedeutung für die Funktion des Betriebssystems. Diese Abkürzung steht für „library files“. Demnach sind hier Bibliotheken abgespeichert. Dabei handelt es sich um Programmbibliotheken, die verschiedene Funktionen des Systems bereitstellen.

Weitere Ordner im root-Verzeichnis

6

Neben den vorgestellten Beispielen sind im root-Verzeichnis noch viele weitere Ordner aufgelistet. Die meisten von ihnen sind von großer Bedeutung für die fundamentalen Funktionen des Systems. Daran wird deutlich, welche Bedeutung das Dateisystem unter Linux hat. Es bietet Zugriff auf alle Hardware-Elemente und auf die grundlegenden Funktionen des Betriebssystems.

6.2 Programme und Prozesse

Bevor wir die verschiedenen Programme vorstellen, die das Linux-Betriebssystem mit sich bringt, befassen wir uns kurz damit, wie diese auf dem Betriebssystem ablaufen und auf welche Weise wir sie steuern können. Hierfür ist es notwendig, sich mit Prozessen zu befassen. Jedes Programm, das Sie auf dem Computer ausführen, erzeugt einen Prozess. Dieser ist für die Ausführung notwendig. Eine der zentralen Aufgaben eines Betriebssystems besteht darin, alle Prozesse, die auf dem Computer ablaufen, zu organisieren.

Was ist ein Prozess?

Ein Computerprogramm besteht aus einer Aneinanderreihung von Befehlen. Wenn es sich dabei um ein ausführbares Programm handelt, ist dieses bereits in Maschinensprache verfasst. Das bedeutet, dass es möglich ist, die entsprechenden Befehle ohne weitere Bearbeitung an den Prozessor zu übermitteln, sodass er diese ausführen kann.

Allerdings befindet sich das *Programm* zunächst auf der Festplatte und wird nicht ausgeführt. Hierfür ist ein Prozess erforderlich. Dabei handelt es sich um eine Instanz des entsprechenden Programms, die aktuell ausgeführt wird. Dazu ist es zunächst notwendig, die zugehörigen Befehle in den Arbeitsspeicher zu laden. Von hier aus werden sie dann einzeln an den Prozessor übermittelt und auf diese Weise abgearbeitet. Eine alternative Bezeichnung für einen Prozess ist Task.

Wenn ein *Prozess* ausgeführt wird, benötigt dieser Ressourcen des Computers. Die beiden wichtigsten von ihnen wurden gerade bereits erwähnt: Arbeitsspeicher und Prozessorleistung. Ohne diese beiden Bestandteile lässt sich kein Prozess ausführen. Wie bereits beschrieben, muss der Prozess zunächst in den Arbeitsspeicher geladen werden und für die Ausführung ist es notwendig, die Befehle an den Prozessor zu übermitteln. Allerdings kommen in vielen Fällen noch weitere Ressourcen hinzu. Beispielsweise nutzen viele Prozesse die Festplatte oder Peripheriegeräte wie eine Kamera oder einen Zugang zu einem Netzwerk.

Wenn Sie den Computer anschalten, wird dabei zunächst ein Prozess gestartet. Dieser ruft dann die grundlegenden Funktionen des Betriebssystems auf. Diese sind mit vielen weiteren Prozessen verbunden. Diese laufen jedoch im Hintergrund ab, sodass Sie diese als Anwender in der Regel nicht wahrnehmen. Sobald Sie ein Programm starten, entsteht dafür ein neuer Prozess, der zu den bisherigen Prozessen hinzugefügt wird.

Für die Verwaltung der Prozesse erstellt Linux für jeden einzelnen von ihnen eine sogenannte `task_struct`-Datenstruktur. Diese enthält zahlreiche Informationen zum Prozess. Von großer Bedeutung ist beispielsweise der Identifier. Dieser muss einzigartig sein und dient dazu, jeden Prozess eindeutig zu identifizieren. Darüber hinaus ist hier der Status festgehalten – beispielsweise, ob der Prozess gerade ausgeführt wird, ob er sich im Wartezustand befindet oder ob er angehalten wurde.

Von großer Bedeutung ist außerdem die sogenannte Scheduling-Information. Hier werden Informationen gesammelt, anhand derer das Betriebssystem entscheiden kann, wie wichtig die Ausführung des ent-

sprechenden Prozesses ist. Das hat erhebliche Auswirkungen auf den Ablauf der Programme, da in der Regel mehrere verschiedene Prozesse gleichzeitig den Anspruch erheben, ausgeführt zu werden. Pro Prozessor kann zu einem bestimmten Zeitpunkt jedoch nur ein einziger Prozess ablaufen. Auch die Summe des eingeforderten Arbeitsspeichers ist in vielen Fällen höher als der verfügbare Speicherbereich. Deshalb muss das Betriebssystem entscheiden, welchen Prozess es zuerst bearbeitet. Für die Entscheidungsfindung nutzt es die Scheduling Information. Das Ergebnis dieser Steuerung ist von großer Bedeutung für die Performance des gesamten Systems.

Die `task_struct`-Datenstruktur für einen einzelnen Prozess wird dann in den sogenannten task-Vektor überführt. Dieser sammelt alle Prozesse, die aktuell auf dem System aktiv sind.

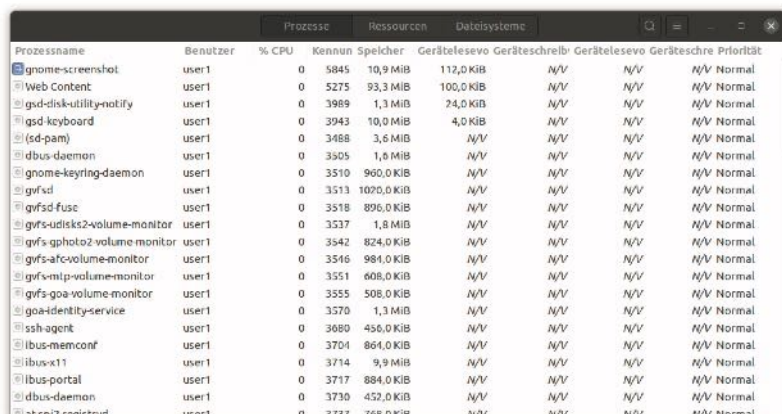
Die Steuerung von Prozessen mit der Systemüberwachung

In der Regel müssen wir uns nicht um die Steuerung der einzelnen Prozesse kümmern. Diese Aufgabe übernimmt das Betriebssystem automatisch. Dennoch können Situationen auftreten, in denen dies notwendig ist. Beispielsweise kann es hierbei zu einer Blockade kommen. Ein Fehler bei der Ausführung kann dazu führen, dass ein Prozess nicht beendet wird, die entsprechenden Ressourcen aber auch nicht mehr freigibt. Das kann dann zu einer Blockade weitreichender Systemfunktionen führen. In diesem Fall ist es sinnvoll, den Prozess manuell zu beenden. Außerdem kann es vorkommen, dass das System sehr langsam abläuft. Ursache hierfür kann es sein, dass ein Prozess sehr umfangreiche Ressourcen benötigt. In diesem Fall ist es sinnvoll, die Auslastung zu kontrollieren und das entsprechende Programm gegebenenfalls zu beenden.

Zu diesem Zweck ist es notwendig, auf die `task_struct`-Inhalte zuzugreifen, die im task-Vektor aufgeführt sind. Ein direkter Zugang ist hierbei jedoch nur schwierig umzusetzen. Aus diesem Grund verwenden wir hierfür ein spezielles Programm. Dabei stehen verschiedene Möglichkeiten zur Auswahl. Sehr beliebt ist unter Linux-Nutzern beispielsweise

6 Die grundlegenden Funktionen von Linux

das Programm htop. Dieses läuft jedoch komplett über die Konsole ab. Da wir uns mit der Nutzung der Konsole erst später befassen, wählen wir daher eine andere Alternative. Ubuntu bietet uns hierfür ein Programm, das in der deutschen Ausführung den Titel Systemüberwachung trägt. In anderen Distributionen sind auch die Bezeichnungen Task Manager und System Monitor üblich. Dieses bietet eine grafische Benutzeroberfläche mit intuitiver Bedienung und erinnert viele Anwender sicherlich an den Task-Manager unter Windows. Um dieses Programm aufzurufen, rufen wir über die neun Quadrate am linken unteren Bildrand das Start-Menü auf. Hier können wir nun die einzelnen Icons nach diesem Programm durchsuchen oder einfach den Begriff „Systemüberwachung“ in das Suchfeld eingeben. Daraufhin erscheint das Fenster, das in Abbildung 6.6 zu sehen ist.



Prozessname	Benutzer	% CPU	Kennzahl	Speicher	Gerätelesev	Geräteschreibv	Gerätelesev	Geräteschreibv	Priorität
gnome-screenshot	user1	0	5845	10,9 MiB	112,0 KiB	N/V	N/V	N/V	Normal
Web Content	user1	0	5275	93,3 MiB	100,0 KiB	N/V	N/V	N/V	Normal
gud-diskutility-notify	user1	0	3989	1,3 MiB	24,0 KiB	N/V	N/V	N/V	Normal
gud-keyboard	user1	0	3943	10,0 MiB	4,0 KiB	N/V	N/V	N/V	Normal
(sd-pam)	user1	0	3488	3,6 MiB	N/V	N/V	N/V	N/V	Normal
dbus-daemon	user1	0	3505	1,8 MiB	N/V	N/V	N/V	N/V	Normal
gnome-keyring-daemon	user1	0	3510	960,0 KiB	N/V	N/V	N/V	N/V	Normal
gvfsd	user1	0	3513	1020,0 KiB	N/V	N/V	N/V	N/V	Normal
gvfsd-fuse	user1	0	3518	896,0 KiB	N/V	N/V	N/V	N/V	Normal
gvfs-gdlsk2-volume-monitor	user1	0	3537	1,8 MiB	N/V	N/V	N/V	N/V	Normal
gvfs-gphoto2-volume-monitor	user1	0	3542	824,0 KiB	N/V	N/V	N/V	N/V	Normal
gvfs-afc-volume-monitor	user1	0	3546	984,0 KiB	N/V	N/V	N/V	N/V	Normal
gvfs-mtp-volume-monitor	user1	0	3551	608,0 KiB	N/V	N/V	N/V	N/V	Normal
gvfs-goa-volume-monitor	user1	0	3555	508,0 KiB	N/V	N/V	N/V	N/V	Normal
goa-identity-service	user1	0	3570	1,3 MiB	N/V	N/V	N/V	N/V	Normal
ssh-agent	user1	0	3680	456,0 KiB	N/V	N/V	N/V	N/V	Normal
ibus-memconf	user1	0	3701	864,0 KiB	N/V	N/V	N/V	N/V	Normal
ibus-x11	user1	0	3714	9,9 MiB	N/V	N/V	N/V	N/V	Normal
ibus-portal	user1	0	3717	884,0 KiB	N/V	N/V	N/V	N/V	Normal
dbus-daemon	user1	0	3730	452,0 KiB	N/V	N/V	N/V	N/V	Normal
atspi2-registrard	user1	0	3737	768,0 KiB	N/V	N/V	N/V	N/V	Normal

Abb. 6.6 Die Systemüberwachung unter Linux

Hier erkennen wir nun vielfältige Informationen. Beispielsweise ist der Name des Prozesses angegeben und der Anwender, der ihn aufgerufen hat. Darüber hinaus sind wichtige Daten wie die Nutzung der CPU und des Speichers angegeben. Außerdem ist die Zahl der Lese- und Schreibvorgänge aufgeführt. Darüber hinaus erkennen Sie die Priorität, die der jeweilige Prozess aufweist.

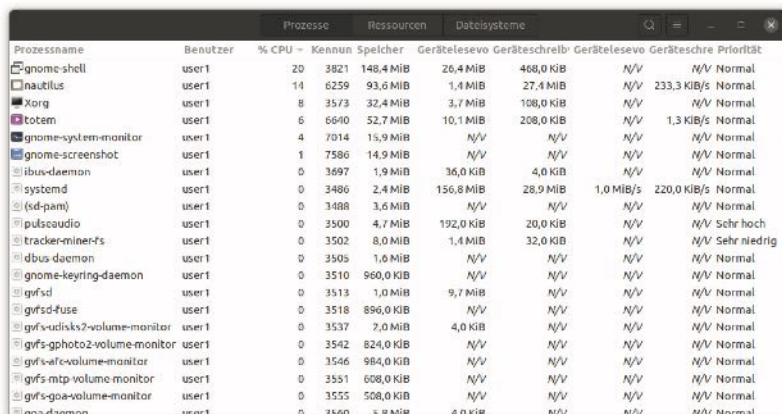
Hierbei haben Sie auch die Möglichkeit, die Liste nach Ihren Bedürfnissen zu ordnen. Wenn Sie beispielsweise herausfinden möchten, welches Programm die CPU am stärksten nutzt, können Sie auf das Feld mit der Aufschrift „% CPU“ klicken. Daraufhin werden die Prozesse entsprechend ihrer CPU-Nutzung geordnet. Das zeigt Abbildung 6.7.

Prozessname	Benutzer	% CPU	Kennen	Speicher	Gerätelesew	Geräteschreib	Gerätelesew	Geräteschreib	Priorität
gnome-shell	user1	8	3821	145,8 MiB	16,9 MiB	164,0 KiB	N/V	N/V	Normal
Xorg	user1	1	3573	31,3 MiB	3,7 MiB	104,0 KiB	N/V	N/V	Normal
gnome-system-monitor	user1	1	5857	15,9 MiB	N/V	N/V	N/V	N/V	Normal
gnome-screenshot	user1	1	5990	10,9 MiB	N/V	N/V	N/V	N/V	Normal
firefox	user1	0	4172	381,0 MiB	127,8 MiB	289,8 MiB	N/V	N/V	Normal
ibus-extension-gtk3	user1	0	3708	13,5 MiB	744,0 KiB	N/V	N/V	N/V	Normal
systemd	user1	0	3486	2,4 MiB	11,5 MiB	2,5 MiB	N/V	N/V	Normal
ibus-dnssd	user1	0	5908	960,0 KiB	40,0 KiB	N/V	N/V	N/V	Normal
gvfsd-network	user1	0	5894	1,1 MiB	40,0 KiB	N/V	N/V	N/V	Normal
Web Content	user1	0	5275	94,3 MiB	100,0 KiB	N/V	N/V	N/V	Normal
gsd-disk-utility-notify	user1	0	3989	1,3 MiB	24,0 KiB	N/V	N/V	N/V	Normal
gsd-keyboard	user1	0	3943	10,0 MiB	4,0 KiB	N/V	N/V	N/V	Normal
(sd-pam)	user1	0	3488	3,6 MiB	N/V	N/V	N/V	N/V	Normal
dbus-daemon	user1	0	3505	1,6 MiB	N/V	N/V	N/V	N/V	Normal
gnome-keyring-daemon	user1	0	3510	960,0 KiB	N/V	N/V	N/V	N/V	Normal
gvfsd-fuse	user1	0	3518	896,0 KiB	N/V	N/V	N/V	N/V	Normal
gvfs-udisks2-volume-monitor	user1	0	3537	1,8 MiB	N/V	N/V	N/V	N/V	Normal
gvfs-gphoto2-volume-monitor	user1	0	3542	824,0 KiB	N/V	N/V	N/V	N/V	Normal
gvfs-afc-volume-monitor	user1	0	3546	984,0 KiB	N/V	N/V	N/V	N/V	Normal
gvfs-mtp-volume-monitor	user1	0	3551	608,0 KiB	N/V	N/V	N/V	N/V	Normal
gvfs-usb-volume-monitor	user1	0	3555	508,0 KiB	N/V	N/V	N/V	N/V	Normal

Abb. 6.7 Die Prozesse nach ihrer CPU-Nutzung geordnet

Nun können Sie auch einmal ein weiteres Programm öffnen. Für unser Beispiel spielen wir ein privates Video ab. Dabei wird automatisch der Video-Player geöffnet, der in Ubuntu die Bezeichnung *totem* trägt. Abbildung 6.8 zeigt, dass dieser nun die CPU beansprucht.

6 Die grundlegenden Funktionen von Linux



Prozessname	Benutzer	% CPU	Kennzahl	Speicher	Gerätelesew	Geräteschreib	Gerätelesew	Geräteschreib	Priorität
gnome-shell	user1	20	3821	148,4 MiB	26,4 MiB	468,0 KiB	N/V	N/V	Normal
nautilus	user1	14	6259	93,6 MiB	1,4 MiB	27,4 MiB	N/V	233,3 KiB/s	Normal
xorg	user1	8	3573	32,4 MiB	3,7 MiB	108,0 KiB	N/V	N/V	Normal
totem	user1	6	6640	52,7 MiB	10,1 MiB	208,0 KiB	N/V	1,3 KiB/s	Normal
gnome-system-monitor	user1	4	7014	15,9 MiB	N/V	N/V	N/V	N/V	Normal
gnome-screenshot	user1	1	7586	14,9 MiB	N/V	N/V	N/V	N/V	Normal
ibus-daemon	user1	0	3697	1,9 MiB	36,0 KiB	4,0 KiB	N/V	N/V	Normal
systemd	user1	0	3486	2,4 MiB	156,8 MiB	28,9 MiB	1,0 MiB/s	220,0 KiB/s	Normal
(sd-pam)	user1	0	3488	3,6 MiB	N/V	N/V	N/V	N/V	Normal
pulseaudio	user1	0	3500	4,7 MiB	192,0 KiB	20,0 KiB	N/V	N/V	Sehr hoch
tracker-miner-fs	user1	0	3502	8,0 MiB	1,4 MiB	32,0 KiB	N/V	N/V	Sehr niedrig
dbus-daemon	user1	0	3505	1,0 MiB	N/V	N/V	N/V	N/V	Normal
gnome-keyring-daemon	user1	0	3510	960,0 KiB	N/V	N/V	N/V	N/V	Normal
gvfsd	user1	0	3513	1,0 MiB	9,7 MiB	N/V	N/V	N/V	Normal
gvfsd-fuse	user1	0	3518	896,0 KiB	N/V	N/V	N/V	N/V	Normal
gvfs-udisks2-volume-monitor	user1	0	3537	7,0 MiB	4,0 KiB	N/V	N/V	N/V	Normal
gvfs-gphoto2-volume-monitor	user1	0	3542	824,0 KiB	N/V	N/V	N/V	N/V	Normal
gvfs-afc-volume-monitor	user1	0	3546	964,0 KiB	N/V	N/V	N/V	N/V	Normal
gvfs-mtp-volume-monitor	user1	0	3551	608,0 KiB	N/V	N/V	N/V	N/V	Normal
gvfs-goa-volume-monitor	user1	0	3555	508,0 KiB	N/V	N/V	N/V	N/V	Normal
nautilus	user1	0	3560	5,8 MiB	4,0 KiB	N/V	N/V	N/V	Normal

Abb. 6.8 Das aufgerufene Programm nimmt Ressourcen in Anspruch

Abschließend befassen wir uns damit, wie wir die Ausführung eines Prozesses in Linux beeinflussen können. Wenn Sie einen Prozess lediglich beenden möchten, müssen Sie ihn zunächst anklicken. Daraufhin wird er orange hinterlegt und im unteren Bildbereich erscheint eine Schaltfläche mit der Aufschrift „Prozess beenden“. Es bestehen aber noch vielfältige weitere Möglichkeiten. Hierfür ist es notwendig, den Prozess mit der rechten Maustaste anzuklicken. Daraufhin erscheint ein Kontextmenü mit verschiedenen Auswahlmöglichkeiten. Hierbei können Sie nicht nur die Eigenschaften und die Bereiche des Arbeitsspeichers überprüfen, die dieser belegt. Darüber hinaus besteht die Möglichkeit, den Prozess anzuhalten, fortzusetzen, zu beenden oder abzuwürgen. Diese letzte Möglichkeit ist immer dann hilfreich, wenn es zu einer Blockade gekommen ist. Eine hilfreiche Steuerungsmöglichkeit besteht auch darin, die Priorität zu verändern. Bei einer hohen Auslastung des Systems können Sie auf diese Weise selbst entscheiden, die Ausführung welcher Programme für Sie von besonderer Bedeutung ist. Abbildung 6.9 zeigt die entsprechenden Auswahlmöglichkeiten.

Prozessname	Benutzer	% CPU	Kennung	Speicher	Gerätelesevo	Geräteschreib	Gerätelesevo	Geräteschreib	Priorität
gnome-shell	user1	33	3821	149,9 MiB	26,4 MiB	1,5 MiB	N/V	N/V	Normal
gnome-system-monitor	user1	33	9128	16,8 MiB	224,0 KiB	8,0 KiB	N/V	N/V	Normal
Firefox	user1	0	4172	436,6 MiB	129,1 MiB	346,1 MiB	N/V	N/V	Normal
Web Content	user1	0	4431	663,5 MiB	26,2 MiB	N/V	N/V	N/V	Normal
totem	user1	0	10010	52,0 MiB	4,3 MiB	316,0 KiB	N/V	N/V	Normal
ibus-e	Eigenschaften	Alt+Eingabe	0	3750	860,0 KiB	N/V	N/V	N/V	Normal
pulse	Speicherfelder	Strg+M	0	3500	4,7 MiB	192,0 KiB	20,0 KiB	N/V	N/V Sehr hoch
system	Offene Dateien	Strg+O	0	3486	2,4 MiB	869,7 MiB	103,2 MiB	N/V	N/V Normal
sd-pa	Priorität ändern								
tracke	Anhalten	Strg+S							
dbus-	Fortsetzen	Strg+C							
gnom	Beenden	Strg+E							
gvfsd	Abwürgen	Strg+K							
gvfsd									
gvfs-udisks2-volume-monitor	user1		0	3542	824,0 KiB	N/V	N/V	N/V	Normal
gvfs-gphoto2-volume-monitor	user1		0	3546	984,0 KiB	N/V	N/V	N/V	Normal
gvfs-afc-volume-monitor	user1		0	3551	608,0 KiB	N/V	N/V	N/V	Normal
gvfs-mtp-volume-monitor	user1		0	3555	508,0 KiB	N/V	N/V	N/V	Normal
gvfs-boa-volume-monitor	user1		0	3555	508,0 KiB	N/V	N/V	N/V	Normal

Abb. 6.9 Verschiedene Möglichkeiten für die Steuerung eines Prozesses

6.3 Textverarbeitung und weitere Office-Programme

Die Erledigung von Büroarbeiten zählt zu den häufigsten Tätigkeiten, die die Anwender am Computer erledigen. Hierfür ist selbstverständlich eine geeignete Software notwendig. Der eindeutige Marktführer in diesem Bereich ist Microsoft mit dem Produkt Office. Wenn Sie nun jedoch auf Linux umsteigen, ergibt sich das Problem, dass Microsoft für dieses Betriebssystem keine passende Version anbietet. Microsoft Office ist nur für Windows und für macOS erhältlich. Die einzige Möglichkeit, die Office-Programme von Microsoft auch unter Linux zu verwenden, besteht in der Nutzung der Web-Apps von Microsoft 365. Allerdings gibt es zahlreiche Situationen, in denen diese Cloud-Version keine optimale Lösung darstellt. Beispielsweise ist für die Nutzung ein Internetanschluss notwendig. Ist dieser nur sehr langsam oder überhaupt nicht vorhanden, ist es nicht möglich, auf die Programme zuzugreifen. Darüber hinaus ist diese Software nur im Abo-Modell erhältlich. Viele Anwender wünschen jedoch keine laufenden Kosten für die Nutzung der Office-Programme. Schließlich sind die Funktionen bei den Web-Apps nicht genau die gleichen wie bei der fest installierten Ausführung. Auch das schreckt viele Nutzer ab.

Doch auch für Anwender, die Online-Version von Microsoft Office ablehnen, steht unter Linux eine passende Alternative bereits. Fast alle Distributionen verfügen bereits über ein vorinstalliertes Office-Paket. Bei Ubuntu handelt es sich hierbei um LibreOffice. Andere Versionen verwenden OpenOffice oder FreeOffice. All diese Office-Suites weisen die Gemeinsamkeit auf, dass es sich hierbei der Linux-Philosophie entsprechend um kostenfreie Software handelt. Allerdings stellt die letztgenannte Alternative eine proprietäre Software dar, während es sich bei den beiden anderen Möglichkeiten um Open-Source-Programme handelt.

Für die Anwender bringt dies den Vorteil mit sich, dass sie keine Nutzungsgebühren für die Verwendung der Software entrichten müssen. Allerdings weisen diese Office-Programme einige Unterschiede zu den Angeboten von Microsoft auf. Das kann eine gewisse Einarbeitung notwendig machen. Zwar ist die Verwendung auch hierbei intuitiv und einfach. Dennoch soll dieser Abschnitt die Verwendung kurz vorstellen.

Das Textverarbeitungsprogramm LibreOffice Writer

Besonders beliebt ist die Verwendung von Textverarbeitungsprogrammen. Damit ist es möglich, Briefe zu verfassen oder Einladungen zu privaten Feiern aufzusetzen. Auch im Bildungsbereich kommt diese Software häufig zum Einsatz – beispielsweise für Hausarbeiten oder zur Gestaltung eines Handouts für ein Referat. Als Alternative zu Microsoft Word können Sie unter Linux den LibreOffice Writer verwenden. Da es sich hierbei um eines der am häufigsten genutzten Office-Programme handelt, ist das entsprechende Symbol direkt in die Favoritenleiste am linken Bildrand integriert. Abbildung 6.10 zeigt dies. Ein Klick auf diese Schaltfläche reicht aus, um das Programm zu öffnen.



Abb. 6.10 Das Symbol für das Programm LibreOffice Writer

Wenn Sie das Programm öffnen, erscheint zunächst ein kleines Fenster mit Tipps zum Umgang mit dieser Software. Ein Beispiel hierfür ist in Abbildung 6.11 zu sehen. Hierbei handelt es sich jedoch nicht um eine allgemeine Einführung in die grundlegenden Funktionen, sondern um Hinweise zu sehr speziellen Anwendungen. Wenn Sie Interesse daran haben, können Sie sich durch viele verschiedene Tipps klicken. Es ist jedoch auch möglich, das Fenster zu schließen und mit der Anwendung des Programms zu beginnen. Wenn Sie zuvor die Checkbox im Fenster deaktivieren, sorgen Sie dafür, dass die Tipps beim nächsten Start der Software nicht mehr angezeigt werden.



Abb. 6.11 Ein Beispiel für die Tipps beim Start der Software

Wenn Sie das Fenster mit den Tipps geschlossen haben, gelangen Sie zur eigentlichen Oberfläche der Software. Diese ist sehr ähnlich aufgebaut wie bei Microsoft Word. In der großen Freifläche im Zentrum des Fensters können Sie nun einen beliebigen Text einfügen. Die grundlegenden Funktionen weisen ebenfalls eine große Ähnlichkeit zu Word auf. Beispielsweise können Sie bestimmte Bereiche als Titel, als Unterüberschrift oder als einfachen Text markieren. Es besteht die Möglichkeit, eine passende Schriftart auszuwählen und die Schriftgröße vorzugeben, Textbereiche fett zu markieren, kursiv zu setzen oder zu unterstreichen und eine passende Hintergrundfarbe auszuwählen. Hinzu kommen vielfältige weitere Gestaltungsmöglichkeiten. Außerdem lassen sich weitere Elemente wie Listen oder Tabellen einfügen. An dieser Stelle wird bereits deutlich, dass der LibreOffice Writer äh-

6 Die grundlegenden Funktionen von Linux

ßerst vielfältige Funktionen anbietet und daher eine gute kostenfreie Alternative zu Microsoft Word darstellt. Abbildung 6.12 zeigt einige der Möglichkeiten für die Textgestaltung mit diesem Programm.

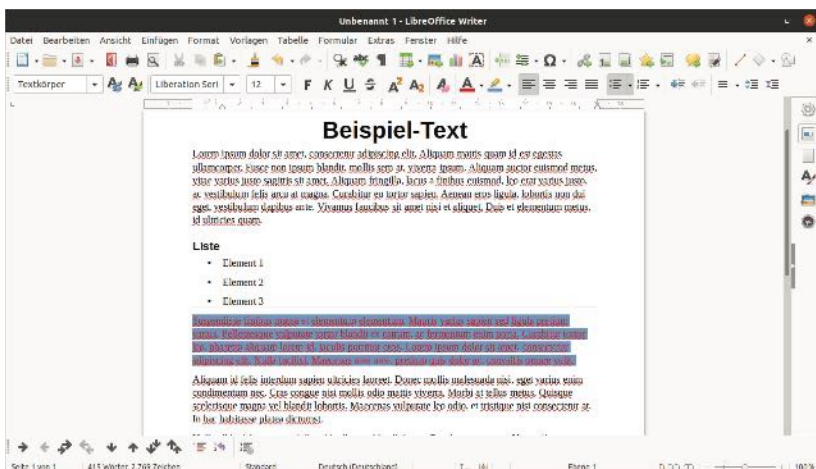


Abb. 6.12 Die Textgestaltung mit dem LibreOffice Writer.

An dieser Stelle ist es nicht möglich, alle Funktionen der Software detailliert zu beschreiben. Aufgrund der intuitiven Bedienung ist dies jedoch auch nicht notwendig. Wenn Sie dennoch etwas Hilfestellung für die Verwendung benötigen, können Sie in der Menüleiste den Begriff „Hilfe“ und anschließend „LibreOffice Hilfe“ anklicken. Daraufhin öffnet sich im Browser die Hilfe-Seite. Hier ist unter anderem ein Link mit der Bezeichnung „Anleitungen für LibreOffice Writer“ vorhanden. Dort entdecken Sie eine detaillierte Anleitung in deutscher Sprache.

Abschließend wollen wir noch einen kurzen Blick auf das verwendete Dateiformat werfen. Wenn Sie auf „Speichern“ klicken, öffnet sich ein Fenster, in dem Sie wie üblich den Namen für das Dokument eingeben können. Unten rechts im Fenster erkennen Sie nun jedoch, dass hierbei als Dateiformat „ODF-Textdokument (.odt)“ in der Vorauswahl erscheint. Hierbei handelt es sich um das Standard-Format für Dokumente im LibreOffice Writer. Diese Auswahl ist in Abbildung 6.13 zu sehen. Wenn Sie Ihre Dokumente ausschließlich mit dieser Software

bearbeiten, stellt dieses Format die richtige Wahl dar. Sollten Sie sie jedoch auch mit Microsoft Word bearbeiten, können Sie das Dateiformat rechts unten im Fenster anklicken. Daraufhin erscheint eine Auswahlliste, die auch die typischen Word-Formate enthält.



Abb. 6.13 Das Speichern der Datei

Das Tabellenkalkulationsprogramm LibreOffice Calc

Auch Tabellenkalkulationsprogramme kommen sehr häufig zum Einsatz. Insbesondere in vielen Büros dient diese Software dazu, um verschiedene Daten zu Kunden, Mitarbeitern und Produkten zu erfassen. Im privaten Umfeld ist die Verwendung zwar etwas seltener. Doch gibt es auch hierbei zahlreiche Anwendungsmöglichkeiten. Häufig kommt zu diesem Zweck Microsoft Excel zum Einsatz. Als Alternative hierzu können Sie unter Linux LibreOffice Calc verwenden.

Um diese Software zu öffnen, müssen Sie das Start-Menü öffnen. Nun können Sie das Suchfeld nutzen und den Begriff „Calc“ eingeben. Eine weitere Möglichkeit besteht darin, im unteren Bereich des Menüs auf

6 Die grundlegenden Funktionen von Linux

„Alle“ zu klicken und anschließend das Symbol zu suchen, das in Abbildung 6.14 zu sehen ist.



Abb. 6.14 Das Symbol für LibreOffice Calc

Nachdem Sie das Programm geöffnet haben, erscheint die typische Oberfläche eines Tabellenkalkulationsprogramms mit vielen einzelnen Zellen. Wenn Sie bereits mit Microsoft Excel vertraut sind, sollte die Verwendung von LibreOffice Calc kein großes Problem darstellen. Die grundlegenden Befehle und Steuerungsmechanismen sind dabei genau die gleichen. Sie können die Werte in die Zellen eintragen und mithilfe verschiedener Formeln Berechnungen damit durchführen. Abbildung 6.15 zeigt, dass es auch mit dieser Software möglich ist, die Daten grafisch aufzubereiten – beispielsweise durch ein Diagramm.

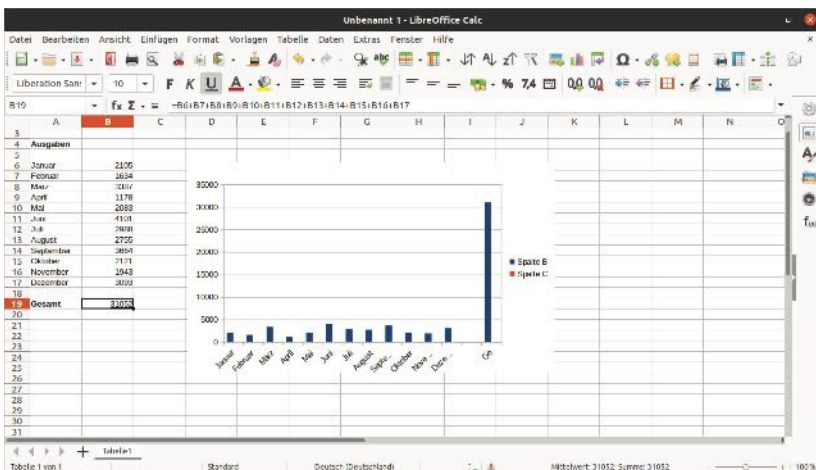


Abb. 6.15 Die Verwendung von LibreOffice Calc

Auch hierbei ist es nicht möglich, alle Funktionen der Software detailliert zu beschreiben. Sollten Sie eine ausführliche Anleitung wünschen, besteht wieder die Möglichkeit, über die Menüleiste die Hilfe zu öffnen. Auf diese Weise können Sie sich mit der Software optimal vertraut machen. Auch für die Tabellenkalkulation verwendet LibreOffice ein eigenes Format, um die Daten zu speichern. Bei Bedarf können Sie die Datei jedoch auch in den typischen Excel-Formaten abspeichern.

Weitere Office-Programme

6

Bei den Programmen für die Textverarbeitung und für die Tabellenkalkulation handelt es sich um die wichtigsten Office-Anwendungen. Es bestehen jedoch noch zahlreiche weitere Möglichkeiten. Von großer Bedeutung ist beispielsweise LibreOffice Impress. Dieses Programm dient dem Erstellen von Präsentationen – ähnlich wie Microsoft PowerPoint. Außerdem ist LibreOffice Draw unter Ubuntu vorinstalliert. Hierbei handelt es sich um ein einfaches Zeichenprogramm. Darüber hinaus bietet LibreOffice noch weitere Möglichkeiten: LibreOffice Base und LibreOffice Math. Beim Programm Base handelt es sich um eine Datenbankanwendung – ähnlich wie Microsoft Access. Mit Math gestalten sie hingegen mathematische Formeln. Diese Programme sind zwar unter Ubuntu nicht vorinstalliert. Es ist jedoch problemlos möglich, diese hinzuzufügen. Ubuntu bietet hierfür ein Installationsprogramm mit grafischer Oberfläche an: das Ubuntu Software Center. Dieses finden Sie über das Start-Menü. Wenn Sie dort die entsprechenden Suchbegriffe eingeben, können Sie die gewünschten Programme herunterladen und installieren. Im folgenden Kapitel lernen Sie dann auch, wie Sie diese Aufgabe über die Konsole erledigen.

Ein weiterer wichtiger Anwendungsbereich von Office-Programmen ist die Bearbeitung von E-Mails. Die Microsoft-Software, die hierfür zum Einsatz kommt, trägt den Namen Outlook. Unter Ubuntu ist zu diesem Zweck Mozilla Thunderbird installiert. Diese Software gehört zwar nicht zu LibreOffice, sie rundet in Ergänzung dazu jedoch die Office-Suite ab. Das zeigt, dass Ihnen auch unter Linux für alle Aufgaben im

Büro hochwertige Software zur Verfügung steht – und das vollkommen kostenfrei.

6.4 Drucker anschließen und Treiber installieren

Der Drucker stellt eines der wichtigsten Peripheriegeräte dar. Er kommt sowohl im Büro als auch im Privathaushalt häufig zum Einsatz. Dieses Gerät benötigt jedoch einen Treiber. Dabei handelt es sich um eine Software, die den Druckbefehl an die speziellen Anforderungen des entsprechenden Geräts anpasst. Ein Treiber muss jedoch stets auf das verwendete Betriebssystem ausgerichtet sein. Das bedeutet, dass es unter Linux nicht möglich ist, einen Windows-Treiber zu verwenden.

Ein Problem bei der Verwendung von Linux stellte es früher dar, dass es sehr schwierig war, einen Drucker zu installieren. Der Grund dafür bestand darin, dass für zahlreiche Geräte kein passender Treiber verfügbar war. Insbesondere in der Anfangszeit boten nur sehr wenige Hersteller eine entsprechende Software an.

Mittlerweile stellt dies jedoch in der Regel kein Problem mehr dar. Die meisten Hersteller bieten heutzutage auch Treiber für Linux an. Insbesondere die Unternehmen HP und Brother zeichnen sich durch eine sehr gute Unterstützung für dieses Betriebssystem aus. Doch auch die meisten anderen Hersteller halten inzwischen die benötigte Software bereit. Dennoch ist es vor dem Kauf eines neuen Druckers stets empfehlenswert, herauszufinden, ob ein Linux-Treiber für das entsprechende Modell verfügbar ist. In der Regel lässt sich diese Information problemlos über die Webseite des Herstellers einholen.

Wenn Sie einen USB-Drucker verwenden, ist die Inbetriebnahme des Geräts normalerweise ausgesprochen einfach. Sie müssen es lediglich mit dem Computer verbinden und einschalten. In der Regel öffnet sich daraufhin ein kleines Fenster, das anzeigt, dass der Treiber installiert wird. Nach einer Wartezeit von wenigen Sekunden ist der Drucker dann einsatzbereit.

Es kann jedoch auch manchmal vorkommen, dass das Betriebssystem den Drucker nicht automatisch installiert. Das ist zum einen bei Netzwerk-Geräten und außerdem häufig bei älteren Druckern der Fall. In diesem Fall müssen Sie auf das Start-Menü klicken und in das Suchfeld „Drucker“ eingeben. Wie in Abbildung 6.16 zu sehen, erscheint nun im unteren Bereich unter „Einstellungen“ eine Anwendung, mit der Sie den Drucker konfigurieren können.



Abb. 6.16 Die Anwendung für die Konfiguration des Druckers suchen

Daraufhin öffnet sich die Anwendung in einem neuen Fenster. Dieses ist in Abbildung 6.17 zu sehen. Hier werden zunächst die bereits verfügbaren Drucker aufgelistet. Oben rechts im Fenster entdecken Sie einen Button mit der Aufschrift „Hinzufügen“. Diesen müssen Sie nun anklicken.

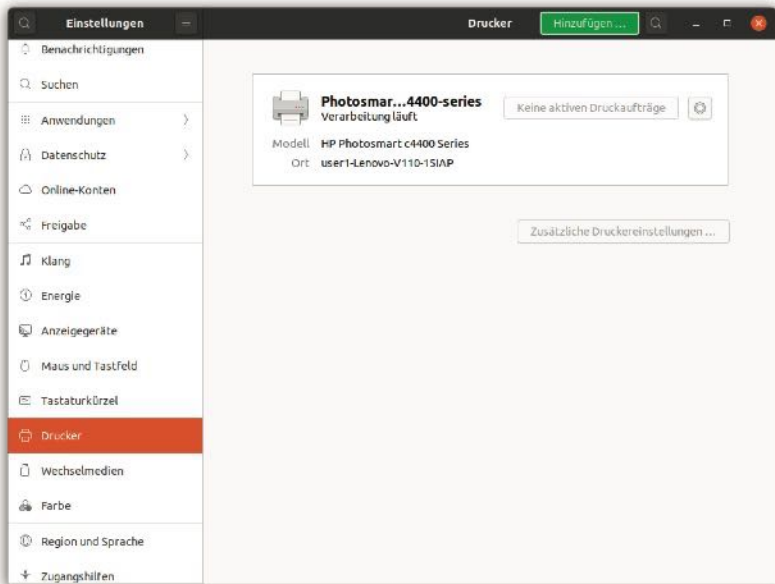


Abb. 6.17 Die Verwaltung der Drucker

Daraufhin wird ein weiteres Fenster geöffnet. Dieses dient der manuellen Verwaltung der Drucker. Wenn das Gerät hier aufgelistet wird, können sie es anklicken und mit einem Klick auf den Button „Hinzufügen“ installieren. Dies ist in Abbildung 6.18 zu erkennen.



Abb. 6.18 Einen Drucker manuell hinzufügen

Über das Drucker-Menü können Sie außerdem die Einstellungen für das Gerät vornehmen. Durch einen Klick auf das Zahnradsymbol neben dem Gerät ist es möglich, das Format vorzugeben oder auszuwählen, ob Sie in Farbe oder in Graustufen drucken möchten.

In seltenen Fällen kann es auch vorkommen, dass der Drucker hier nicht aufgelistet ist. In diesem Fall ist es sinnvoll, zunächst nochmals die Anschlüsse zu überprüfen. Sollte das nicht zum Ziel führen, besteht das Problem wahrscheinlich darin, dass kein passender Treiber zur Verfügung steht. Es besteht jedoch auch die Möglichkeit, nach weite-

ren Treibern zu suchen. Standardmäßig nutzt Ubuntu ausschließlich Open-Source-Treiber. Manche Hersteller bieten hierfür jedoch nur proprietäre Versionen an. Diese stehen aber dennoch kostenfrei zur Verfügung. Um sie in Ubuntu zu nutzen, ist es jedoch notwendig, sie gezielt zu aktivieren. Zu diesem Zweck müssen Sie in das Suchfeld des Start-Menüs den Begriff „Zusätzliche Treiber“ eingeben. Daraufhin öffnet sich das Fenster, das in Abbildung 6.19 zu sehen ist. Hier werden nun optionale Treiber für verschiedene Geräte aufgelistet. Wenn Sie hier einen Treiber für Ihren Drucker entdecken, müssen Sie diesen auswählen und aktivieren.



Abb. 6.19 Die Auswahl an zusätzlichen Treibern

Sollte auch das nicht zum Ziel führen, besteht eine weitere Option darin, den Internetauftritt des Herstellers zu besuchen. Manchmal werden die Treiber hier direkt zum Download angeboten. In der Regel ist dann auch eine Installationsanweisung vorhanden, der Sie folgen können. Ist dies nicht der Fall, lernen Sie im weiteren Verlauf dieses Buchs noch, wie Sie Programme unter Linux installieren können.

6.5 Texteditor zum Programmieren

Ein weiteres wichtiges Werkzeug ist der Texteditor. Dieser kommt in erster Linie zum Programmieren und für die Erstellung von HTML-, XML-, JSON- oder ähnlichen Dokumenten zum Einsatz. Da zahlreiche Linux-Anwender derartige Aufgaben erledigen, ist der Texteditor von großer Bedeutung.

Allerdings stellt sich für viele Leser an dieser Stelle die Frage, was ein Texteditor ist und weshalb man für die entsprechenden Aufgaben nicht einfach den LibreOffice Writer verwenden kann, der bereits in Kapitel 6.3 vorgestellt wurde.

Beim LibreOffice Writer handelt es sich um ein Textverarbeitungsprogramm. Diese Art von Software dient zwar genau wie ein Texteditor dazu, Texte zu erstellen. Allerdings liegt hier das Hauptaugenmerk auf einer ansprechenden Gestaltung. Es wurde gezeigt, dass Sie hierbei vielfältige Möglichkeiten für das Layout des Texts vorfinden. Sie können Schriftgrößen, Farben, Schriftarten und zahlreiche weitere Details vorgeben. Wenn Sie das entsprechende Dokument erstellen, ist es notwendig, all diese Informationen ebenfalls abzuspeichern. Auf diese Weise entstehen Dateien, die neben dem eigentlichen Text noch zahlreiche weitere Inhalte aufweisen.

Wenn Sie nun jedoch ein Computerprogramm erstellen oder ein Dokument in einer der genannten Auszeichnungssprachen gestalten, ist hierfür lediglich der reine Text erforderlich. Um ein Programm auszuführen, ist es notwendig, den Programmcode mit einem Compiler oder mit einem Interpreter in die Maschinensprache umzuwandeln. Diese Software kann jedoch nur den reinen Programmcode in Textform verarbeiten. Sollten hierbei zusätzliche Angaben zum Layout vorhanden sein, ist eine Ausführung nicht möglich.

Textverarbeitungsprogramme wie Word oder der LibreOffice Writer sind jedoch nicht dazu in der Lage, reine Textdateien zu erstellen. Sie speichern immer auch weitere Informationen ab. Deshalb ist es damit nicht möglich, Computerprogramme oder andere Dateien für die automatische Bearbeitung zu erstellen. Deshalb kommt für diese Aufgabe ein Texteditor zum Einsatz.

Der vorinstallierte Texteditor gedit

Ubuntu wird – wie die meisten anderen Betriebssysteme auch – mit einem vorinstallierten Texteditor ausgeliefert. Hierbei handelt es sich

um das Programm gedit. Dieses stellt einen Teil der GNOME-Desktopumgebung dar und ist in allen Distributionen enthalten, die diese nutzen. In der aktuellen Version wird diese Bezeichnung jedoch nicht mehr angegeben. Das Programm trägt hier die Bezeichnung Textbearbeitung. Dass es sich hierbei dennoch um den bekannten Texteditor gedit handelt, wird jedoch deutlich, wenn Sie das Programm aufrufen und daraufhin in der Menüleiste das Symbol mit den drei waagerechten Strichen und anschließend „Hilfe zum Texteditor“ anklicken. Das Fenster, das in Abbildung 6.20 zu sehen ist, zeigt deutlich, dass es sich bei dieser Anwendung um gedit handelt. Auch wenn Sie die Ubuntu-Suchfunktion verwenden, wird der Zusammenhang deutlich.



Abb. 6.20 Die Informationen zum Texteditor

Um gedit zu starten, müssen Sie das Startmenü aufrufen. Nun können Sie das Icon mit der Aufschrift „Textbearbeitung“ suchen. Eine andere Alternative besteht darin, die Suchfunktion zu verwenden. Hier kommen Sie sowohl mit dem Begriff „Textbearbeitung“ als auch mit dem Ausdruck „gedit“ ans Ziel.

Nun erscheint eine leere Fläche, in die Sie den Programmtext oder einen beliebigen anderen Inhalt eintragen können. Obwohl es sich bei gedit um einen recht einfachen Texteditor handelt, bietet dieser bereits deutlich mehr Funktionen als Notepad – der Standard-Texteditor unter Windows.

Eine dieser Funktionen ist die Syntaxhervorhebung. Wenn Sie ein Programm schreiben, erkennt gedit die Schlüsselbegriffe der entsprechenden Programmiersprache selbstständig. Der Editor hebt diese dann farbig hervor. Das ist in Abbildung 6.21 zu erkennen. Das führt zu einem wesentlich übersichtlicheren Programmcode. Um diese Funktion zu aktivieren, ist es jedoch notwendig, die Datei zunächst mit einer zur verwendeten Programmiersprache passenden Endung abzuspeichern.



```

1 #include <stdio.h>
2
3
4 int main () {
5     int anzahl;
6     printf("Anzahl der Worte: ");
7     scanf("%i", &anzahl);
8     int array[anzahl];
9     int zahl;
10    for (int i = 0; i < anzahl; i++) {
11        printf("Geben Sie den %i. Wert ein: ", i + 1);
12        scanf("%i", &zahl);
13        array[i] = zahl;
14    }
15    printf("L\u0084nge: %i", sizeof(array)/sizeof(array[0]));
16    return 0;
17 }

```

Abb. 6.21 Die Syntaxhervorhebung bei gedit am Beispiel eines Programms in C

Darüber hinaus bietet gedit noch einige weitere Funktionen, die beim Programmieren hilfreich sind. Beispielsweise werden zusammengehörige öffnende und schließende Klammern markiert. Außerdem behält der Texteditor die getätigten Einrückungen auch in der folgenden Zeile bei. Darüber hinaus profitieren Sie von einer praktischen Nummerierung der Zeilen.

Weitere Möglichkeiten

Mit gedit ist bereits ein Texteditor vorinstalliert, der die grundlegenden Funktionen zum Programmieren zur Verfügung stellt. Allerdings gibt es in diesem Bereich noch zahlreiche weitere Software-Produkte, die viele zusätzliche Funktionen bieten. Beispielsweise gibt es Texteditoren, die automatisch die Befehle erkennen, die Sie verwenden. Sobald Sie die ersten Buchstaben eingeben, erscheint eine Liste mit passenden Auswahlmöglichkeiten. Hier können Sie dann eine passende Option auswählen. Diese Funktion wird als Autovervollständigung bezeichnet. Viele Texteditoren erkennen hierbei auch selbst erstellte Variablen und Objekte und fügen diese ebenfalls automatisch in den Code ein. Ein weiteres Beispiel besteht darin, dass die Software geöffnete Klammern und Anführungszeichen automatisch wieder schließt. Darüber hinaus gibt es noch viele weitere praktische Funktionen. Diese erleichtern die Arbeit beim Programmieren deutlich.

Hierfür stehen mehrere weitere Texteditoren bereit, die sich problemlos installieren lassen. Hierfür fallen keine Nutzungsgebühren an. Insbesondere wenn Sie Linux verwenden möchten, um eigene Programme zu gestalten, ist es sinnvoll, einen etwas umfangreicheren Texteditor zu verwenden. Beliebte Beispiele hierfür sind Atom, Vim, SublimeText und Geany. Abbildung 6.22 zeigt als Beispiel hierfür die Darstellung einer Programmdatei im Texteditor Atom. Dabei ist auch die Autovervollständigungs-Funktion zu erkennen.

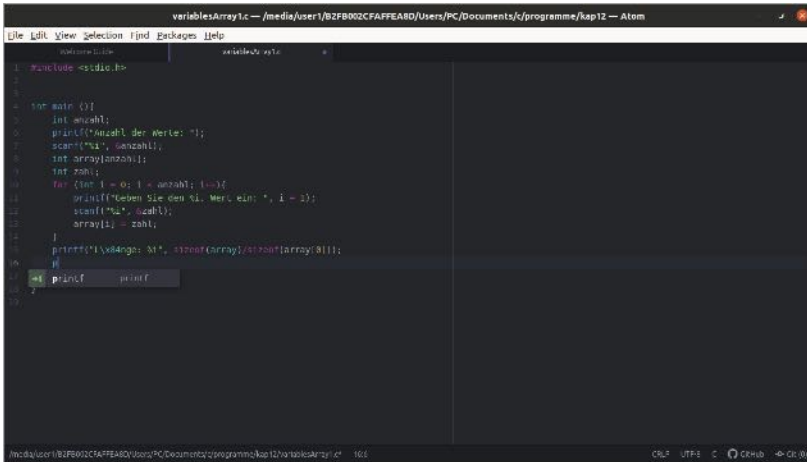


Abb. 6.22 Ein Programm in Atom mit Vorschlag für die Autovervollständigung

Um die angegebenen Texteditoren zu installieren, können Sie wieder das Ubuntu Software Center verwenden. Sollten Sie Interesse daran haben, einen der genannten Editoren zu installieren, ist dies daher problemlos möglich. Alternativ dazu ist die Installation über das Terminal möglich. Dessen Verwendung wird jedoch erst in Kapitel 7 vorgestellt.

6.6 Multimedia-Anwendungen

Multimedia-Anwendungen gehören für viele Anwender zu den grundlegenden Funktionen eines Computers. Auf diese Weise ist es möglich, private Videos anzuschauen oder Live-Streams aus dem Internet zu nutzen. Hinzu kommen viele weitere Möglichkeiten, die eine abwechslungsreiche Unterhaltung oder eine effiziente Informationsbeschaffung ermöglichen.

Multimedia-Anwendungen wurden insbesondere in den Jahren nach der Jahrtausendwende sehr beliebt. Der Grund dafür war zum einen, dass zu dieser Zeit immer mehr günstige Geräte auf den Markt kamen, die die Möglichkeit boten, digitale Videos aufzunehmen. Zum anderen

wurden zu dieser Zeit die Möglichkeiten, die das Internet bot, deutlich umfangreicher. Die meisten Linux-Distributionen boten damals jedoch eine relativ schlechte Unterstützung für Multimedia-Inhalte.

Als Beispiel hierfür soll die Linux-Distribution Ubuntu dienen. Diese unterstützte zu dieser Zeit keine proprietären Codecs. Bei einem Codec handelt es sich um ein Programm, das notwendig ist, um ein bestimmtes Audio- oder Videoformat wiederzugeben. Die meisten dieser Codecs lassen sich zwar unentgeltlich nutzen. Dennoch handelt es sich dabei nicht um Open-Source-Software. Ubuntu lehnte die Einbindung proprietärer Software jedoch ab. Daher verzichtete diese Distribution auf viele beliebte Formate – wie etwa MP3 oder MP4. Daher war es ohne weitere Maßnahmen nicht möglich, die entsprechenden Dateien wiederzugeben. Zwar konnten die Anwender die erforderlichen Codecs separat installieren. Das war jedoch mit einem erheblichen Zusatzaufwand verbunden. Aus diesem Grund erhielt Linux den Ruf, für Multimedia-Anwendungen nicht allzu gut geeignet zu sein.

Eine der ersten Distributionen, die diese Vorgehensweise änderte, war Linux Mint. Einer der wesentlichen Gründe für die Abspaltung von Ubuntu war hierbei die Unterstützung proprietärer Video-Codecs. Das stellte für den durchschnittlichen Anwender eine erhebliche Verbesserung dar und führte dazu, dass Linux Mint innerhalb kurzer Zeit sogar beliebter als die Ursprungs-Version Ubuntu wurde.

Mittlerweile haben die meisten Linux-Distributionen die Unterstützung für proprietäre Codecs jedoch deutlich verbessert – so auch Ubuntu. Wenn Sie sich nochmals an den Installationsprozess zurückerinnern, wurden Sie hierbei gefragt, ob Sie Software von Drittanbietern installieren möchten – unter anderem für Grafik- und Medieninhalte. Dabei handelt es sich genau um die proprietären Codecs, die soeben beschrieben wurden. Wenn Sie bei der Installation die entsprechende Auswahl bestätigt haben, sollten bereits alle gängigen Video- und Audioformate verfügbar sein.

Das wollen wir nun ausprobieren. Unter Ubuntu ist das Programm GNOME Videos bereits vorinstalliert. Dieses ist auch unter der frühe-

ren Bezeichnung totem bekannt. Wie Sie in Kapitel 6.2 gesehen haben, kommt diese Bezeichnung auch noch für viele interne Abläufe zum Einsatz. Um das Programm aufzurufen, können Sie entweder die Icons im Start-Menü nach dem entsprechenden Eintrag durchsuchen oder Sie können „totem“ oder „Videos“ in die Suchleiste eintragen.

Im Fenster, das sich daraufhin öffnet, können Sie links oben auf das Pluszeichen klicken, um ein Video aufzurufen. Abbildung 6.23 zeigt, dass Sie dabei die Auswahl aus den Videos auf Ihrer Festplatte und Internet-Videos haben. Auf diese Weise können Sie vielfältige Multimedia-Inhalte wiedergeben.

6

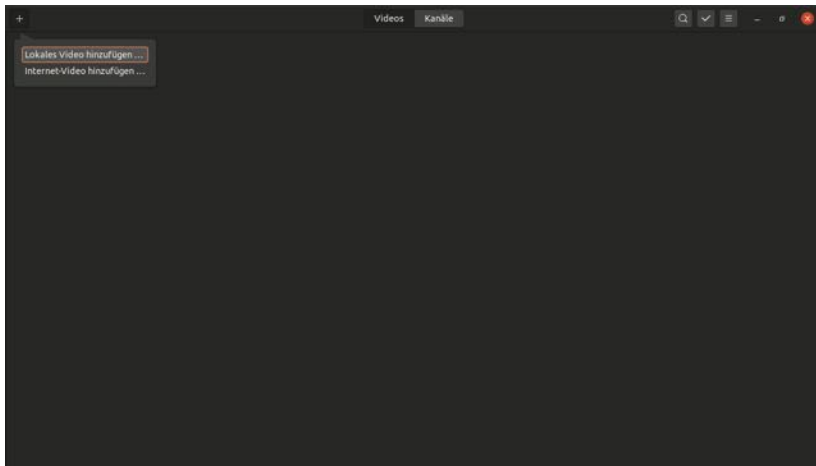


Abb. 6.23 Ein Video für die Wiedergabe auswählen

Neben dem vorinstallierten Multimedia-Player GNOME Videos bestehen noch zahlreiche weitere Möglichkeiten. Sehr beliebt ist beispielsweise der VLC Media Player. Dieser bietet eine Unterstützung für vielfältige Formate und zeichnet sich durch eine gute Performance und eine einfache Bedienung aus. Ebenfalls sehr beliebt sind der MPlayer und der SMPlayer. Eine weitere interessante Option stellt der MPV Player dar. All diese Programme müssen Sie jedoch separat installieren. Auch hierbei gilt wieder: Entweder sie warten ab, bis Sie im nächsten Kapitel

erfahren, wie Sie Programme über das Terminal installieren oder Sie verwenden das Ubuntu Software Center für diese Aufgabe.

6.7 Übungsaufgaben

In diesem Kapitel wurden nur die wichtigsten Programme vorgestellt, die Ubuntu bietet – beispielsweise für Büroarbeiten, Programmieraufgaben und für die Wiedergabe von Multimedia-Inhalten. Betrachten Sie das Start-Menü, um weitere Inhalte zu finden, die für Sie von Interesse sind. Dabei entdecken Sie noch viele weitere Möglichkeiten – von Programmen für die Bildbetrachtung über die Verwaltung von Scannern bis hin zur Verwaltung der Aktualisierungen. Darüber hinaus entdecken Sie einige Spiele. Gehen sie das Startmenü durch und halten Sie fest, welche Spiele standardmäßig in Ubuntu installiert sind.

Lösung:

Unter Ubuntu sind folgende Spiele vorinstalliert:

- ▶ AisleRiot Solitaire
- ▶ Mahjongg
- ▶ Minen
- ▶ Sudoku

Kapitel 7

Die Shell: Wichtiges Werkzeug für die Arbeit mit Linux

Im vorigen Kapitel haben wir uns mit den wichtigsten Programmen befasst, die unter Ubuntu bereits im System vorinstalliert sind. Zwar gab es auch hierbei einige Unterschiede zu den Programmen, die Sie aus Windows oder macOS kennen. Dennoch sind die grundsätzliche Funktionsweise und die Bedienung dabei recht ähnlich. In diesem Kapitel befassen wir uns mit der Shell. Dabei handelt es sich um ein Werkzeug, das bereits aus dem Betriebssystem UNIX bekannt ist. Auch alle unixoiden Betriebssysteme bieten eine entsprechende Shell an. Sie stellt eine der wesentlichen Gemeinsamkeiten zwischen UNIX und Linux dar. Zwar gibt es auch ähnliche Werkzeuge unter Windows und macOS. Allerdings ist deren Bedeutung für die Steuerung des Betriebssystems deutlich geringer. Auch der Funktionsumfang ist dabei nicht immer der gleiche. Unter Linux und anderen UNIX-artigen Betriebssystemen ist die Shell hingegen ausgesprochen wichtig. Wenn Sie alle Möglichkeiten, die das Betriebssystem Linux bietet, ausnutzen möchten, ist es daher sehr wichtig, sich in die Verwendung der *Shell* einzuarbeiten.

7.1 Was ist eine Shell?

Die Steuerung eines Computers hat sich seit der Entwicklung der ersten Rechner stark verändert. Ganz zu Beginn kamen vorwiegend Lochkarten zum Einlesen der Daten und Lampen oder Bänder für die Anzeige der Ergebnisse zum Einsatz. Als die ersten Geräte mit Bildschirm und Tastatur entwickelt wurden, kam dafür auch ein neues Eingabesystem zum Einsatz. Die Anwender konnten dabei bestimmte Befehle in Schriftform eintippen, die jeweils eine bestimmte Aktion nach sich zogen. Dabei handelt es sich um das Funktionsprinzip der Shell. Die Steuerung über eine intuitive grafische Benutzeroberfläche kam erst deutlich später hinzu.

Bevor wir uns den spezifischen Eigenschaften der Shell zuwenden, ist es wichtig, kurz auf die Namensgebung einzugehen. Hierfür gibt es zahlreiche Möglichkeiten. Eine alternative Bezeichnung für die Shell ist der Ausdruck Terminal. Obwohl die beiden Begriffe nicht vollkommen identisch sind, werden sie dennoch häufig als Synonyme verwendet. Unter Linux ist die Bezeichnung Terminal weit verbreitet. Die Shell ist beispielsweise über diesen Ausdruck erreichbar. Doch auch macOS verwendet diese Bezeichnung. Ein weiterer Begriff, der hierfür häufig zum Einsatz kommt, ist Kommandozeileninterpreter. Dieser Namen ist darauf zurückzuführen, dass die Shell eine einzelne Kommandozeile, die jedoch aus mehreren einzelnen Befehlen bestehen kann, einliest und anschließend interpretiert – oder mit anderen Worten, dass sie sie ausführt. Ein sehr ähnlicher Ausdruck ist dem Englischen entlehnt: Command Line Interface. Insbesondere die zugehörige Abkürzung CLI ist in vielen Werken zu diesem Thema zu finden. Hinzu kommen Bezeichnungen für bestimmte Arten von Shells – beispielsweise Bash oder cmd.exe. Mit diesem Thema befassen wir uns jedoch erst in Kapitel 7.2.

Nun wollen wir die Shell unter Linux kennenlernen. Zu diesem Zweck klicken wir das Startmenü an und suchen daraufhin nach dem Programm mit der Bezeichnung „Terminal“. Wenn wir dieses anklicken, erscheint das Fenster, das in Abbildung 7.1 zu sehen ist. Da wir das Terminal jedoch im Verlauf dieses Buchs noch häufiger verwenden werden, ist es empfehlenswert, sich direkt den Shortcut für diese Aufgabe zu merken: Mit Strg + Alt + T öffnen Sie das Terminal innerhalb von Sekundenbruchteilen.

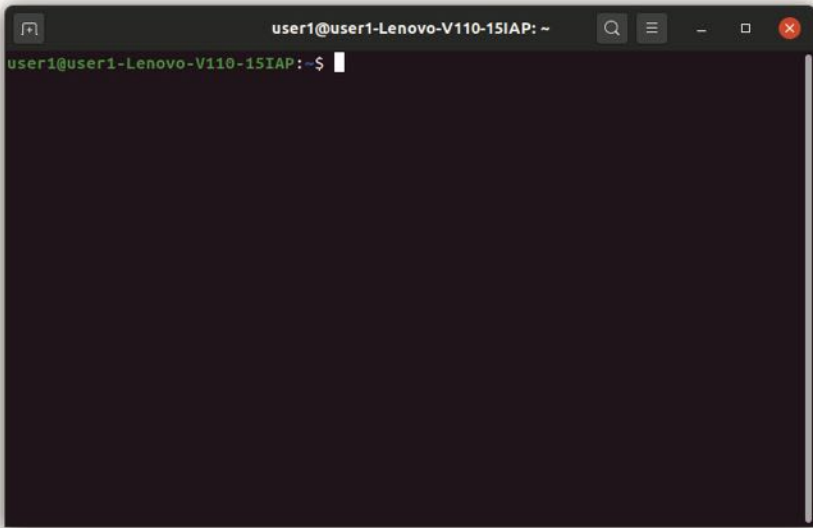
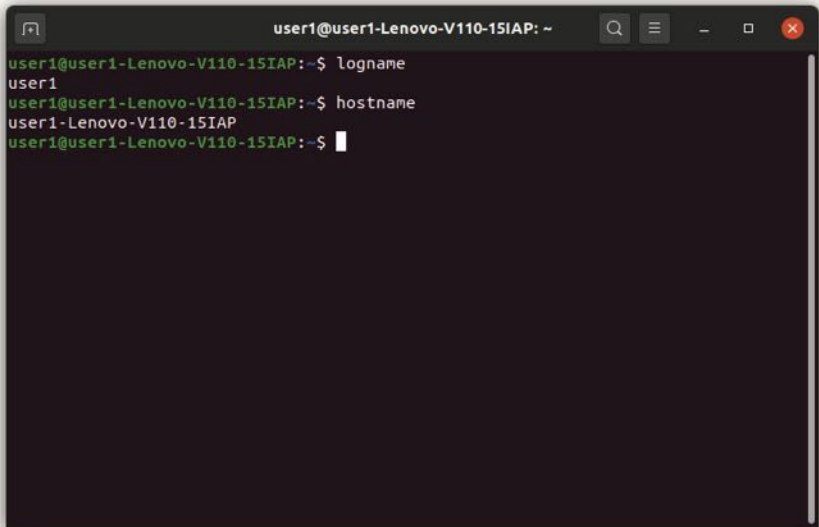


Abb. 7.1 Das Terminal mit der Standard-Shell in Ubuntu

Die Abbildung zeigt, dass die Shell hierbei zunächst anzeigt, welcher Nutzer im Moment registriert ist. Hierbei wird zunächst der Nutzername angezeigt, den wir bei der Installation des Betriebssystems ausgewählt haben – in diesem Beispiel `user1`. Danach folgt das `@`-Zeichen und anschließend der sogenannte Hostname. Hierbei handelt es sich um eine Bezeichnung, die der Computer automatisch erhalten hat. Diese setzt sich zum einen aus dem selbst gewählten Nutzernamen und zum anderen aus der Modellbezeichnung des Rechners zusammen. Danach folgen ein Doppelpunkt, eine Tilde und schließlich das Dollarzeichen. Diese Details sind für uns jedoch zunächst nicht von Bedeutung. Wichtiger ist es, dass es anschließend möglich ist, einen Befehl in die Shell einzugeben.

Das probieren wir nun einmal aus. Im weiteren Verlauf dieses Kapitels werden wir hierfür viele verschiedene Kommandos kennenlernen. Für unseren ersten Test verwenden wir zwei ganz einfache Befehle: `logname` und `hostname`. Der erste dient dazu, den Nutzernamen an-

zuzeigen. Mit dem zweiten Befehl geben wir den Hostnamen aus. Jeden dieser beiden Befehle schreiben wir in eine eigene Zeile. Wie bei Shell-Kommandos üblich, führen wir diese Befehle aus, indem wir die Eingabetaste betätigen. Auf diese Weise kommt es zu der Darstellung, wie sie in Abbildung 7.2 zu sehen ist.

A screenshot of a Linux terminal window. The title bar at the top reads 'user1@user1-Lenovo-V110-15IAP: ~'. The terminal has a dark background with green text. The prompt 'user1@user1-Lenovo-V110-15IAP:~\$' is followed by the command 'logname'. The output 'user1' is displayed on the next line. The prompt is followed by the command 'hostname'. The output 'user1-Lenovo-V110-15IAP' is displayed on the next line. The prompt is followed by a dollar sign '\$' and a cursor, indicating the terminal is ready for the next command.

```
user1@user1-Lenovo-V110-15IAP:~$ logname
user1
user1@user1-Lenovo-V110-15IAP:~$ hostname
user1-Lenovo-V110-15IAP
user1@user1-Lenovo-V110-15IAP:~$
```

Abb. 7.2 Die Ausgabe des Benutzernamens und des Hostnamens

Bei diesen beiden Befehlen handelt es sich nur um zwei kleine Beispiele für die Anwendung der Shell. In der Praxis kommen diese jedoch nur selten zum Einsatz, sodass es nicht unbedingt notwendig ist, sich diese zu merken. Sie sollen lediglich die grundsätzliche Verwendung der Shell vorstellen. Im weiteren Verlauf dieses Kapitels lernen wir dann noch zahlreiche weitere Befehle kennen, die sehr hilfreich für die Steuerung des Betriebssystems sind.

Die Shell hat in Linux eine sehr große Bedeutung. Das zeigt sich bereits an der Geschichte des Betriebssystems. Wie Sie in der Einleitung zu diesem Buch gelesen haben, entwickelte Torvalds ursprünglich eine Ter-

minal-Emulation – also ein Programm, das eine UNIX-Shell nachstellt. Aus dieser Shell entstand dann schließlich das gesamte Betriebssystem. Sie stellt daher die zentrale Steuerungseinheit dar. Viele Aufgaben lassen sich zwar mittlerweile auch unter Linux über eine grafische Benutzeroberfläche erledigen – wie etwa die Installation von Programmen oder die Verwaltung des Dateisystems. Doch lassen sich bis heute einige Details nur über die Shell festlegen.

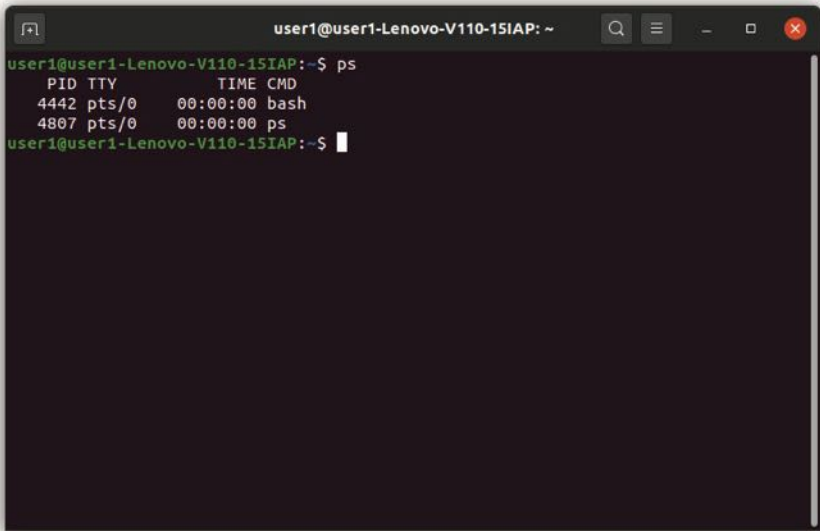
Als Beispiel für Anwendungen, die ausschließlich auf der Shell ablaufen, können wir die Anfertigung von Shellscrippts nennen. Hierbei handelt es sich um Computerprogramme, die sich direkt in der Shell ausführen lassen. In Kapitel 8 werden wir uns damit noch ausführlicher beschäftigen. Außerdem bietet die Shell uns die Möglichkeit, bestimmte Aufgaben zu automatisieren. Auf diese Weise ist es möglich, sie in regelmäßigen Zeitabständen auszuführen.

Darüber hinaus gibt es noch einen weiteren Grund, der für die Verwendung der Shell spricht: Sie gestaltet die Arbeit am Computer ausgesprochen effizient. Selbstverständlich ist für ihre Verwendung eine gewisse Einarbeitung notwendig. Wenn sie jedoch einmal die wesentlichen Befehle beherrschen und sich an die Arbeitsweise mit der Shell gewöhnt haben, können Sie die verschiedenen Funktionen des Computers damit ausgesprochen schnell abrufen. Das ist einer der wesentlichen Gründe dafür, dass die Shell bis heute die zentrale Steuerungseinheit des Betriebssystems Linux darstellt.

7.2 Verschiedene Shells in Linux

Um mit Linux zu arbeiten, ist es möglich, unterschiedliche Shells zu nutzen. Im letzten Abschnitt haben wir bereits das standardmäßig installierte Terminal ausprobiert. Um uns mit den verschiedenen Shells zu befassen, beginnen wir damit, hierbei den Typ herauszufinden. Zu diesem Zweck öffnen wir das Terminal und geben folgenden Befehl ein:

```
ps
```



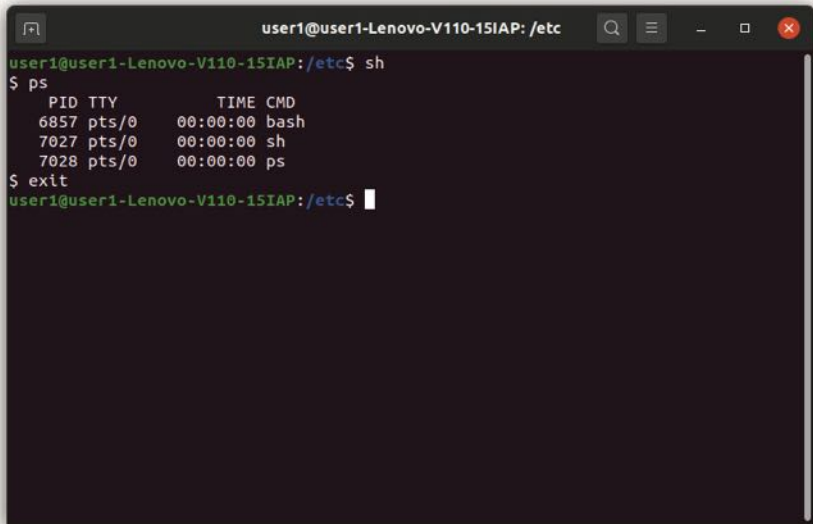
```
user1@user1-Lenovo-V110-15IAP: ~  
user1@user1-Lenovo-V110-15IAP:~$ ps  
  PID TTY          TIME CMD  
 4442 pts/0    00:00:00 bash  
 4867 pts/0    00:00:00 ps  
user1@user1-Lenovo-V110-15IAP:~$
```

Abb. 7.3 Die Anzeige der Shell-Version

Das Kommando `ps` dient dazu, die aktiven Prozesse auf dem Computer anzuzeigen. Wenn wir es ohne weiteren Zusatz verwenden, zeigt es jedoch lediglich die Prozesse an, die gerade im Rahmen der Shell ausgeführt werden. Abbildung 7.3 zeigt, dass hierbei zwei Prozesse aktiv sind. In der Spalte mit der Bezeichnung PID wird zunächst die Identifikationsnummer des Prozesses angezeigt. Die Spalte TTY gibt an, über welches Terminal der Prozess abläuft. Schließlich wird noch der Zeitpunkt angegeben. All diese Werte sind für uns jedoch nicht von großer Bedeutung. Um herauszufinden, welche Art von Shell wir benutzen, ist die letzte Spalte interessant. Hier sind zwei Werte aufgeführt. Zum einen `ps`. Diese Angabe bezieht sich auf den Befehl, den wir gerade eingegeben haben. Zum anderen steht hier der Ausdruck `bash`. Dieser bezieht sich auf den Prozess der Shell. Daran sehen wir, dass es sich hierbei um eine Shell des Typs Bash handelt.

Der Ausdruck Bash steht für Bourne-Again-Shell. Hierbei handelt es sich um einen zentralen Bestandteil von GNU. Auch die meisten Linux-Distributionen verwenden Bash. Sogar unter macOS war Bash zwischen 2003 und 2019 die voreingestellte Shell. Der Ausdruck leitet sich von der Bourne-Shell ab. Dabei handelt es sich um eine Shell, die bereits 1977 erschien und lange Zeit den Standard für UNIX-Betriebssysteme darstellte. Der Ausdruck Bourne-Again-Shell bezieht sich darauf, dass es sich hierbei um eine Neugestaltung der Bourne-Shell handelt. In der gesprochenen Sprache hört sich dieser Ausdruck jedoch fast gleich wie „born again“ an. Auch das war nach Angaben der Entwickler durchaus beabsichtigt, um zum Ausdruck zu bringen, dass es sich hierbei um eine Wiedergeburt der Shell-Funktionen handelt. Die Entwicklung von Bash begann 1988 im Rahmen des GNU-Projekts unter Leitung des Informatikers Brian Fox.

Die ursprüngliche Bourne-Shell war über den Befehl `sh` erreichbar. Das probieren wir nun aus, indem wir diesen Befehl ebenfalls eingeben. Dabei öffnet sich kein neues Fenster. Stattdessen wird die Bourne-Shell im bisherigen Terminal angezeigt. Allerdings hat sich die Darstellung dabei etwas geändert. Abbildung 7.4 zeigt, dass nun vor der Eingabe der Befehle nur noch das Dollarzeichen angezeigt wird – und nicht mehr der User- und der Hostname. Zusätzlich überprüfen wir wieder mit dem Befehl `ps`, welche Prozesse nun im Terminal ablaufen. Dabei wird deutlich, dass hier nun auch die Shell mit der Bezeichnung `sh` ausgeführt wird. Darüber hinaus zeigt dies, dass viele Befehle bei den verschiedenen Shells vollkommen identisch sind. Um wieder zur Bash zurückzukehren, verwenden wir den Befehl `exit`.

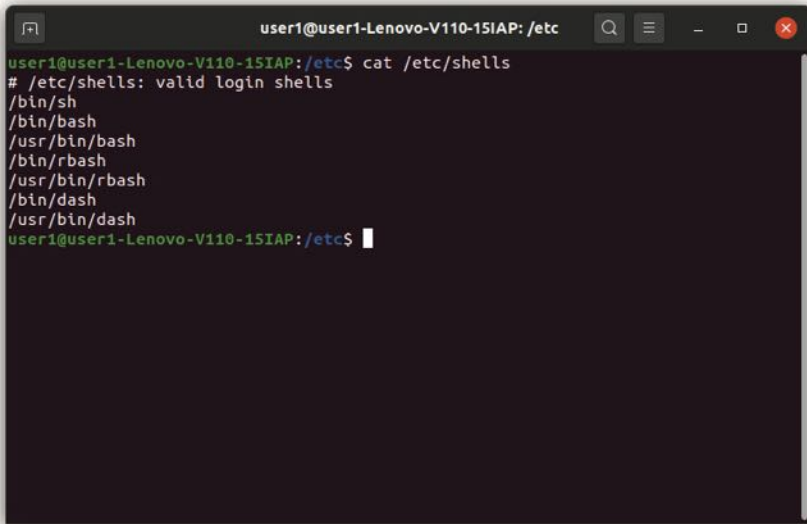


```
user1@user1-Lenovo-V110-15IAP: /etc
user1@user1-Lenovo-V110-15IAP:/etc$ sh
$ ps
  PID TTY          TIME CMD
  6857 pts/0        00:00:00 bash
  7027 pts/0        00:00:00 sh
  7028 pts/0        00:00:00 ps
$ exit
user1@user1-Lenovo-V110-15IAP:/etc$
```

Abb. 7.4 Der Aufruf der Bourne-Shell

Hierbei handelt es sich jedoch nicht um die einzige Alternative zu Bash. Es gibt noch viele weitere Möglichkeiten – von denen einige bereits vorinstalliert sind. Um die Angebote kennenzulernen, verwenden wir zunächst einmal den Befehl `cat`. Dieser steht für “concatenation” und erledigt vielfältige Aufgaben. Unter anderem lässt sich damit der Inhalt einer Datei ausgeben. Auf diese Weise geben wir nun den Inhalt der shells-Datei aus, die sich im Ordner `etc` innerhalb des `root`-Verzeichnisses befindet. Dafür ist der folgende Befehl notwendig:

```
1 cat /etc/shells
```

```
user1@user1-Lenovo-V110-15IAP: /etc$ cat /etc/shells
# /etc/shells: valid login shells
/bin/sh
/bin/bash
/usr/bin/bash
/bin/rbash
/usr/bin/rbash
/bin/dash
/usr/bin/dash
user1@user1-Lenovo-V110-15IAP: /etc$
```

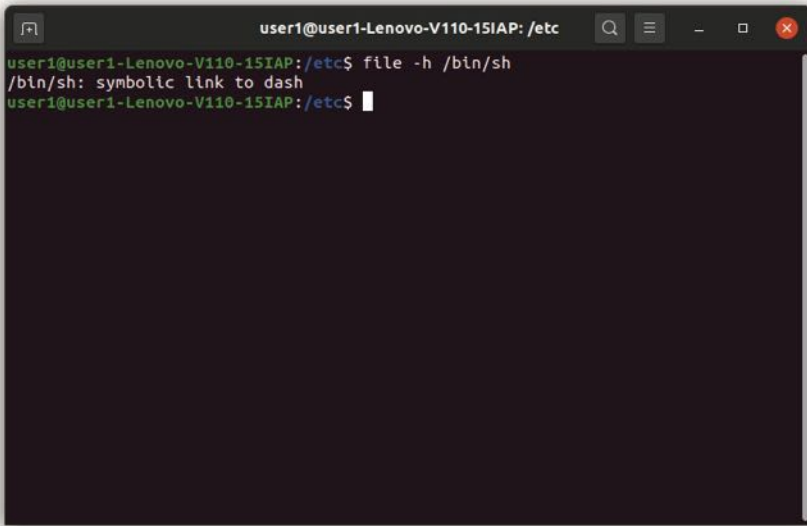
Abb. 7.5 Die Ausgabe der verfügbaren Shells

Hierbei erkennen wir, dass neben den bereits vorgestellten Möglichkeiten `sh` und `bash` noch weitere Alternativen zur Verfügung stehen: `rbash` und `dash`. Diese lassen sich ebenfalls über den gleichnamigen Befehl aufrufen. Mit Ausnahme des Begriffs `sh` sind alle Alternativen doppelt aufgeführt – sowohl im Verzeichnis `bin` als auch im Verzeichnis `usr/bin`. Das liegt jedoch an der internen Struktur des Dateisystems und muss uns an dieser Stelle nicht weiter beschäftigen. Bei beiden Alternativen handelt es sich jeweils um die gleichen Shells.

Neben den vorinstallierten Shells bestehen noch einige weitere Möglichkeiten. Um diese zu nutzen, wäre es jedoch notwendig, sie separat zu installieren. Da dies für die Aufgaben in diesem Buch nicht erforderlich ist, verzichten wir jedoch darauf. Dennoch soll kurz darauf hingewiesen werden, dass es beispielsweise noch die C-Shell (`csh`) und die Z-Shell (`zsh`) gibt. Von einiger Bedeutung ist auch die Korn-Shell (`ksh`). Diese ist insbesondere erwähnenswert, da sie als Vorbild für den POSIX-Standard diente.

Aufgrund der vielfältigen Auswahlmöglichkeiten stellt sich nun die Frage, worin der Unterschied zwischen den verschiedenen Shells besteht und welche Vor- und Nachteile diese jeweils bieten. Dabei sei zunächst angemerkt, dass die grundlegenden Funktionen dabei jeweils genau die gleichen sind. Das zeigt sich beispielsweise beim gerade verwendeten Befehl `ps`. Dieser lässt sich in allen vier vorinstallierten Shells – genauso wie in bei den meisten weiteren Optionen – verwenden. Das Gleiche trifft auf viele weitere Shell-Kommandos zu. Bei einfachen Anwendungen ist daher keinerlei Unterschied zu erkennen. Dennoch sei angemerkt, dass sich Bash durch einen deutlich größeren Befehlsumfang auszeichnet als die ursprüngliche Bourne-Shell. Für die Anwendungen in diesem Buch spielt dies allerdings keine große Rolle, da wir auf diese spezialisierten Anwendungen nicht eingehen werden.

An dieser Stelle sei auch noch ein weiteres Detail zur Verwendung der Shells erwähnt. Es wurde gesagt, dass es mit dem Befehl `sh` möglich ist, die ursprüngliche Bourne-Shell zu starten. Dieser kam über Jahrzehnte hinweg bei UNIX-Systemen zum Einsatz, um die Bourne-Shell zu starten. Als Linux entstand, konnte das Betriebssystem diese Shell aus lizenzrechtlichen Gründen jedoch nicht verwenden. Daher fügten die Entwickler für den Befehl `sh` einen symbolischen Link zu einer anderen Shell ein. Über viele Jahre hinweg war dies Bash. Mittlerweile verlinken die meisten modernen Linux-Systeme den Befehl `sh` jedoch mit Dash (Debian Alquist Shell). Hierbei handelt es sich um eine moderne Shell, die jedoch in vielen Bereichen mit der originalen Bourne-Shell übereinstimmt. Im Vergleich zu Bash zeichnet sie sich insbesondere durch einen deutlich geringeren Umfang aus. Das führt zu einer besseren Performance. Daher ist die Verwendung von Dash stets dann zu empfehlen, wenn es auf einen schnellen Ablauf der Programme ankommt. Um zu überprüfen, auf welche Shell der Ausdruck `sh` in Ihrem System verlinkt, können Sie das Kommando `file -h /bin/sh` verwenden. Abbildung 7.6 zeigt, dass unter Ubuntu 20.04 der Link zu Dash führt.

A terminal window with a dark background. The title bar shows 'user1@user1-Lenovo-V110-15IAP: /etc'. The prompt is 'user1@user1-Lenovo-V110-15IAP:/etc\$'. The command 'file -h /bin/sh' has been entered. The output is '/bin/sh: symbolic link to dash'. The prompt is now 'user1@user1-Lenovo-V110-15IAP:/etc\$' with a cursor.

```
user1@user1-Lenovo-V110-15IAP: /etc
user1@user1-Lenovo-V110-15IAP:/etc$ file -h /bin/sh
/bin/sh: symbolic link to dash
user1@user1-Lenovo-V110-15IAP:/etc$
```

Abb. 7.6 Die Überprüfung des Links beim Kommando `sh`

Abschließend sei noch kurz die Shell `rbash` erwähnt. Diese ist sehr ähnlich wie `Bash` aufgebaut. Der Buchstabe `R` steht hierbei für `restricted` und sagt aus, dass dabei einige Funktionen nicht ausführbar sind. Das ist insbesondere für sicherheitskritische Anwendungen sinnvoll.

Für die Anwendungen in diesem Buch ist es jedoch nicht notwendig, andere Shells zu verwenden. Sie können hierfür problemlos die Standard-Shell `Bash` nutzen. Allerdings ist es auch möglich, für die eine oder andere Aufgabe auch einmal eine der anderen Möglichkeiten auszuprobieren.

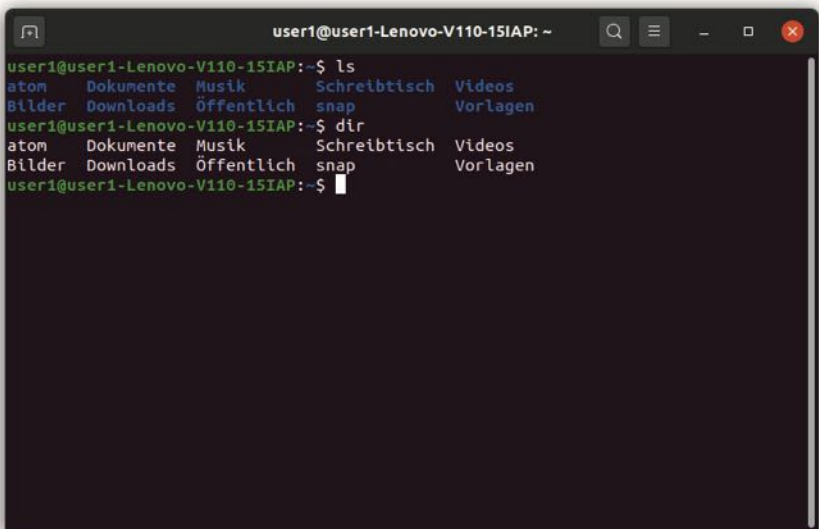
7.3 Dateien und Ordner verwalten

Eine der wichtigsten Aufgaben der Shell besteht darin, die Dateien und Ordner des Rechners zu verwalten. Wie wir bereits gesehen haben, bietet Ubuntu hierfür zwar auch eine grafische Oberfläche an. Die Verwaltung der Ordner und Dateien über die Shell ist jedoch sehr effizient und zählt außerdem zu den typischen Eigenschaften eines Linux-Betriebs-

systems. Wie Sie dabei vorgehen können, erfahren Sie in den folgenden Abschnitten.

Die Inhalte eines Ordners anzeigen

Um mit Ordnern und Dateien arbeiten zu können, ist es zunächst wichtig, sich die Inhalte anzeigen zu lassen. Hierfür können wir zwei verschiedene Befehle verwenden: `ls` und `dir`. Abbildung 7.7 zeigt, dass deren Funktion beinahe identisch ist. Beide Kommandos listen den Inhalt des aktuellen Ordners auf. Unterschiede bestehen dabei lediglich in der Darstellung und in manchen Fällen in der Sortierung.

A screenshot of a terminal window titled 'user1@user1-Lenovo-V110-15IAP: ~'. The window shows the execution of two commands: 'ls' and 'dir'. Both commands output a list of files and directories in the current directory. The output is as follows:

```
user1@user1-Lenovo-V110-15IAP:~$ ls
atom  Dokumente  Musik      Schreibtisch  Videos
Bilder Downloads  Öffentlich snap         Vorlagen
user1@user1-Lenovo-V110-15IAP:~$ dir
atom  Dokumente  Musik      Schreibtisch  Videos
Bilder Downloads  Öffentlich snap         Vorlagen
user1@user1-Lenovo-V110-15IAP:~$
```

Abb. 7.7 Die Auflistung der Verzeichnisse im aktuellen Ordner

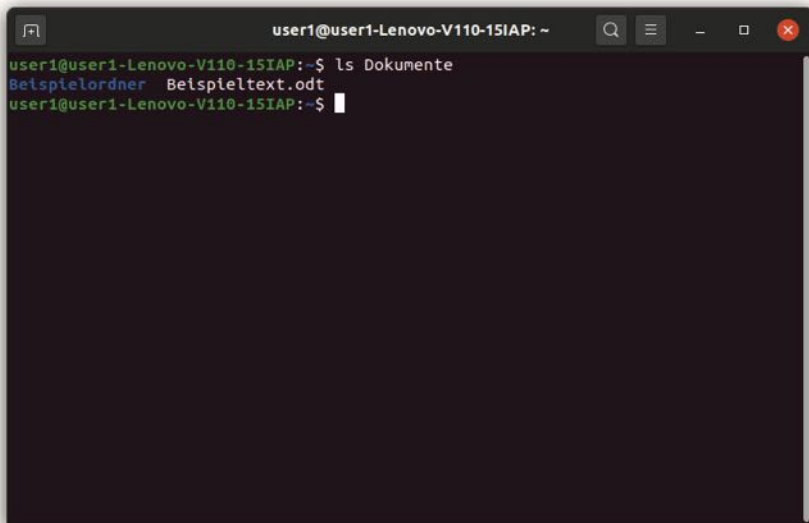
Für unsere weiteren Beispiele verwenden wir stets das Kommando `ls`. Hierbei handelt es sich um den standardkonformen Befehl, sodass dieser vorzuziehen ist. Die obige Abbildung zeigt, dass hier genau die gleichen Ordner aufgelistet werden wie beim Aufruf der grafischen Benutzeroberfläche. Hierbei handelt es sich um den sogenannten persön-

lichen Ordner. Dieser trägt Ihren Benutzernamen und dient dazu, alle Ihre persönlichen Dokumente aufzunehmen – Schriftstücke, Fotos, Videos und viele weitere. Wenn Sie die Shell öffnen, befinden Sie sich zu Beginn stets in diesem Ordner.

Nun ist es aber selbstverständlich auch häufig notwendig, den Inhalt weiterer Ordner anzuzeigen. Das ist möglich, indem Sie einfach den Ordernamen hinter den `ls`-Befehl schreiben. Beispielsweise können Sie auf diese Weise einmal den Inhalt des Ordners Dokumente anzeigen lassen: `ls Dokumente`. Abbildung 7.8 zeigt das.

7

Hinweis: Achten Sie darauf, dass Linux bei Dateien und Ordnern zwischen Groß- und Kleinschreibung unterscheidet. Das bedeutet, dass Sie mit dem Befehl `ls dokumente` nicht ans Ziel kommen. Das stellt einen wichtigen Unterschied zum Betriebssystem Windows dar, das nicht zwischen Groß- und Kleinschreibung unterscheidet.



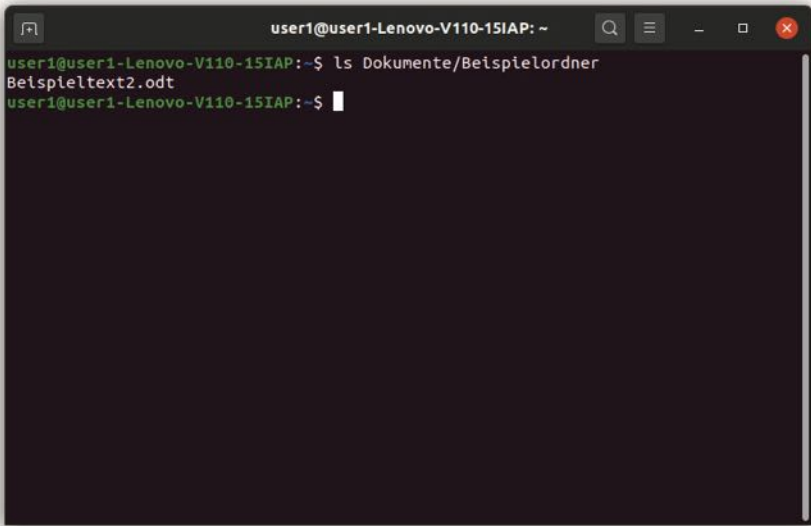
```
user1@user1-Lenovo-V110-15IAP: ~  
user1@user1-Lenovo-V110-15IAP:~$ ls Dokumente  
Beispielordner  Beispieltext.odt  
user1@user1-Lenovo-V110-15IAP:~$
```

Abb. 7.8 Der Inhalt des Ordners Dokumente

Sie sehen, dass hier nun zwei Einträge aufgelistet sind: ein Ordner und eine Datei. Um den Unterschied deutlich zu machen, sind diese in verschiedenen Farben dargestellt. In diesem Beispiel werden die Inhalte aufgezeigt, die wir selbst auf dem Rechner, mit dem dieser Versuch durchgeführt wurde, gespeichert haben. Wenn Sie diesen Befehl auf Ihrem eigenen Computer ausprobieren, kommt es hierbei sicherlich zu einer anderen Anzeige. Mithilfe der grafischen Benutzeroberfläche, die wir bereits kennengelernt haben, können Sie nun jedoch auch eigene Ordner erstellen, um die entsprechenden Funktionen auszuprobieren. Mithilfe des LibreOffice Writers oder eines anderen Programms können Sie auch einige Beispieldateien erstellen, um die Inhalte der Ordner zu überprüfen.

Wenn Sie nun innerhalb des Ordners Dokumente einen weiteren Ordner erstellt haben, können Sie auch dessen Inhalt anzeigen lassen. Hierfür müssen Sie dessen Pfad angeben. Dabei verwenden wir zunächst den *relativen Pfad*. Das bedeutet, dass wir den Ordner als Ausgangspunkt nehmen, in dem sich die Shell gerade befindet. Danach geben wir zunächst den Ordner Dokumente an. Anschließend geben wir den Namen des untergeordneten Verzeichnisses an, das sich darin befindet. Um diese voneinander zu trennen, verwenden wir den Schrägstrich. Das führt dann zu folgendem Befehl: `ls Dokumente/Beispielordner`.

Hinweis: Das Betriebssystem Windows verwendet zur Trennung der Verzeichnisse normalerweise den Rückwärts-Schrägstrich beziehungsweise Backslash (\). Dabei ist jedoch auch der gewöhnliche Schrägstrich zulässig. Unter Linux müssen Sie darauf achten, dass hierbei ausschließlich die Verwendung des normalen Schrägstrichs zulässig ist.

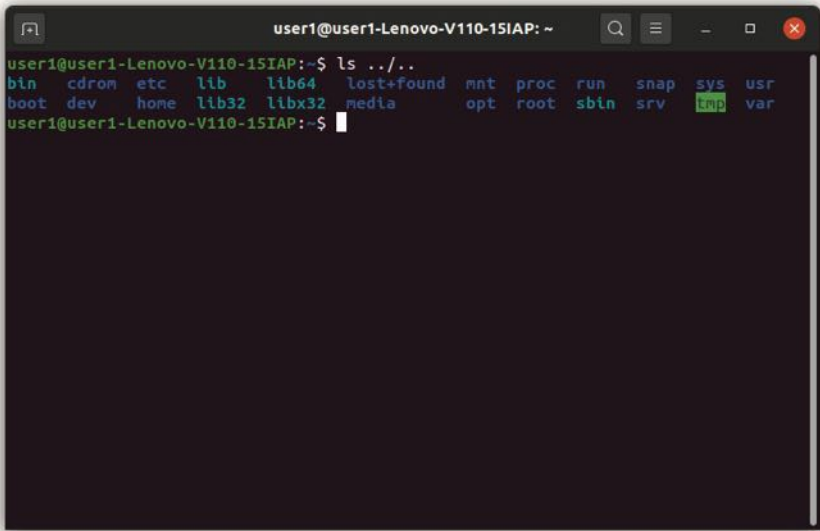


```
user1@user1-Lenovo-V110-15IAP: ~  
user1@user1-Lenovo-V110-15IAP:~$ ls Dokumente/Beispielordner  
Beispieltext2.odt  
user1@user1-Lenovo-V110-15IAP:~$
```

Abb. 7.9 Die Inhalte des untergeordneten Verzeichnisses

In Abbildung 7.9 ist zu sehen, dass wir auf diese Weise den Inhalt einer untergeordneten Ebene der Verzeichnisstruktur erreichen können. Durch das Hinzufügen weiterer Ordner, jeweils durch einen Schrägstrich voneinander getrennt, ist es möglich, beliebig viele weitere Ebenen zu erreichen. Doch auch die umgekehrte Richtung ist hierbei häufig notwendig. Dazu müssen wir zwei aufeinanderfolgende Punkte eingeben. Das führt dazu, dass wir eine Ebene höher in der Verzeichnisstruktur aufsteigen. Um mehrere Ebenen aufzusteigen, ist es auch möglich, mehrmals zwei Punkte anzugeben – auch hierbei wieder durch einen Schrägstrich voneinander getrennt. Das probieren wir nun aus, indem wir zwei Ebenen aufsteigen:

```
1 ls ../../..
```

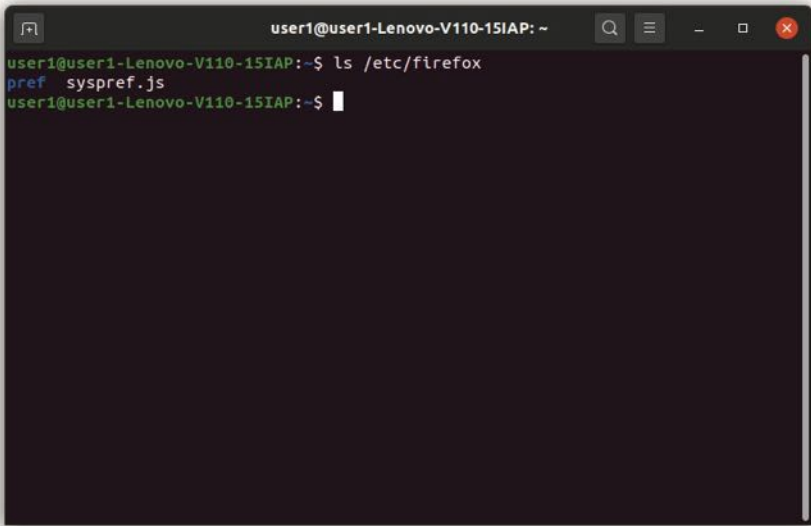


```
user1@user1-Lenovo-V110-15IAP: ~  
user1@user1-Lenovo-V110-15IAP:~$ ls ../../  
bin  cdrom  etc  lib  lib64  lost+found  mnt  proc  run  snap  sys  usr  
boot  dev  home  lib32  libx32  media  opt  root  sbin  srv  tmp  var  
user1@user1-Lenovo-V110-15IAP:~$
```

Abb. 7.10 Der Inhalt des root-Verzeichnisses

Abbildung 7.10 zeigt, dass wir mit diesem Befehl in das root-Verzeichnis vorgedrungen sind. Das macht deutlich, dass der persönliche Ordner des Anwenders in der zweiten Ebene der Verzeichnisstruktur liegt.

Neben dem relativen Pfad ist es auch möglich, den *absoluten Pfad* zu nutzen. Das bedeutet, dass wir hierbei stets im root-Verzeichnis beginnen. Dazu müssen wir den Pfadnamen mit einem Schrägstrich beginnen. Danach können wir - vom root-Verzeichnis ausgehend - durch die Ordnerstruktur navigieren, bis wir das gewünschte Ziel erreichen. Um dies zu zeigen, wählen wir im Stammverzeichnis den Ordner `etc` und daraufhin `firefox` aus. Auf diese Weise sehen wir, aus welchen Ordnern und Dateien der Webbrowser Firefox besteht. Das ist in Abbildung 7.11 zu erkennen. Der zugehörige Befehl sieht dann so aus: `ls /etc/firefox.`



```
user1@user1-Lenovo-V110-15IAP: ~  
user1@user1-Lenovo-V110-15IAP:~$ ls /etc/firefox  
pref  syspref.js  
user1@user1-Lenovo-V110-15IAP:~$
```

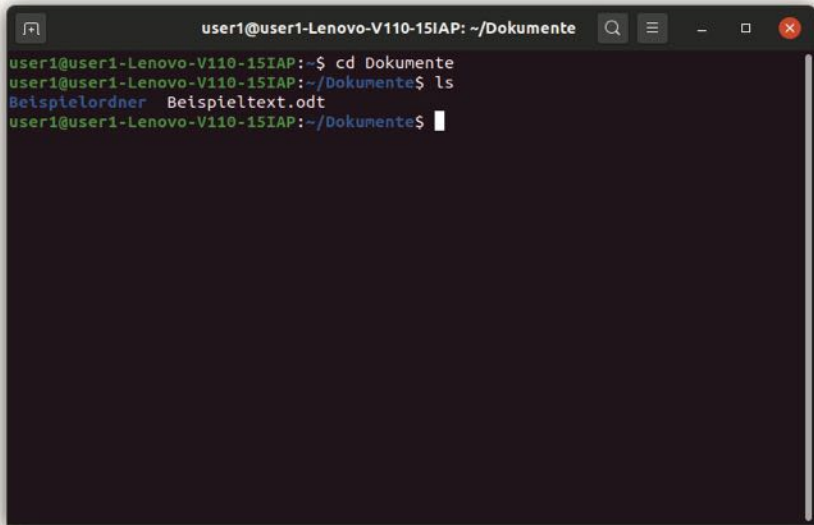
Abb. 7.11 Der Inhalt des Ordners firefox

Tipp: Oftmals ist es notwendig, in der Shell einen Befehl mehrere Male auszuführen. Dabei ist es jedoch nicht notwendig, diesen erneut einzutippen. Indem sie die Pfeiltaste nach oben betätigen, gelangen Sie direkt zum letzten Befehl, den Sie verwendet haben. Betätigen Sie diese Taste mehrere Male, können Sie sich auch zu den vorherigen Befehlen durcharbeiten. Das erleichtert die Arbeit deutlich.

Das Verzeichnis wechseln

Bislang haben wir lediglich die Inhalte verschiedener Verzeichnisse angezeigt. Dabei sind wir jedoch stets in unserem persönlichen Ordner geblieben. Es ist aber auch möglich, das Verzeichnis zu wechseln. Zu diesem Zweck kommt der Befehl `cd` zum Einsatz. Wenn wir beispielsweise in das Verzeichnis Dokumente wechseln möchten, erreichen wir das mit dem Befehl `cd Dokumente`. Dort können wir nun nochmals den Befehl `ls` eingeben. In Abbildung 7.12 ist zu sehen, dass nun der In-

halt des Ordners Dokumente angezeigt wird, ohne dass wir dafür beim `ls`-Befehl das entsprechende Verzeichnis angeben.

A terminal window titled 'user1@user1-Lenovo-V110-15IAP: ~/Dokumente'. The prompt is 'user1@user1-Lenovo-V110-15IAP:~\$'. The user enters 'cd Dokumente'. The prompt changes to 'user1@user1-Lenovo-V110-15IAP:~/Dokumente\$'. The user enters 'ls'. The output is 'Beispielordner Beispieltext.odt'. The prompt returns to 'user1@user1-Lenovo-V110-15IAP:~/Dokumente\$' with a cursor at the end.

```
user1@user1-Lenovo-V110-15IAP:~$ cd Dokumente
user1@user1-Lenovo-V110-15IAP:~/Dokumente$ ls
Beispielordner  Beispieltext.odt
user1@user1-Lenovo-V110-15IAP:~/Dokumente$
```

Abb. 7.12 Der Wechsel in den Ordner Dokumente und die Darstellung des Inhalts

Mit dem `cd`-Befehl können wir uns nun Ordner für Ordner durch die Verzeichnisstruktur arbeiten. Es ist aber auch möglich, die Ordner genau wie in den Beispielen zum `ls`-Befehl mit einem Schrägstrich miteinander zu verbinden. Auch hierbei können wir uns mit den zwei Punkten auf eine höhere Ebene begeben. Außerdem ist es möglich, absolute Pfade zu verwenden.

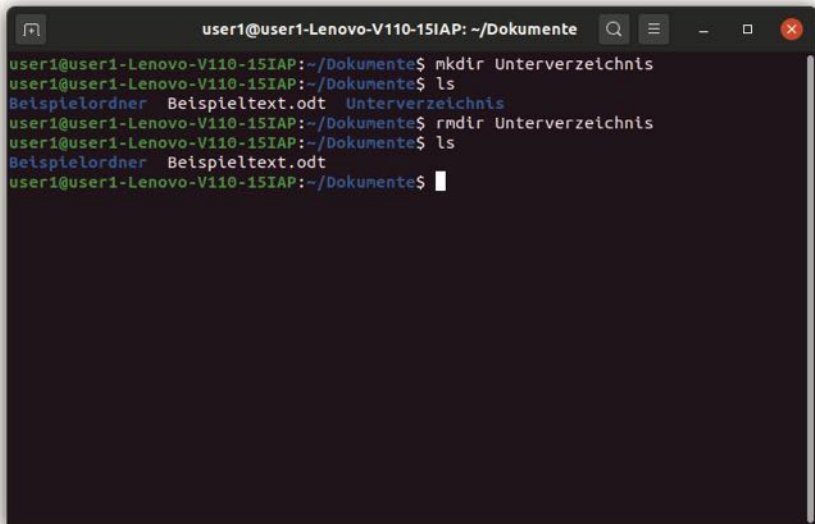
Ein neues Verzeichnis anlegen oder löschen

Häufig ist es auch notwendig, einen neuen Ordner anzulegen. Zu diesem Zweck verwenden wir den Befehl `mkdir` – gefolgt vom gewünschten Namen. Auch hierbei ist es möglich, einen Pfad aus mehreren Ordnern anzugeben. Dabei ist es jedoch wichtig, darauf zu achten, dass

hierbei nur der letzte Ordner neu angelegt wird. Das bedeutet, dass die übrigen Verzeichnisse des Pfads bereits bestehen müssen, um den entsprechenden Befehl auszuführen.

Darüber hinaus ist es gelegentlich notwendig, ein Verzeichnis zu löschen. Hierfür verwenden wir den Befehl `rmdir`. Auch hierbei müssen wir anschließend den Namen des Ordners angeben, den wir löschen möchten. Im folgenden Beispiel legen wir zunächst ein neues Verzeichnis an. Um die Auswirkungen zu kontrollieren, geben wir mit dem `ls`-Befehl den Inhalt des aktuellen Ordners aus. Daran erkennen wir, dass wir das neue Verzeichnis erstellt haben. Danach löschen wir es wieder und kontrollieren auch dies mit dem `ls`-Befehl. Abbildung 7.13 zeigt die Auswirkungen.

7



```
user1@user1-Lenovo-V110-15IAP: ~/Dokumente
user1@user1-Lenovo-V110-15IAP:~/Dokumente$ mkdir Unterverzeichnis
user1@user1-Lenovo-V110-15IAP:~/Dokumente$ ls
Beispielordner  Beispieltext.odt  Unterverzeichnis
user1@user1-Lenovo-V110-15IAP:~/Dokumente$ rmdir Unterverzeichnis
user1@user1-Lenovo-V110-15IAP:~/Dokumente$ ls
Beispielordner  Beispieltext.odt
user1@user1-Lenovo-V110-15IAP:~/Dokumente$
```

Abb. 7.13 Das Erstellen und das Löschen eines Ordners

Dieser Befehl gilt nur für Verzeichnisse. Um eine Datei zu löschen, kommt der Befehl `rm` zum Einsatz – gefolgt vom Dateinamen. Es ist auch möglich, diesen Befehl mit dem Zusatz `-r` zu versehen. Bei sol-

chen Ergänzungen spricht man von einer Flag. Flags erlauben es bei vielen Befehlen, die Funktionsweise zu beeinflussen. Unterschiedliche Buchstaben haben dabei verschiedene Auswirkungen. Beim Befehl `rm` bewirkt die Flag `-r`, dass sich damit ein Verzeichnis löschen lässt. Dieser Befehl ist sehr praktisch, da Sie damit auch einen vollen Ordner mit all seinen Inhalten entfernen können. Der Befehl `rmdir` lässt sich im Gegensatz dazu nur auf ein leeres Verzeichnis anwenden.

Dateien in einem Ordner aufrufen

Wenn wir mit Ordnern arbeiten, ist es häufig notwendig, eine der Dateien, die sich darin befinden, aufzurufen. Auch diese Aufgabe lässt sich über die Shell erledigen. Allerdings gibt es hierbei einen wichtigen Unterschied zur Verwendung der grafischen Benutzeroberfläche. Während diese automatisch erkennt, mit welchem Programm die entsprechende Datei gestartet werden muss, ist dies über die Shell nicht der Fall. Daher müssen wir hier immer das gewünschte Programm angeben. In unserem Beispiel öffnen wir eine odt-Datei für den LibreOffice Writer. In dem Ordner, in dem wir uns befinden, ist bereits eine passende Datei vorhanden: `Beispieltext.odt`. Diese können wir für unseren kleinen Test verwenden.

Um das gewünschte Programm vorzugeben, reicht es aus, den Begriff `libreoffice` zu verwenden. Dabei wird automatisch erkannt, dass es sich hierbei um eine odt-Datei handelt, die für den Writer vorgesehen ist. Anschließend müssen wir dann nur noch die Datei angeben:

```
1 libreoffice Beispieltext.odt
```

Das macht es erforderlich, für jede Datei, die wir aufrufen möchten, genau zu wissen, welche Anwendung wir dafür benötigen. Das erscheint auf den ersten Blick etwas schwierig. Meistens ist es aber zielführend, einfach den Namen des Programms einzugeben. Sollte dies einmal nicht den gewünschten Erfolg mit sich bringen, sollte es aber auch kein Problem darstellen, über eine kurze Internetrecherche herauszufinden, welcher Befehl für die entsprechende Anwendung notwendig ist. Nach

kurzer Zeit kennen Sie sicherlich die Befehle für alle Anwendungen, mit denen Sie regelmäßig arbeiten.

7.4 Anwender und Zugriffsrechte verwalten

Linux ist ein Mehrbenutzersystem. Das bedeutet, dass es vorgesehen ist, dass mehrere Anwender auf die gleiche Oberfläche zugreifen können. Im Gegensatz zu vielen anderen Betriebssystemen kann der Zugriff der verschiedenen Anwender dabei auch gleichzeitig stattfinden. Auf einem gewöhnlichen Gerät, das nur über eine Tastatur und über einen Bildschirm verfügt, lässt sich dies selbstverständlich nicht ganz so einfach umsetzen. Zahlreiche Server sind jedoch mit mehreren Ein- und Ausgabegeräten ausgestattet, sodass hierbei eine parallele Nutzung keine Seltenheit darstellt. Doch auch bei gewöhnlichen Computern kann ein paralleler Zugriff durch mehrere Nutzer stattfinden – beispielsweise über ein Netzwerk. Aus Sicherheitsgründen ist es hierfür jedoch wichtig, genau festzulegen, welcher Nutzer dabei Zugriff auf welche Dateien und Ordner erhält und welche Aktionen er damit durchführen kann.

Einen Anwender haben wir bereits bei der Installation des Betriebssystems festgelegt. Für dieses Buch haben wir ihn einfach `user1` genannt. Wenn Sie an dieser Stelle Ihren eigenen Namen eingetragen haben, dann hat er jedoch eine andere Bezeichnung erhalten. Dieser bei der Installation angelegte Nutzer-Account hat Zugriff auf die persönlichen Daten im `home`-Verzeichnis. Darüber hinaus kann er als sogenannter `root`-User tätig werden. Auf diese Weise hat er auch vollen Zugriff auf die System-Dateien im `root`-Verzeichnis und auf alle Funktionen, die der Systemsteuerung dienen. Um diese Rolle zu übernehmen, muss sich der entsprechende Anwender lediglich erneut mit seinem Passwort authentifizieren.

Wenn nun ein fremder Benutzer mit allen beschriebenen Rechten auf das System zugreift, könnte er umfangreiche Veränderungen daran vornehmen. Das könnte die Sicherheit gefährden. Aus diesem Grund haben neu hinzugefügte Anwender keine solch umfassenden Rechte.

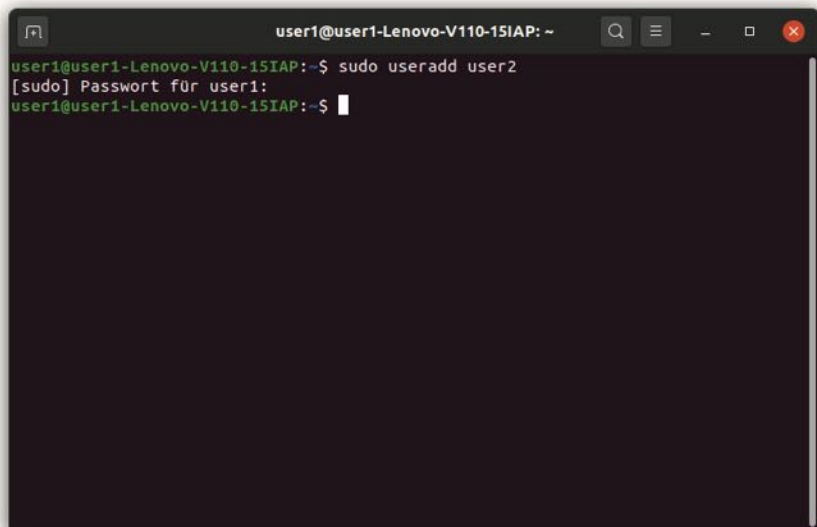
Für jeden neuen Nutzer wird ein persönliches home-Verzeichnis angelegt. Standardmäßig hat er nur Zugriff auf die Dateien, die darin enthalten sind. Über die Verwaltung der Zugriffsrechte lässt sich dies jedoch ändern. Auf diese Weise ist es beispielsweise möglich, Dateien für eine gemeinsame Bearbeitung freizugeben.

Zunächst sei erwähnt, dass die Verwaltung der Benutzerrechte auch über den Dateimanager möglich ist. Wenn Sie hierfür jedoch die Shell verwenden, können Sie nach einer gewissen Einarbeitung die Rechte schneller und flexibler verwalten. Darüber hinaus ist die Verwaltung der Rechte insbesondere auf einem Server von großer Bedeutung. Manche Server verfügen jedoch über keine grafische Oberfläche. In diesem Fall ist es sehr wichtig, dass Sie die entsprechenden Aufgaben über das Terminal erledigen können.

Nutzer anlegen, löschen und mit einem Passwort ausstatten

Der erste Schritt für die Verwaltung der Rechte besteht darin, einen neuen Benutzer hinzuzufügen. Zu diesem Zweck kommt der Befehl `useradd` zum Einsatz. Danach müssen Sie den Namen des neuen Anwenders angeben – in unserem Beispiel `user2`. Wenn Sie den Befehl nun ohne weitere Zusätze ausführen wollen, kommt es jedoch zu einem Problem. Das liegt daran, dass es sich hierbei um eine sicherheitsrelevante Tätigkeit handelt. Deshalb müssen wir den Ausdruck `sudo` voranstellen. Dieser Begriff steht für `superuser do`. Dieser Zusatz bewirkt, dass Sie als root-User tätig werden. Das macht es möglich, den entsprechenden Befehl mit allen Privilegien auszuführen. Dafür ist jedoch eine Authentifizierung notwendig. Daher fragt die Shell im nächsten Schritt nach dem Passwort. Hier müssen Sie die Zeichenfolge eingeben, die Sie bei der Installation des Systems vorgegeben haben. Das ist in Abbildung 7.14 zu sehen. Der komplette Befehl zum Anlegen eines neuen Anwenders sieht dann so aus:

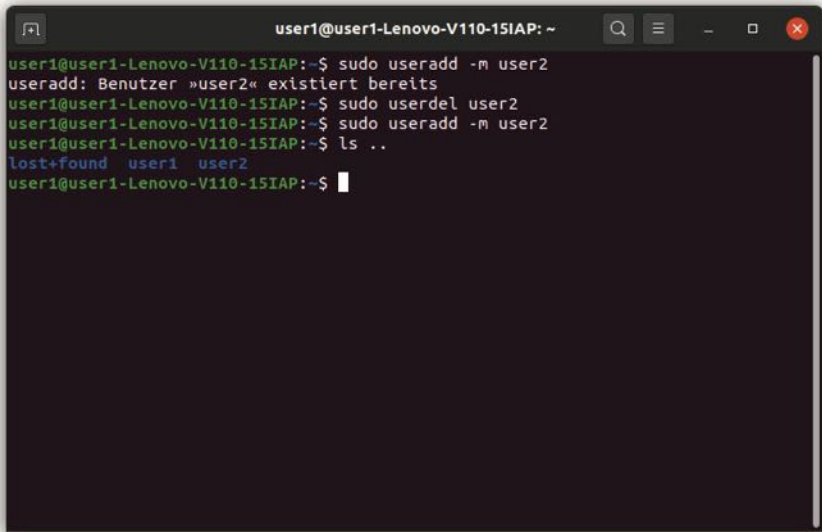
```
1 sudo useradd user2
```

A terminal window titled 'user1@user1-Lenovo-V110-15IAP: ~' with standard window controls. The terminal shows the command 'sudo useradd user2' being executed. The output is '[sudo] Passwort für user1:' followed by a prompt for the password. The terminal then shows the prompt 'user1@user1-Lenovo-V110-15IAP: ~\$' with a cursor, indicating the command has completed.

```
user1@user1-Lenovo-V110-15IAP: ~  
user1@user1-Lenovo-V110-15IAP: ~$ sudo useradd user2  
[sudo] Passwort für user1:  
user1@user1-Lenovo-V110-15IAP: ~$
```

Abb. 7.14 Einen neuen Anwender hinzufügen

Wenn Sie einen neuen Anwender auf diese Weise hinzufügen, erhält dieser jedoch kein eigenes home-Verzeichnis. Das bedeutet, dass er nur auf Ordner und Dateien zugreifen darf, die die anderen Anwender explizit für ihn freigeben. Er erhält jedoch keinen eigenen Bereich. Das lässt sich ändern, indem Sie nach dem Befehl `useradd` die Flag `-m` hinzufügen. Diese bewirkt, dass ein eigenes home-Verzeichnis hinzugefügt wird. Daher führen wir nun den folgenden Befehl aus: `sudo useradd -m user2`.



```

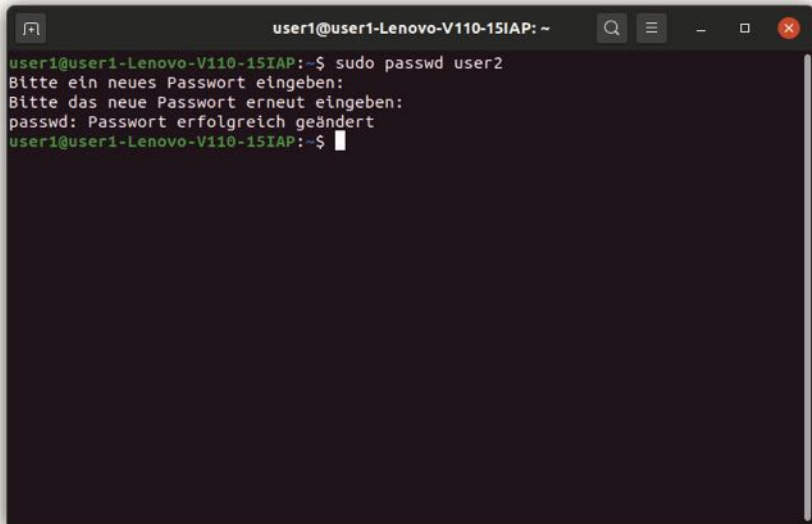
user1@user1-Lenovo-V110-15IAP: ~
user1@user1-Lenovo-V110-15IAP:~$ sudo useradd -m user2
useradd: Benutzer »user2« existiert bereits
user1@user1-Lenovo-V110-15IAP:~$ sudo userdel user2
user1@user1-Lenovo-V110-15IAP:~$ sudo useradd -m user2
user1@user1-Lenovo-V110-15IAP:~$ ls ..
lost+found user1 user2
user1@user1-Lenovo-V110-15IAP:~$

```

Abb. 7.15 Einen Anwender mit eigenem home-Verzeichnis hinzufügen

Hierbei erscheint jedoch zunächst eine Fehlermeldung. Das liegt daran, dass bereits ein Anwender mit dieser Bezeichnung existiert, da wir diesen ja gerade angelegt haben. Nun wäre es selbstverständlich möglich, einen beliebigen anderen Benutzernamen auszuwählen. Doch bietet es sich an, an dieser Stelle gleich darauf einzugehen, wie wir einen Anwender wieder löschen können. Hierzu kommt der Befehl `userdel` zum Einsatz – wieder mit dem Zusatz `sudo` und gefolgt vom Namen, der gelöscht werden soll. Abbildung 7.15 zeigt, dass es danach möglich ist, den Anwender erneut hinzuzufügen. Der Zusatz `-m` sorgt dafür, dass dabei ein neues Verzeichnis für dessen persönliche Dateien entsteht. Das überprüfen wir, indem wir den Inhalt des home-Verzeichnisses mit dem `ls`-Befehl ausgeben. Da sich dieses eine Ebene über unserem aktuellen Verzeichnis befindet, verwenden wir hierfür zwei Punkte als Pfadangabe. Hier erkennen wir nun den neuen Ordner mit der Bezeichnung `user2`. Hier kann der neue Anwender all seine persönlichen Dateien speichern und verwalten.

Beim Hinzufügen eines neuen Anwenders ist es sinnvoll, diesem auch ein Passwort zuzuweisen. Zu diesem Zweck verwenden wir den Befehl `passwd`. Auch hierbei ist der Zusatz `sudo` notwendig. Danach geben wir den Anwender an, für den wir das Passwort vorgeben möchten: `sudo passwd user2`.

A terminal window titled 'user1@user1-Lenovo-V110-15IAP: ~' with standard window controls. The terminal shows the command 'user1@user1-Lenovo-V110-15IAP:~\$ sudo passwd user2' and its output: 'Bitte ein neues Passwort eingeben:', 'Bitte das neue Passwort erneut eingeben:', 'passwd: Passwort erfolgreich geändert', and the prompt 'user1@user1-Lenovo-V110-15IAP:~\$' with a cursor.

```
user1@user1-Lenovo-V110-15IAP:~$ sudo passwd user2
Bitte ein neues Passwort eingeben:
Bitte das neue Passwort erneut eingeben:
passwd: Passwort erfolgreich geändert
user1@user1-Lenovo-V110-15IAP:~$
```

7

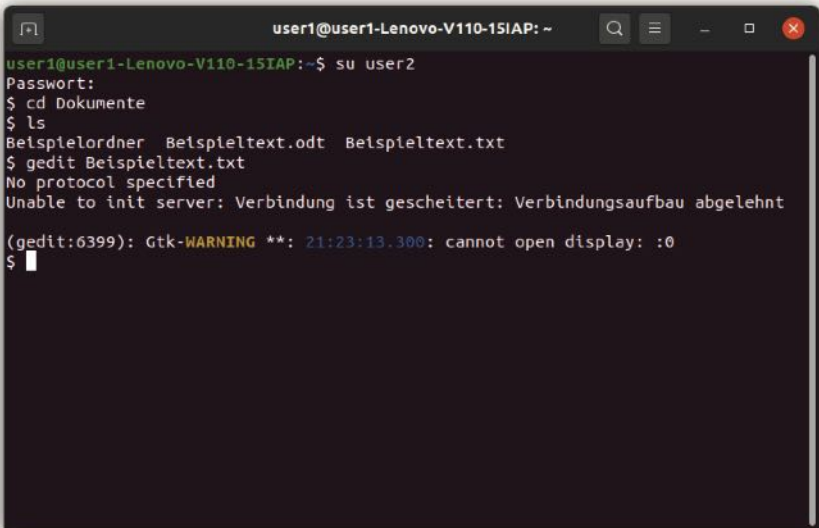
Abb. 7.16 Das Hinzufügen eines Passworts

Den Anwender wechseln

Nun ist es möglich, den Anwender zu wechseln. Zu diesem Zweck verwenden wir den Befehl `su` – gefolgt vom entsprechenden Nutzernamen. Danach müssen Sie das Passwort eingeben, das Sie zuvor vorgegeben haben.

Die weiteren Befehle in der Shell führen Sie dann im Namen des neuen Anwenders aus. Das bringt es mit sich, dass sich Ihre Rechte geändert haben. Das können Sie ausprobieren, indem Sie versuchen, auf eine der

Dateien im Ordner des anderen Anwenders zuzugreifen. Abbildung 7.17 zeigt, dass es in diesem Fall zu einem Fehler kommt.

A terminal window titled 'user1@user1-Lenovo-V110-15IAP: ~' with standard window controls. The terminal shows a sequence of commands: 'su user2' (password prompt), 'cd Dokumente', 'ls' (listing files), and 'gedit Beispieltext.txt'. The output shows the file list and then an error message: 'No protocol specified' and 'Unable to init server: Verbindung ist gescheitert: Verbindungsaufbau abgelehnt'. A GTK warning is also visible at the bottom.

```
user1@user1-Lenovo-V110-15IAP: ~  
user1@user1-Lenovo-V110-15IAP:~$ su user2  
Passwort:  
$ cd Dokumente  
$ ls  
Beispielordner  Beispieltext.odt  Beispieltext.txt  
$ gedit Beispieltext.txt  
No protocol specified  
Unable to init server: Verbindung ist gescheitert: Verbindungsaufbau abgelehnt  
(gedit:6399): Gtk-WARNING **: 21:23:13.300: cannot open display: :0  
$
```

Abb. 7.17 Der neue Anwender kann nicht auf die bisherigen Dateien zugreifen

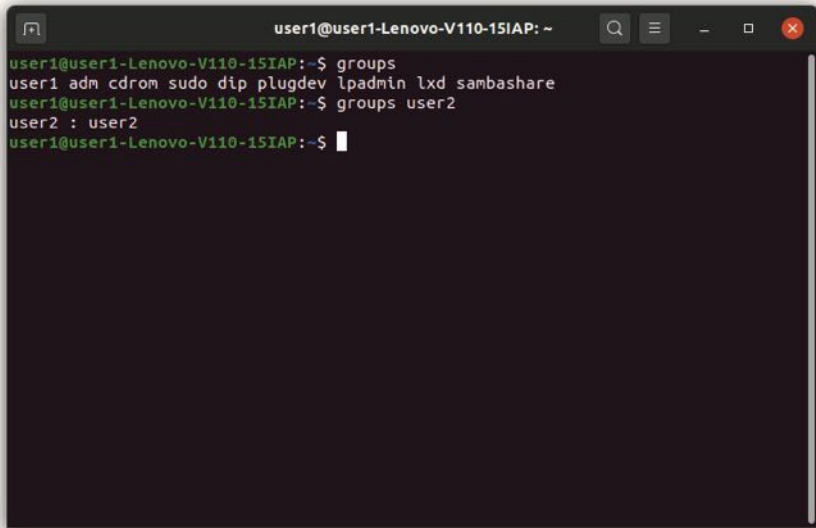
Der einfache Wechsel des Anwenders zeigt auch, dass hierbei die Nutzung der Shell Vorteile mit sich bringt. Über die grafische Benutzeroberfläche ist dieser Vorgang aufwendiger. Wenn sie beispielsweise über den Dateimanager versuchen, den neu angelegten Ordner für user2 zu öffnen, erhalten Sie keinen Zugriff darauf. Um ihn nutzen zu können, müssen Sie sich zunächst abmelden. Wenn Sie sich das nächste Mal neu anmelden, können Sie dann zwischen den beiden registrierten Nutzern auswählen. Nur wenn Sie sich hierbei für user2 entscheiden, können Sie auf den neu angelegten Ordner zugreifen.

Gruppen erstellen und verwalten

Linux erlaubt es nicht nur, einzelne Nutzer zu erstellen. Darüber hinaus ist es möglich, Gruppen zu gestalten. Wenn Sie einen neuen Anwender hinzufügen, wird automatisch eine neue Gruppe für diesen erstellt – vorausgesetzt, dass Sie das nicht durch das Hinzufügen der Flag `-N` verhindern. Der neue Anwender ist daraufhin das einzige Mitglied der neuen Gruppe. Das bedeutet, dass ohne weitere Maßnahmen die einzelnen Anwender und die Gruppen stets identisch sind. Allerdings bestehen auch Möglichkeiten, dies zu ändern.

7

Im ersten Schritt finden wir heraus, zu welchen Gruppen ein Anwender gehört. Dazu kommt der Befehl `groups` zum Einsatz. Ohne weiteren Zusatz gibt dieser die Gruppen aus, zu denen der aktuell angemeldete Anwender gehört. Abbildung 7.18 zeigt, dass der Nutzer, den Sie bei der Installation des Systems angegeben haben, in zahlreichen Gruppen aufgelistet ist. Das liegt daran, dass dieser alle Rechte innehat. Viele der hier genannten Gruppen dienen der Verwaltung des Systems und der Durchführung speziell geschützter Aktionen. Sie können nun jedoch auch nach dem Befehl `groups` den Namen eines weiteren Anwenders angeben – im nächsten Beispiel `user2`. Nun wird deutlich, dass dieser nur zu der Gruppe gehört, die beim Hinzufügen des Anwenders für ihn erstellt wurde. Er gehört zu keiner der übrigen Gruppen und hat daher wesentlich weniger Rechte.

A terminal window titled 'user1@user1-Lenovo-V110-15IAP: ~' with standard window controls. The terminal shows the following commands and output:

```
user1@user1-Lenovo-V110-15IAP:~$ groups
user1 adm cdrom sudo dip plugdev lpadmin lxd sambashare
user1@user1-Lenovo-V110-15IAP:~$ groups user2
user2 : user2
user1@user1-Lenovo-V110-15IAP:~$
```

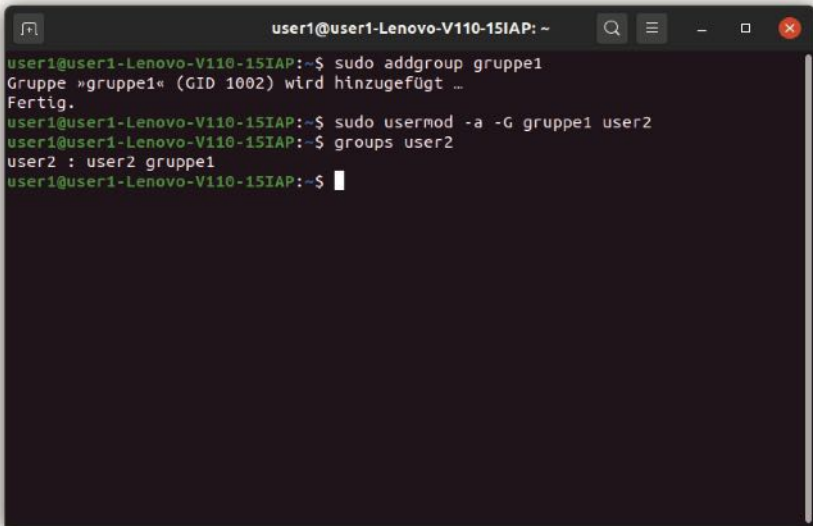
Abb. 7.18 Die Gruppen, zu denen unsere beiden Anwender gehören.

Nun fügen wir im ersten Schritt eine neue Gruppe hinzu. Dazu verwenden wir den Befehl `addgroup` – mit dem Zusatz `sudo`. Danach geben wir den Namen der neuen Gruppe vor:

```
1 sudo addgroup gruppel
```

Nun können wir einen Anwender zur neuen Gruppe hinzufügen. Zu diesem Zweck kommt das Kommando `usermod` zum Einsatz. Für diese Aufgabe müssen wir außerdem die beiden Flags `-a` und `-G` setzen. Danach folgen die Namen der Gruppe und des Anwenders, der zu ihr hinzugefügt werden soll:

```
1 sudo usermod -a -G gruppel user2
```

A terminal window titled 'user1@user1-Lenovo-V110-15IAP: ~' with standard window controls. It shows the execution of three commands: 'sudo addgroup gruppe1' which outputs 'Gruppe »gruppe1« (GID 1002) wird hinzugefügt ... Fertig.', 'sudo usermod -a -G gruppe1 user2', and 'groups user2' which outputs 'user2 : user2 gruppe1'.

```
user1@user1-Lenovo-V110-15IAP: ~  
user1@user1-Lenovo-V110-15IAP:~$ sudo addgroup gruppe1  
Gruppe »gruppe1« (GID 1002) wird hinzugefügt ...  
Fertig.  
user1@user1-Lenovo-V110-15IAP:~$ sudo usermod -a -G gruppe1 user2  
user1@user1-Lenovo-V110-15IAP:~$ groups user2  
user2 : user2 gruppe1  
user1@user1-Lenovo-V110-15IAP:~$
```

Abb. 7.19 Einen Anwender zur Gruppe hinzufügen

Abbildung 7.19 zeigt, dass wenn wir nun erneut die Gruppen des entsprechenden Anwenders auflisten, hier auch die neue Gruppe erscheint. Auf diese Weise ist es nicht nur möglich, Anwender zu neuen Gruppen hinzuzufügen. Es wäre auch möglich, bereits bestehende Gruppen zu erweitern – beispielsweise die persönliche Gruppe von user1 oder die Gruppen, die für die Verwaltung des Systems automatisch erzeugt wurden.

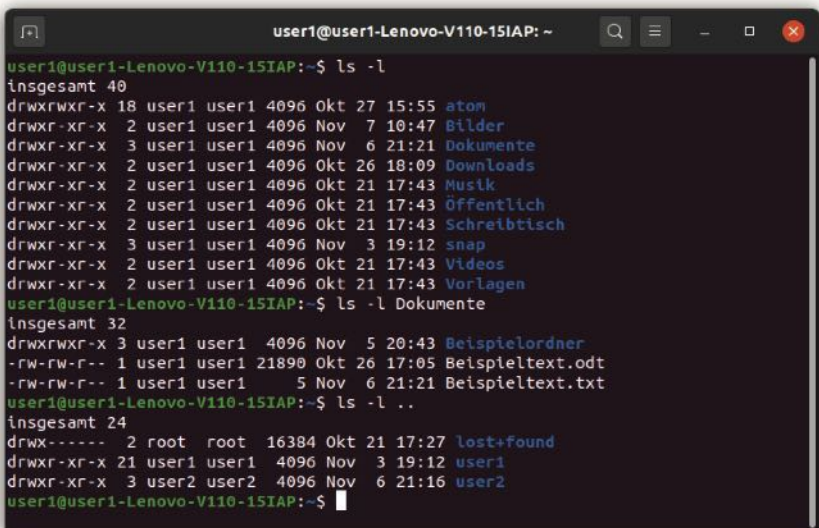
Auch wenn Sie einen Anwender wieder aus einer Gruppe entfernen möchten, kommt der Befehl `usermod` zum Einsatz. Dabei müssen Sie nun jedoch nicht die Gruppe angeben, aus der er gelöscht werden soll, sondern die Gruppen, zu denen er weiterhin gehören soll. Wenn Sie ihn beispielsweise wieder aus `gruppe1` entfernen möchten, erreichen Sie dies mit dem folgenden Befehl: `sudo usermod -G user2 user2`. Hierbei ist nur die eigene Gruppe dieses Anwenders – `user2` – angegeben. Aus allen übrigen Gruppen, zu denen er bislang gehörte, wird

er dadurch entfernt. Wenn mehrere Gruppen erhalten bleiben sollen, können Sie diese durch ein Komma voneinander trennen.

Sie können auch eine Gruppe komplett löschen. Hierfür kommt der Befehl `groupdel` zum Einsatz – mit vorangestelltem `sudo`. Danach müssen Sie lediglich den Namen der Gruppe angeben: `sudo groupdel gruppe1`.

Die Nutzungsrechte festlegen und verwalten

Nachdem wir nun Anwender und Gruppen gestaltet haben, müssen wir noch die Nutzungsrechte festlegen. Der erste Schritt besteht dabei darin, dass wir uns diese anzeigen lassen, um sie zu überprüfen. Dazu verwenden wir wieder den Befehl `ls` – allerdings nun mit der Flag `-l`. Abbildung 7.20 zeigt die Benutzerrechte für den aktuellen Ordner (das persönliche Verzeichnis von `user1`), für das darin befindliche Verzeichnis `Dokumente` und für den `home`-Ordner.



```
user1@user1-Lenovo-V110-15IAP: ~  
user1@user1-Lenovo-V110-15IAP:~$ ls -l  
insgesamt 40  
drwxrwxr-x 18 user1 user1 4096 Okt 27 15:55 atom  
drwxr-xr-x 2 user1 user1 4096 Nov 7 10:47 Bilder  
drwxr-xr-x 3 user1 user1 4096 Nov 6 21:21 Dokumente  
drwxr-xr-x 2 user1 user1 4096 Okt 26 18:09 Downloads  
drwxr-xr-x 2 user1 user1 4096 Okt 21 17:43 Musik  
drwxr-xr-x 2 user1 user1 4096 Okt 21 17:43 Öffentlich  
drwxr-xr-x 2 user1 user1 4096 Okt 21 17:43 Schreibtisch  
drwxr-xr-x 3 user1 user1 4096 Nov 3 19:12 snap  
drwxr-xr-x 2 user1 user1 4096 Okt 21 17:43 Videos  
drwxr-xr-x 2 user1 user1 4096 Okt 21 17:43 Vorlagen  
user1@user1-Lenovo-V110-15IAP:~$ ls -l Dokumente  
insgesamt 32  
drwxrwxr-x 3 user1 user1 4096 Nov 5 20:43 Beispielordner  
-rw-rw-r-- 1 user1 user1 21890 Okt 26 17:05 Beispieltext.odt  
-rw-rw-r-- 1 user1 user1 5 Nov 6 21:21 Beispieltxt.txt  
user1@user1-Lenovo-V110-15IAP:~$ ls -l ..  
insgesamt 24  
drwx----- 2 root root 16384 Okt 21 17:27 lost+found  
drwxr-xr-x 21 user1 user1 4096 Nov 3 19:12 user1  
drwxr-xr-x 3 user2 user2 4096 Nov 6 21:16 user2  
user1@user1-Lenovo-V110-15IAP:~$
```

Abb. 7.20 Die Rechte für einige Ordner und Dateien

Hier werden nun viele verschiedene Informationen zu den einzelnen Dateien aufgelistet. Dabei ist insbesondere von Bedeutung, wer der Besitzer einer Datei oder eines Ordners ist. Diese Angabe erhalten wir in der dritten und in der vierten Spalte. Jede Datei und jeder Ordner kann zwei Besitzer haben – einen einzelnen Anwender und eine Gruppe. Die dritte Spalte gibt den Anwender an, der die Datei besitzt, und die vierte Spalte die Gruppe. Da die Default-Einstellungen den gleichen Namen für den Anwender und für die Gruppe vorgeben, erscheinen in der obigen Abbildung jedoch stets die gleichen Werte. Das ist allerdings nicht zwingend so. Es wäre auch möglich, andere Bezeichnungen für die Gruppe zu verwenden. Es ist wichtig, in jedem Fall darauf zu achten, dass es sich bei der Angabe in der dritten Spalte um den Anwender handelt und in der vierten Spalte um die Gruppe.

Von Bedeutung ist außerdem die erste Spalte. Hier sehen wir eine kryptisch wirkende Angabe, die eine genaue Erklärung benötigt. Dieser Wert besteht immer aus zehn Zeichen. Das erste gibt an, ob es sich um einen Ordner oder um eine Datei handelt. Der Wert `d` steht für einen Ordner und das Minuszeichen (`-`) für eine Datei. Die nächsten drei Zeichen geben die Rechte für den Anwender vor, der die Datei oder den Ordner besitzt. Dafür kommen drei Werte zum Einsatz: `r` für das Lesen, `w` für das Schreiben und `x` für das Ausführen. Wenn der Anwender über die entsprechenden Rechte verfügt, wird der zugehörige Buchstabe aufgeführt. Sind die Rechte nicht vorhanden, wird ein Minuszeichen gesetzt. Danach folgen an den Positionen 5 bis 7 die Rechte für die Gruppe, die die Datei oder den Ordner besitzt. Das System ist hierbei genau das gleiche. An letzter Stelle folgen die Rechte für alle übrigen Anwender, die die Datei weder besitzen noch zur entsprechenden Gruppe gehören.

Um dieses System besser zu verstehen, erklären wir einmal den Ausdruck `drwxr-xr-x`, der beispielsweise beim Ordner Dokumente angegeben ist. Der erste Buchstabe sagt aus, dass es sich hierbei um einen Ordner handelt. Die folgenden drei Buchstaben (`rw`~~`x`~~) sagen aus, dass der Anwender, der diesen Ordner besitzt, Dateien lesen, schreiben und ausführen darf. Er hat also alle Rechte. Die nächsten drei Zeichen (~~`r`~~-~~`x`~~) beziehen sich auf die Gruppe. Die Mitglieder der Gruppe dürfen die enthaltenen Dateien lesen und ausführen, aber nicht überschrei-

ben. Auch für alle übrigen Anwender gelten die gleichen Rechte. Das zeigen die letzten drei Zeichen (r-x).

Im nächsten Schritt lernen wir, wie wir die Rechte für eine Datei oder einen Ordner ändern können. Hierfür wählen wir die Datei `Beispieltext.txt` aus. Dazu wechseln wir zunächst mit dem `cd`-Befehl in den Ordner `Dokumente`. Daraufhin geben wir zunächst einmal die Rechte für die Dateien aus, um den aktuellen Stand zu überprüfen. Um die Rechte zu ändern, kommt dann der Befehl `chmod` zum Einsatz. Danach müssen wir angeben, welche Rechte wir entfernen oder hinzufügen möchten. Hierfür verwenden wir wieder die drei bekannten Buchstaben `r`, `w` und `x`. Zum Entfernen verwenden wir das Minuszeichen und zum Hinzufügen das Pluszeichen. Danach müssen wir die Datei nennen, auf die wir die Aktion anwenden möchten. Da wir uns bereits im richtigen Ordner befinden, ist es lediglich notwendig, deren Namen aufzuführen. Mit einem passenden Pfadnamen wäre es aber auch möglich, die Rechte für Dateien in anderen Ordnern zu bearbeiten. Wenn wir beispielsweise das Recht zum Lesen der Dateien entfernen möchten, verwenden wir den folgenden Befehl:

```
1 chmod -r Beispieltext.txt
```

Um das Recht zum Ausführen hinzuzufügen, nutzen wir diesen Befehl:

```
1 chmod +x Beispieltext.txt
```



```

user1@user1-Lenovo-V110-15IAP: ~/Dokumente
user1@user1-Lenovo-V110-15IAP:~$ cd Dokumente
user1@user1-Lenovo-V110-15IAP:~/Dokumente$ ls -l
Insgesamt 32
drwxrwxr-x 3 user1 user1 4096 Nov  5 20:43 Beispielordner
-rw-rw-r-- 1 user1 user1 21890 Okt 26 17:05 Beispieltext.odt
-rw-rw-r-- 1 user1 user1  5 Nov  6 21:21 Beispieltext.txt
user1@user1-Lenovo-V110-15IAP:~/Dokumente$ chmod -r Beispieltext.txt
user1@user1-Lenovo-V110-15IAP:~/Dokumente$ ls -l
Insgesamt 32
drwxrwxr-x 3 user1 user1 4096 Nov  5 20:43 Beispielordner
-rw-rw-r-- 1 user1 user1 21890 Okt 26 17:05 Beispieltext.odt
--w--w---- 1 user1 user1  5 Nov  6 21:21 Beispieltext.txt
user1@user1-Lenovo-V110-15IAP:~/Dokumente$ chmod +x Beispieltext.txt
user1@user1-Lenovo-V110-15IAP:~/Dokumente$ ls -l
Insgesamt 32
drwxrwxr-x 3 user1 user1 4096 Nov  5 20:43 Beispielordner
-rw-rw-r-- 1 user1 user1 21890 Okt 26 17:05 Beispieltext.odt
--wx-wx--x 1 user1 user1  5 Nov  6 21:21 Beispieltext.txt
user1@user1-Lenovo-V110-15IAP:~/Dokumente$

```

7

Abb. 7.21 Die Änderung der Rechte

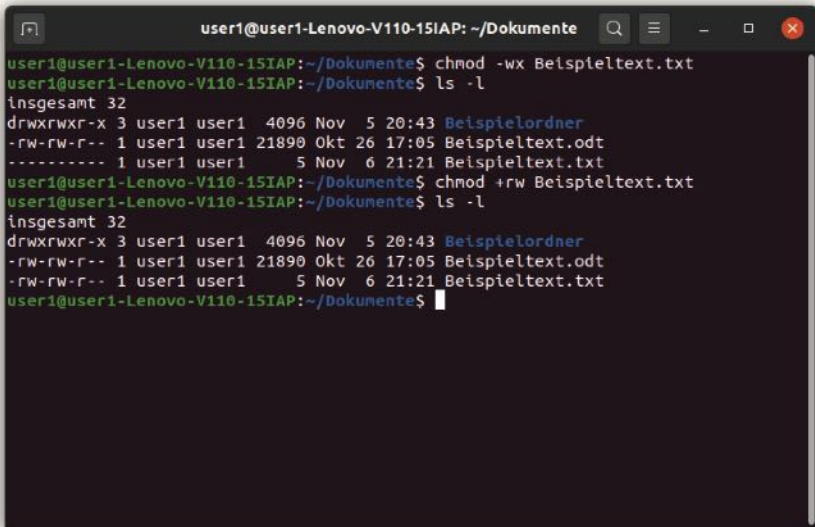
In Abbildung 7.21 ist zu sehen, welche Folgen diese Befehle haben. Zu diesem Zweck geben wir nach jeder Ausführung die Rechte erneut aus. Daran ist zu sehen, dass sich die neuen Vorgaben sowohl auf den Besitzer als auch auf die Gruppe und auf die übrigen Anwender auswirkt. Darüber hinaus zeigt die Abbildung, dass hierbei keine Kontrolle stattfindet, ob die Änderung sinnvoll ist. In unserem Beispiel handelt es sich um eine Textdatei. Diese lässt sich grundsätzlich nicht direkt ausführen. Dennoch können wir die Rechte dazu vergeben, obwohl dies nicht möglich ist.

Hierbei ist es auch möglich, mehrere Rechte gleichzeitig zu entfernen oder hinzuzufügen. Dazu ist es lediglich notwendig, die entsprechenden Buchstaben hintereinander zu setzen. Um beispielsweise das Recht zum Lesen und zum Schreiben zu entziehen, verwenden wir den folgenden Befehl:

```
1 chmod -wx Beispieltext.txt
```

Danach fügen wir die Rechte zum Lesen und zum Schreiben hinzu:

```
1 chmod +rw Beispieltext.txt
```



```
user1@user1-Lenovo-V110-15IAP: ~/Dokumente
user1@user1-Lenovo-V110-15IAP:~/Dokumente$ chmod -wx Beispieltext.txt
user1@user1-Lenovo-V110-15IAP:~/Dokumente$ ls -l
insgesamt 32
drwxrwxr-x 3 user1 user1 4096 Nov  5 20:43 Beispielordner
-rw-rw-r-- 1 user1 user1 21890 Okt 26 17:05 Beispieltext.odt
----- 1 user1 user1    5 Nov  6 21:21 Beispieltext.txt
user1@user1-Lenovo-V110-15IAP:~/Dokumente$ chmod +rw Beispieltext.txt
user1@user1-Lenovo-V110-15IAP:~/Dokumente$ ls -l
insgesamt 32
drwxrwxr-x 3 user1 user1 4096 Nov  5 20:43 Beispielordner
-rw-rw-r-- 1 user1 user1 21890 Okt 26 17:05 Beispieltext.odt
-rw-rw-r-- 1 user1 user1    5 Nov  6 21:21 Beispieltext.txt
user1@user1-Lenovo-V110-15IAP:~/Dokumente$
```

Abb. 7.22 Das Hinzufügen und Entfernen mehrerer Rechte

In Abbildung 7.22 ist zu sehen, dass sich auf diese Weise zwei Bearbeitungsschritte zusammenfassen lassen. Wenn Sie die Auswirkungen des letzten Befehls nochmals genau betrachten, erkennen Sie jedoch, dass die Berechtigung zum Schreiben nur auf den Besitzer und auf die Gruppe angewendet wurde. Die übrigen Anwender erhalten dieses Recht hingegen nicht. Diese Funktion dient der Sicherheit der Daten.

Die bisherige Änderung der Rechte wurde stets auf alle Bereiche angewendet – auf den Besitzer, die Gruppe und mit Ausnahme des Schreib-Rechts auch auf alle übrigen Anwender. Es ist jedoch auch möglich, die Änderungen gezielt umzusetzen. Zu diesem Zweck müssen Sie vor der Angabe, welche Rechte Sie hinzufügen oder entfernen möchten, einen

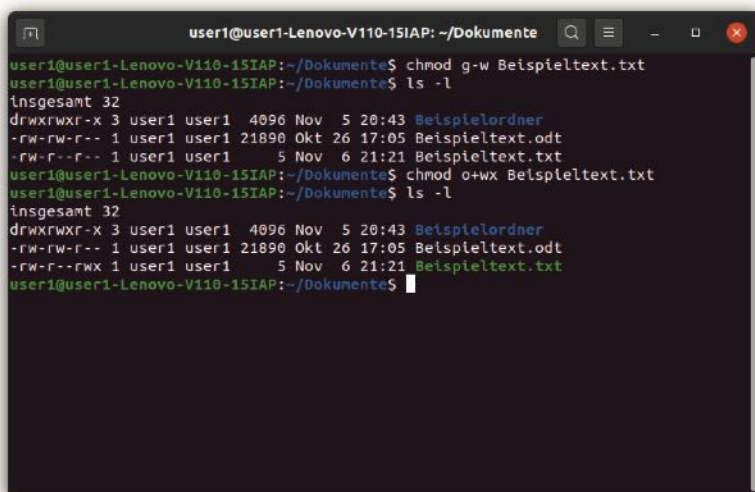
weiteren Buchstaben hinzufügen: *u* für den Besitzer, *g* für die Gruppe und *o* für die übrigen Anwender.

Wollen wir beispielsweise der Gruppe das Recht zum Schreiben der Daten entziehen, können wir dafür den folgenden Befehl verwenden:

```
1 chmod g-w Beispieltext.txt
```

Dabei ist es auch möglich, zwei Rechte gleichzeitig zu benennen. Im Folgenden geben wir allen Anwendern das Recht zum Schreiben und zum Ausführen der Datei. In Abbildung 7.23 ist zu sehen, dass auf diese Weise auch das Schreib-Recht umgesetzt wird, was bei unserem letzten Beispiel nicht der Fall war. Der erforderliche Befehl sieht so aus:

```
1 chmod o+wx Beispieltext.txt
```



```

user1@user1-Lenovo-V110-15IAP: ~/Dokumente
user1@user1-Lenovo-V110-15IAP:~/Dokumente$ chmod g-w Beispieltext.txt
user1@user1-Lenovo-V110-15IAP:~/Dokumente$ ls -l
insgesamt 32
drwxrwxr-x 3 user1 user1 4096 Nov  5 20:43 Beispiellordner
-rw-rw-r-- 1 user1 user1 21890 Okt 26 17:05 Beispieltext.odt
-rw-r--r-- 1 user1 user1  5 Nov  6 21:21 Beispieltext.txt
user1@user1-Lenovo-V110-15IAP:~/Dokumente$ chmod o+wx Beispieltext.txt
user1@user1-Lenovo-V110-15IAP:~/Dokumente$ ls -l
insgesamt 32
drwxrwxr-x 3 user1 user1 4096 Nov  5 20:43 Beispiellordner
-rw-rw-r-- 1 user1 user1 21890 Okt 26 17:05 Beispieltext.odt
-rw-r--rwx 1 user1 user1  5 Nov  6 21:21 Beispieltext.txt
user1@user1-Lenovo-V110-15IAP:~/Dokumente$
  
```

Abb. 7.23 Die Veränderung der Rechte nur für die Gruppe beziehungsweise für alle übrigen Anwender

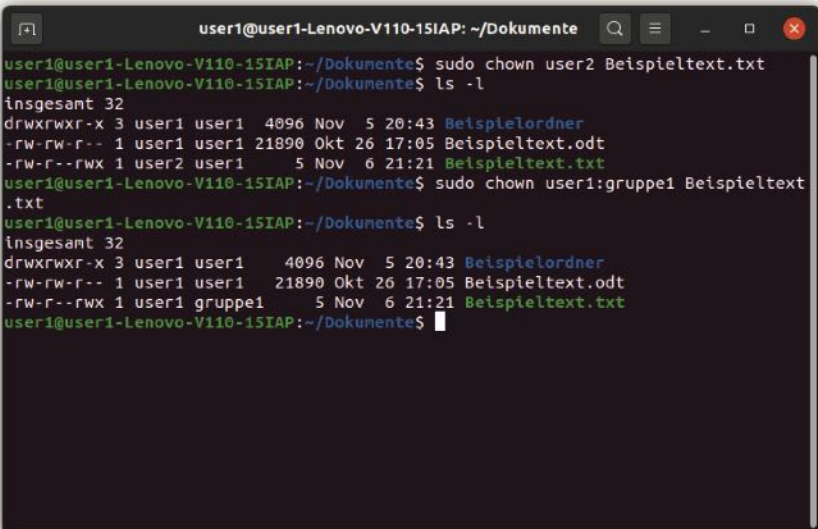
Den Besitzer von Dateien und Ordnern ändern

Eine weitere wichtige Aufgabe bei der Verwaltung der Rechte besteht darin, den Besitzer einer Datei oder eines Ordners zu ändern. Hierfür kommt der Befehl `chown` zum Einsatz – mit vorangestelltem `sudo`. Danach folgen der Name des neuen Besitzers und abschließend der Name der Datei. Wenn wir die Datei `Beispieltext.txt` nun etwa `user2` zuweisen möchten, erledigen wir dies mit dem folgenden Befehl:

```
1 sudo chown user2 Beispieltext.txt
```

Außerdem ist es möglich, die Gruppe neu zu bestimmen. Dazu fügen wir nach dem Namen des Anwenders einen Doppelpunkt ein und geben anschließend die Gruppe an. Mit dem folgenden Befehl weisen wir die Datei wieder `user1` zu. Darüber hinaus legen wir jedoch `gruppe1` als Besitzerin fest:

```
1 sudo chown user1:gruppe1 Beispieltext.txt
```



```
user1@user1-Lenovo-V110-15IAP: ~/Dokumente
user1@user1-Lenovo-V110-15IAP:~/Dokumente$ sudo chown user2 Beispieltext.txt
user1@user1-Lenovo-V110-15IAP:~/Dokumente$ ls -l
insgesamt 32
drwxrwxr-x 3 user1 user1 4096 Nov  5 20:43 Beispielordner
-rw-rw-r-- 1 user1 user1 21890 Okt 26 17:05 Beispieltext.odt
-rw-r--rwx 1 user2 user1    5 Nov  6 21:21 Beispieltext.txt
user1@user1-Lenovo-V110-15IAP:~/Dokumente$ sudo chown user1:gruppe1 Beispieltext
.txt
user1@user1-Lenovo-V110-15IAP:~/Dokumente$ ls -l
insgesamt 32
drwxrwxr-x 3 user1 user1 4096 Nov  5 20:43 Beispielordner
-rw-rw-r-- 1 user1 user1 21890 Okt 26 17:05 Beispieltext.odt
-rw-r--rwx 1 user1 gruppe1    5 Nov  6 21:21 Beispieltext.txt
user1@user1-Lenovo-V110-15IAP:~/Dokumente$
```

Abb. 7.24 Die neuen Besitzer der Datei vorgeben

In Abbildung 7.24 ist zu sehen, dass sich auf diese Weise sowohl ein neuer Anwender als auch eine neue Gruppe als Besitzer festlegen lassen. Es ist allerdings auch möglich, nur eine neue Gruppe vorzugeben. Dazu ist es lediglich notwendig, den Anwender wegzulassen. Vor der Nennung der neuen Gruppe muss aber dennoch ein Doppelpunkt stehen. Mit dem folgenden Befehl können wir beispielsweise die Datei wieder der Gruppe `user1` übergeben:

```
1 sudo chown :user1 Beispieltext.txt
```

7

7.5 Programme installieren

Um ein Programm unter Linux zu installieren, kommt ein sogenannter *Paketmanager* zum Einsatz. Hierbei handelt es sich um ein Programm, das den kompletten Installationsprozess übernimmt. Das beginnt mit dem Herunterladen der Software. Linux-Programme sind in der Regel gratis über das Internet erhältlich. Wenn Sie ein entsprechendes Programm installieren möchten, sucht der Paketmanager eine passende Seite, um es herunterzuladen. Daraufhin kümmert er sich um die Installation und die Konfiguration der Software. Hinsichtlich der Funktionen, die er erfüllt, ist der Paketmanager vergleichbar mit dem Installationsassistenten unter Windows. Ein wesentlicher Unterschied besteht darin, dass die Assistenten unter Windows nur für ein einziges Programm vorgesehen sind. Ein Paketmanager unter Linux ermöglicht es jedoch, viele verschiedene Programme zu installieren.

Es gibt mehrere verschiedene Paketmanager. Jede Distribution ist normalerweise mit einem von ihnen ausgestattet. Zwar ist es auch möglich, einen anderen Paketmanager zu installieren. Das würde jedoch einen erheblichen Zusatzaufwand mit sich bringen. Daher stellt die für diesen Zweck verwendete Software eines der prägenden Elemente einer jeden Distribution dar. Ubuntu verwendet – genau wie Debian, Linux Mint und viele weitere Distributionen – den Paketmanager `apt` (Advanced Package Tool). Dieser ist sehr einfach anzuwenden. Eine weitere beliebte Alternative stellt `yum` dar. Dieser Paketmanager wurde von Red Hat entwickelt und kommt daher bei den meisten Red-Hat Derivaten zum Einsatz. Darüber hinaus gibt es jedoch noch einige weitere Möglichkeiten. In diesem

Lehrbuch beschränken wir uns jedoch auf `apt`. Wenn Sie sich in die Verwendung der übrigen Paketmanager einarbeiten möchten, finden Sie hierzu im Internet verschiedene Tutorials.

Ubuntu verfügt auch über ein Programm mit einer grafischen Benutzeroberfläche, das Sie zu diesem Zweck verwenden können: das Ubuntu Software Center. Dieses ermöglicht eine einfache und intuitive Anwendung. Damit können Sie Ihre Programme ebenfalls problemlos installieren. Das Ubuntu-Software-Center stellt jedoch nur eine grafische Oberfläche für den Paketmanager `apt` dar. Die zugrunde liegenden Funktionen sind genau die gleichen. Daher ist es sinnvoll, sich auch mit der Anwendung des Programms über die Shell zu beschäftigen. Darüber hinaus können Sie auf diese Weise auch auf anderen Distributionen Software installieren, die über keine entsprechenden grafischen Oberflächen verfügen.

Bevor wir mit den praktischen Beispielen beginnen, sei noch erwähnt, dass bei der Verwendung von `apt` verschiedene Befehle infrage kommen: `apt`, `apt-get` und `apt-cache`. Keiner dieser Befehle kommt jedoch alleine zum Einsatz. Für jede Aktion ist ein entsprechender Zusatz notwendig, der angibt, welche Aufgabe zu erledigen ist. Die entsprechenden Begriffe geben lediglich an, dass man sich dabei auf den `apt`-Paketmanager bezieht. Bis 2014 gab es lediglich die Alternativen `apt-get` und `apt-cache`. Diese standen jedoch nicht in Konkurrenz zueinander, sondern ergänzten sich gegenseitig. Für manche Aufgaben war es notwendig, `apt-get` zu verwenden und für andere `apt-cache`. 2014 wurde dann die neue Version `apt` vorgestellt. Dieses ersetzte sowohl `apt-get` als auch `apt-cache`. Das stellt bereits einen der wichtigsten Vorteile und Gründe für die Einführung dieser neuen Alternative dar. Die einheitliche Verwendung von `apt` für alle Aktionen macht es für den Anwender einfacher, Programme zu installieren. Hinzu kommen noch einige kleinere Änderungen und Verbesserungen.

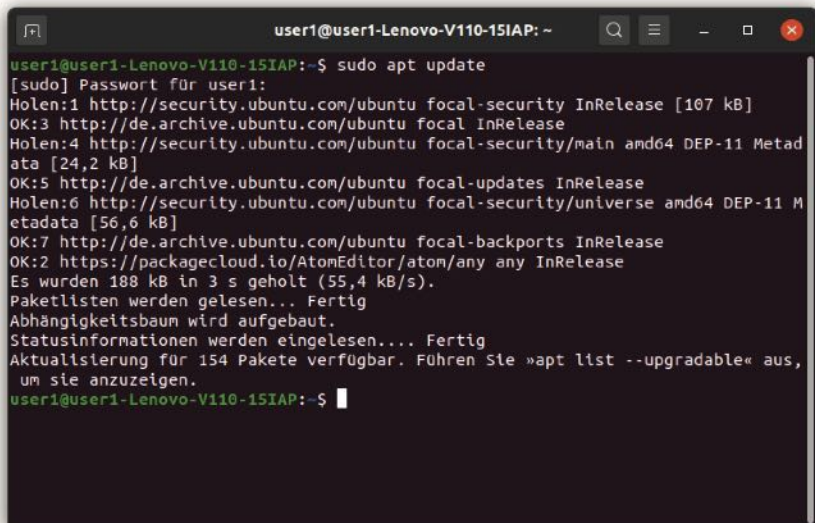
Allerdings werden `apt-get` und `apt-cache` bis heute unterstützt. Das bedeutet, dass Sie diese Befehle nach wie vor verwenden können. Wenn Sie Installationsanweisungen oder Online-Tutorials suchen, werden Sie diese Alternative sicherlich häufig antreffen. In diesem Buch

beschränken wir uns jedoch auf die Verwendung von `apt`, da das die etwas einfachere und modernere Alternative darstellt.

Ein wesentlicher Bestandteil des Paketmanagers besteht in einer Liste, in der alle verfügbaren Programme enthalten sind. Hinzu kommen die Angaben, wo die entsprechenden Programme zum Download erhältlich sind. Diese Daten sind für eine erfolgreiche Installation unverzichtbar.

Diese Liste ist im Internet erhältlich. Allerdings arbeitet `apt` mit einer lokalen Kopie auf dem Rechner. Wenn es hierbei zu einer Überarbeitung kommt, kann das dazu führen, dass die lokale Kopie nicht mehr aktuell ist. Daher besteht der erste Schritt vor der Installation eines Programms stets darin, die Liste zu aktualisieren. Das erreichen wir mit dem Befehl `sudo apt update`. Abbildung 7.25 zeigt, wie das Update der Liste aussehen kann. Die genauen Details hängen dabei jedoch stets davon ab, welche Pakete aktualisiert werden.

7



```

user1@user1-Lenovo-V110-15IAP: ~
user1@user1-Lenovo-V110-15IAP:~$ sudo apt update
[sudo] Passwort für user1:
Holen:1 http://security.ubuntu.com/ubuntu focal-security InRelease [107 kB]
OK:3 http://de.archive.ubuntu.com/ubuntu focal InRelease
Holen:4 http://security.ubuntu.com/ubuntu focal-security/main amd64 DEP-11 Metad
ata [24,2 kB]
OK:5 http://de.archive.ubuntu.com/ubuntu focal-updates InRelease
Holen:6 http://security.ubuntu.com/ubuntu focal-security/universe amd64 DEP-11 M
etaddata [56,6 kB]
OK:7 http://de.archive.ubuntu.com/ubuntu focal-backports InRelease
OK:2 https://packagecloud.io/AtomEditor/atom/any any InRelease
Es wurden 188 kB in 3 s geholt (55,4 kB/s).
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut.
Statusinformationen werden eingelesen.... Fertig
Aktualisierung für 154 Pakete verfügbar. Führen Sie »apt list --upgradable« aus,
um sie anzuzeigen.
user1@user1-Lenovo-V110-15IAP:~$

```

Abb. 7.25 Das Update des Paketmanagers

In Abbildung 7.25 ist auch zu sehen, dass hierbei Aktualisierungen für zahlreiche Pakete verfügbar sind – in diesem Beispiel für 154 Pakete. Die Zahl variiert dabei jedoch stets in Abhängigkeit davon, wann Sie den Update-Befehl zuletzt ausgeführt haben und wie viele Aktualisierungen in diesem Zeitraum erschienen sind. Hierbei handelt es sich um Erweiterungen und Sicherheits-Updates der Programme. Wenn hierbei verfügbare Aktualisierungen angezeigt werden, ist es sinnvoll, diese auch umzusetzen. Zu diesem Zweck verwenden wir den Befehl `sudo apt upgrade`.

Nachdem das System nun auf dem neuesten Stand ist, können wir mit der Installation beginnen. Wenn Sie bereits den genauen Namen des Programms kennen, ist es möglich, dieses direkt zu installieren. Oftmals ist dies jedoch noch nicht bekannt, sodass es sinnvoll ist, eine Suche durchzuführen. Hierfür verwenden wir den Befehl `apt search`. Danach ist es notwendig, einen Suchbegriff anzugeben. Nun gehen wir davon aus, dass wir die Unterhaltung auf dem Gerät etwas verbessern möchten. Zu diesem Zweck wollen wir ein Spiel installieren. Nun wäre es möglich, mit dem sehr allgemeinen Begriff `game` nach einem passenden Angebot zu suchen. Das würde jedoch sehr vielfältige Ergebnisse mit sich bringen. Deshalb präzisieren wir die Suche etwas weiter und suchen nach einem der populärsten Spiele der 90er Jahre: Tetris. Hierfür suchen wir nun mit dem folgenden Befehl eine passende Version für unseren Linux-Rechner: `apt search tetris`.


```

user1@user1-Lenovo-V110-15IAP: ~
Textmodusclient für das tetris-artige Mehrspielerspiel tetrinet

tetrinet-server/focal 0.11+CVS20070911-2build2 amd64
  Server für tetrinet, einen tetrisartigem Spiel für mehrere Spieler

tetrinetx/focal 1.13.16-14build1 amd64
  Spiel-Server für Tetrinet

tint/focal 0.05 amd64
  Tetris clone for text based terminal

vdr-plugin-games/focal 0.6.3-46.1build1 amd64
  VDR plugin providing OSD games like tetris, snake and more

vim-scripts/focal,focal 20180807ubuntu1 all
  Erweiterungen für vim (was man sonst noch so braucht)

vltetris/focal 0.58.0-2 amd64
  Virtual terminal *tris clone

xwelltris/focal 1.0.1-17build1 amd64
  populäres 3D-Tetris, ähnelt Welltris

user1@user1-Lenovo-V110-15IAP: ~$

```

Abb. 7.26 Ein Ausschnitt aus den Suchergebnissen für die Suche nach einem Tetris-Spiel

Abbildung 7.26 zeigt, dass wir hierbei vielfältige Möglichkeiten entdecken. Eine kurze Beschreibung hilft uns bei der Auswahl. Von großer Bedeutung sind auch die Begriffe ganz zu Beginn der Einträge. Diese sind in grüner Farbe dargestellt. Hierbei handelt es sich um den Namen der Pakete. Wenn wir eines der Programme installieren möchten, müssen wir uns die entsprechende Bezeichnung für die Installation merken. Für unser Anwendungsbeispiel entscheiden wir uns für den letzten Eintrag in der Liste: `xwelltris`.

Um das Spiel zu installieren, verwenden wir den Befehl `sudo apt install`. An diesen müssen wir dann lediglich den Namen des Pakets anhängen: `sudo apt install xwelltris`. Daraufhin installiert der Paketmanager das Spiel für uns vollkommen automatisch. Wir müssen den Installationsvorgang lediglich einmal bestätigen. Nach einer kurzen Wartezeit steht das Programm dann bereits zur Verfügung. Wir können es daraufhin entweder über das Startmenü auf-

rufen oder über das Terminal. Zu diesem Zweck müssen wir lediglich den Namen des entsprechenden Pakets eingeben: `xwelltris`.

Wenn wir nach einigem Ausprobieren festgestellt haben, dass die Faszination für Tetris nicht mehr die gleiche wie in den 90er Jahren ist, stellt es auch kein Problem dar, das Programm wieder zu entfernen. Hierfür verwenden wir den Befehl `apt remove`. Diesem stellen wir `sudo` voran und danach geben wir den Namen des Pakets an: `sudo apt remove xwelltris`.

7.6 Wildcards

Bei der Arbeit mit der Shell ist es gelegentlich notwendig, bestimmte Dateien zu finden oder eine Aktion mit mehreren Dateien durchzuführen. Zu diesem Zweck ist es möglich, Wildcards zu verwenden. Damit lassen sich Ausdrücke aufstellen, die in manchen Bereichen feste Vorgaben haben, in anderen Teilen jedoch frei sind. Die Shell bezieht den Befehl dann auf alle Dateien, die den gemachten Angaben entsprechen. Häufig kommen Wildcards zusammen mit dem Befehl `ls` zum Einsatz. Auf diese Weise lassen sich Dateien und Ordner finden, bei denen nicht der komplette Name bekannt ist, sondern nur ein Teilbereich. Eine weitere Möglichkeit besteht darin, Wildcards zusammen mit weiteren Befehlen zu verwenden – beispielsweise mit `rm` oder `cp`. Damit können Sie Inhalte löschen oder kopieren. Insbesondere im ersten Fall ist hierbei jedoch große Vorsicht angebracht. Sobald der Befehl eingegeben ist, werden die entsprechenden Dateien unwiederbringlich gelöscht. Insbesondere bei Ordnern, die viele Dateien enthalten, kann es leicht vorkommen, dass neben den eigentlich beabsichtigten Dateien auch weitere Inhalte der Wildcard entsprechen, sodass sie versehentlich ebenfalls gelöscht werden.

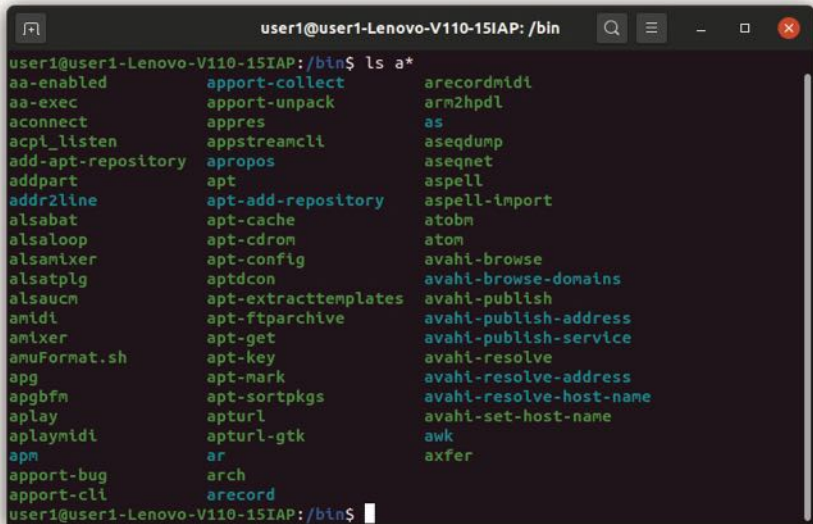
Es gibt mehrere unterschiedliche Wildcards. Dabei sind insbesondere das Sternsymbol, das Fragezeichen sowie eckige und geschweifte Klammern von Bedeutung. Die folgenden Abschnitte stellen diese nacheinander vor.

Das Sternsymbol

Das Sternsymbol steht für eine beliebige Zeichenfolge. Dabei sind weder die Zeichen selbst noch deren Anzahl vorgegeben. Wenn Sie lediglich das Sternsymbol verwenden, trifft dieses daher auf alle vorhandenen Inhalte zu. Daher ist es sinnvoll, hierbei stets einen Zusatz zu verwenden, um die Suche einzugrenzen.

Um die Verwendung an einem sinnvollen Beispiel darzustellen, ist es notwendig, einen Ordner mit zahlreichen verschiedenen Inhalten zu verwenden. Da die Zahl der persönlichen Dateien wahrscheinlich noch gering ist, wechseln wir in einen Ordner, in dem sich Systemdateien befinden: `/bin`. Da wir hier keine Dateien löschen sollten, ist es sinnvoll, zunächst ausschließlich mit dem Befehl `ls` zu arbeiten. Um in diesen Ordner zu wechseln, verwenden wir zunächst diesen Befehl: `cd /bin`.

Danach geben wir den folgenden Befehl ein: `ls a*`. Der Buchstabe 'a', den wir hier angeben, ist ein fester Bestandteil der Inhalte, die hier aufgelistet werden sollen. Dieser muss auf jeden Fall vorhanden sein – genau an der vorgegebenen Stelle. Danach folgt jedoch das Sternsymbol, das für beliebig viele verschiedene Zeichen stehen kann. Das bedeutet, dass hierbei alle Dateien und Ordner aufgelistet werden, die mit dem Buchstaben 'a' beginnen. Abbildung 7.27 zeigt dies.



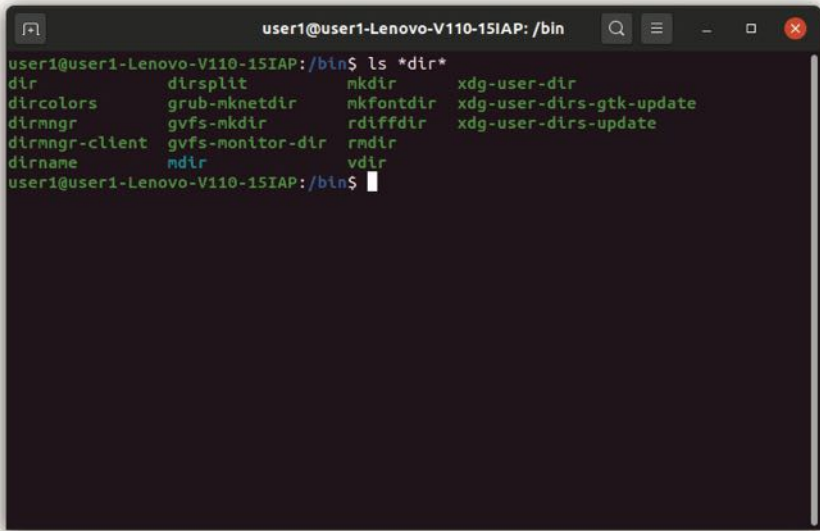
```

user1@user1-Lenovo-V110-15IAP: /bin
user1@user1-Lenovo-V110-15IAP:/bin$ ls a*
aa-enabled          apport-collect      arecordmidi
aa-exec            apport-unpack       arm2hpd1
aconnect           appres             as
acpi_listen        appstreamcli       aseqdump
add-apt-repository apropos            aseqnet
addpart           apt                aspell
addr2line         apt-add-repository aspell-import
alsabat          apt-cache          atobn
alsaloop         apt-cdrom          atom
alsamixer        apt-config         avahi-browse
alsatplg         aptdcon            avahi-browse-domains
alsaucm         apt-extracttemplates avahi-publish
amidi           apt-ftparchive     avahi-publish-address
amixer          apt-get            avahi-publish-service
amuFormat.sh    apt-key            avahi-resolve
apg            apt-mark           avahi-resolve-address
apgbfn         apt-sortpkgs       avahi-resolve-host-name
aplay          apturl             avahi-set-host-name
aplaymidi      apturl-gtk         awk
apm            ar                 axfer
appport-bug    arch
appport-cli    arecord

```

Abb. 7.27 Die Auflistung aller Dateien und Ordner, die mit 'a' beginnen.

Wildcards sind insbesondere dann hilfreich, wenn wir eine Datei suchen, bei der wir uns nicht mehr an den genauen Namen erinnern, bei der aber ein Teil bekannt ist. Beispielsweise haben wir bereits den Befehl zum Erstellen eines Ordners kennengelernt. Jetzt gehen wir davon aus, dass wir uns nicht mehr genau daran erinnern, wie dieser aussieht. Es ist lediglich im Gedächtnis geblieben, dass der Bestandteil „dir“ (für directory) darin enthalten war. Der Ordner bin enthält diese Konsolenbefehle, sodass dieser hier aufgelistet sein müsste. Daher probieren wir es mit der folgenden Suche: `ls *dir*`. In Abbildung 7.28 ist zu sehen, dass nun alle Dateien aufgelistet werden, die diesen Bestandteil enthalten. Indem wir sowohl vor der Zeichenfolge als auch danach das Sternsymbol angebracht haben, kann dieser Bestandteil an einer beliebigen Stelle des Befehls stehen – am Anfang, am Ende oder in der Mitte. Die Abbildung zeigt, dass die Auswahl damit gut eingeschränkt wurde. So können wir alle Möglichkeiten schnell überprüfen und stellen fest, dass es der Befehl `mkdir` war, den wir zum Erstellen eines Verzeichnisses verwendet haben.



```

user1@user1-Lenovo-V110-15IAP: /bin
user1@user1-Lenovo-V110-15IAP:/bin$ ls *dir*
dir          dirsplitt    mkdir        xdg-user-dir
dircolors    grub-mknetdir  mkfontdir    xdg-user-dirs-gtk-update
dirmngr      gvfs-mkdir    rdiffdir     xdg-user-dirs-update
dirmngr-client gvfs-monitor-dir rmdir
dirname      mdir         vdir
user1@user1-Lenovo-V110-15IAP:/bin$

```

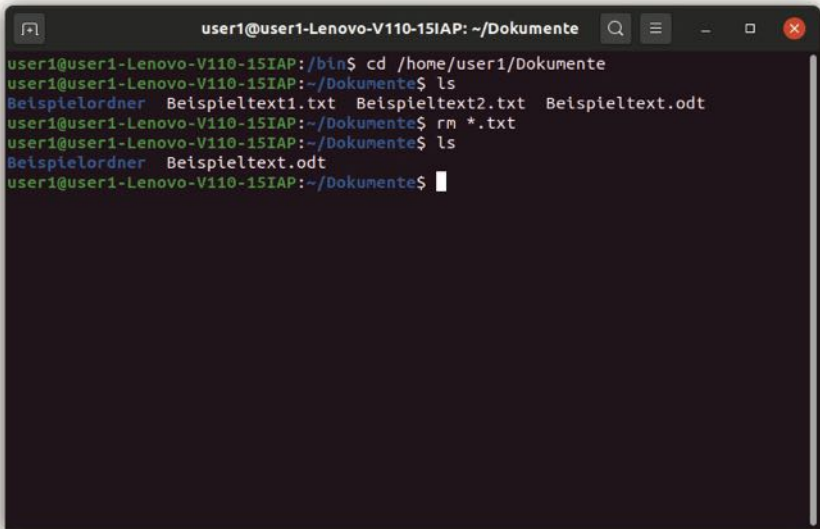
7

Abb. 7.28 Die Suche nach Inhalten, die die Zeichenfolge „dir“ enthalten.

Nun probieren wir die Verwendung von Wildcards noch mit dem Befehl `rm` aus. Da wir jedoch keine Systemdateien löschen möchten, wechseln wir zuvor in unseren persönlichen Ordner. Hier haben wir einige Beispieldateien erstellt – mit den Endungen `.odt` und `.txt`. In unserem Beispiel möchten wir nun nur alle Dateien mit der Endung `.txt` löschen. Das ist mit einer Wildcard ganz einfach möglich:

```
1 rm *.txt
```

Wenn Sie diesen Befehl ebenfalls ausprobieren möchten, sollten Sie darauf achten, dass sich im entsprechenden Ordner keine wichtigen Dateien mit dieser Endung befinden. Am besten ist es, hierfür ein neues Verzeichnis mit einigen Test-Dateien zu gestalten, die problemlos gelöscht werden können.



```
user1@user1-Lenovo-V110-15IAP: ~/Dokumente
user1@user1-Lenovo-V110-15IAP:~/bin$ cd /home/user1/Dokumente
user1@user1-Lenovo-V110-15IAP:~/Dokumente$ ls
Beispielordner  Beispieltext1.txt  Beispieltext2.txt  Beispieltext.odt
user1@user1-Lenovo-V110-15IAP:~/Dokumente$ rm *.txt
user1@user1-Lenovo-V110-15IAP:~/Dokumente$ ls
Beispielordner  Beispieltext.odt
user1@user1-Lenovo-V110-15IAP:~/Dokumente$
```

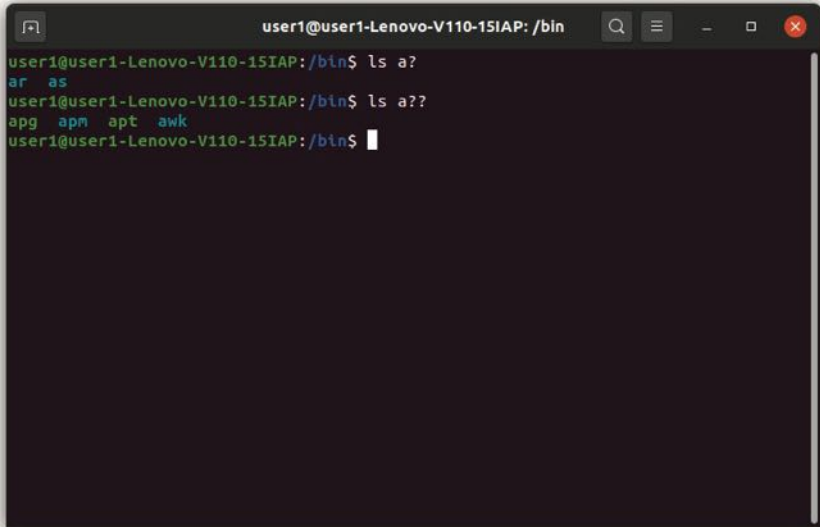
Abb. 7.29 Das Löschen aller Dateien mit der Endung `.txt`.

In Abbildung 7.29 ist zu sehen, dass wir auf diese Weise mit einem einzigen Befehl alle Dateien mit der entsprechenden Endung entfernen können. Die übrigen Dateien und Ordner sind dabei hingegen nicht betroffen.

Das Fragezeichen

Eine weitere Wildcard wird durch das Fragezeichen symbolisiert. Dieses steht ebenfalls für ein beliebiges Zeichen. Die Anzahl ist hierbei jedoch im Gegensatz zum Sternsymbol genau vorgegeben. Es handelt sich hierbei um genau ein Zeichen – nicht mehr und nicht weniger. Um dies auszuprobieren, wechseln wir wieder zum Ordner `/bin`. Hier geben wir nun den Befehl `ls a?` ein. In Abbildung 7.30 ist zu sehen, dass hierbei genau wie beim Sternsymbol nur Inhalte angezeigt werden, die mit dem Buchstaben ‘a’ beginnen. Allerdings bestehen die Namen aller aufgeführten Dateien und Ordner aus genau zwei Buchstaben. Im nächs-

ten Beispiel fügen wir das Fragezeichen zweimal ein: `ls a??`. Nun erscheinen nur Ergebnisse, die aus drei Zeichen bestehen. Die Inhalte, die aus zwei Buchstaben bestehen, sind hingegen nicht aufgeführt. Das zeigt, dass die Anzahl der Zeichen, für die das Fragezeichen steht, genau 1 betragen muss und auch nicht bei 0 liegen darf.

A terminal window titled 'user1@user1-Lenovo-V110-15IAP: /bin' with search, menu, and window control icons. It shows two commands: 'ls a?' which outputs 'ar as', and 'ls a??' which outputs 'apg apm apt awk'.

```
user1@user1-Lenovo-V110-15IAP: /bin$ ls a?  
ar as  
user1@user1-Lenovo-V110-15IAP: /bin$ ls a??  
apg apm apt awk  
user1@user1-Lenovo-V110-15IAP: /bin$
```

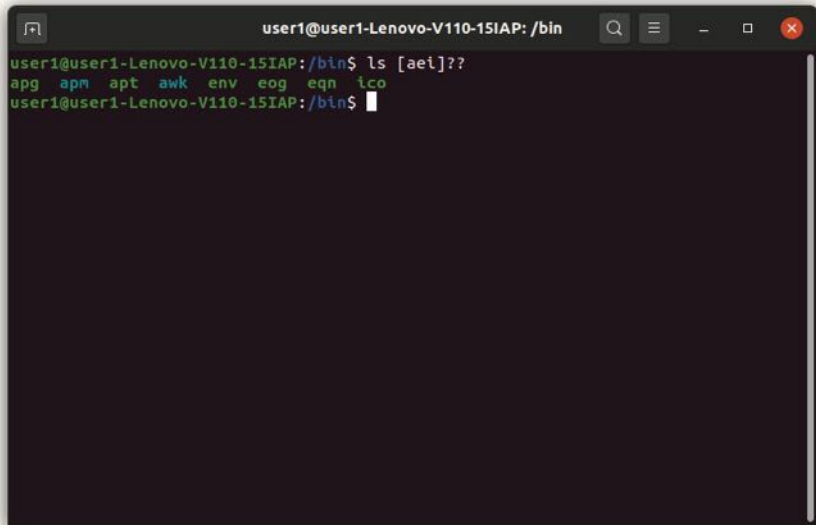
7

Abb. 7.30 Die Verwendung des Fragezeichens als Wildcard

Eckige Klammern

Eine weitere Möglichkeit besteht darin, eckige Klammern als Wildcard zu verwenden. In diese können wir beliebig viele verschiedene Zeichen einfügen. Alle Ausdrücke, bei denen eines der aufgeführten Zeichen an der entsprechenden Stelle steht, werden daraufhin aufgelistet. Dabei ist es wichtig, darauf zu achten, dass jede eckige Klammer nur für ein einzelnes Zeichen steht. Zeichenfolgen können wir damit nicht vorgeben. Meistens kommen die geschweiften Klammern gemeinsam mit einer anderen Wildcard zum Einsatz.

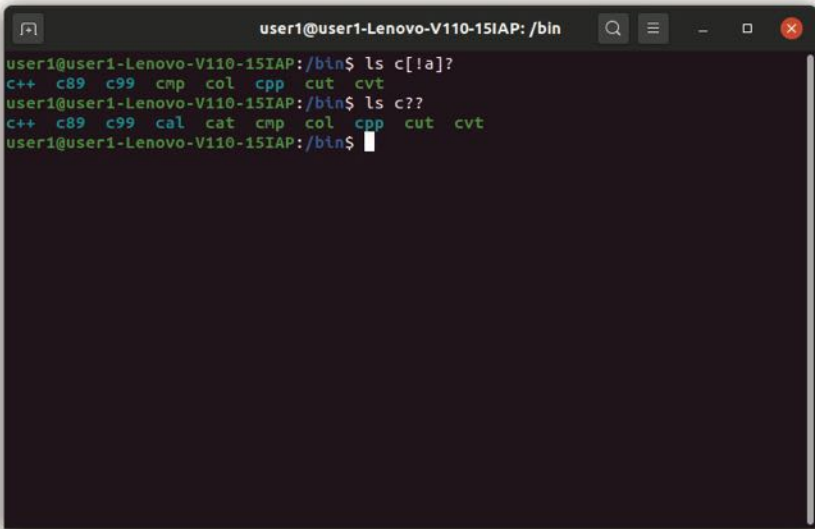
Das zeigt das folgende Beispiel. Darin geben wir in der eckigen Klammer die Buchstaben 'a', 'e' und 'i' an. Danach setzen wir zwei Fragezeichen. Auf diese Weise geben wir alle Inhalte aus, die mit a, e oder i beginnen und aus drei Buchstaben bestehen. Das ist in Abbildung 7.31 zu sehen.



```
user1@user1-Lenovo-V110-15IAP: /bin$ ls [aei]??
apg  apm  apt  awk  env  eog  eqn  ico
user1@user1-Lenovo-V110-15IAP: /bin$
```

Abb. 7.31 Inhalte, die mit a', 'e' oder 'i' beginnen und aus drei Buchstaben bestehen.

Es ist auch möglich, in der eckigen Klammer vorzugeben, welches Zeichen nicht vorhanden sein soll. Dazu kommt das Ausrufungszeichen zum Einsatz. Danach stellen wir das Zeichen, das an dieser Stelle nicht stehen darf. Auf diese Weise können wir beispielsweise alle Ausdrücke suchen, die aus drei Buchstaben bestehen, bei denen an erster Stelle der Buchstabe 'c' steht und bei denen an zweiter Stelle nicht das Zeichen 'a' stehen darf: `ls c[!a]??`. Als Vergleich geben wir auch alle Inhalte mit drei Zeichen aus, die mit 'c' beginnen und bei denen an zweiter Stelle ein beliebiger Wert stehen darf: `ls c??`. In Abbildung 7.32 wird der Unterschied deutlich.

A terminal window titled 'user1@user1-Lenovo-V110-15IAP: /bin' with search, menu, and window control icons. It shows two 'ls' commands. The first, 'ls c[!a]?', lists files 'c++', 'c89', 'c99', 'cmp', 'col', 'cpp', 'cut', and 'cvt'. The second, 'ls c??', lists the same files. The prompt 'user1@user1-Lenovo-V110-15IAP: /bin\$' is shown at the end of each line.

```
user1@user1-Lenovo-V110-15IAP: /bin$ ls c[!a]?
c++  c89  c99  cmp  col  cpp  cut  cvt
user1@user1-Lenovo-V110-15IAP: /bin$ ls c??
c++  c89  c99  cal  cat  cmp  col  cpp  cut  cvt
user1@user1-Lenovo-V110-15IAP: /bin$
```

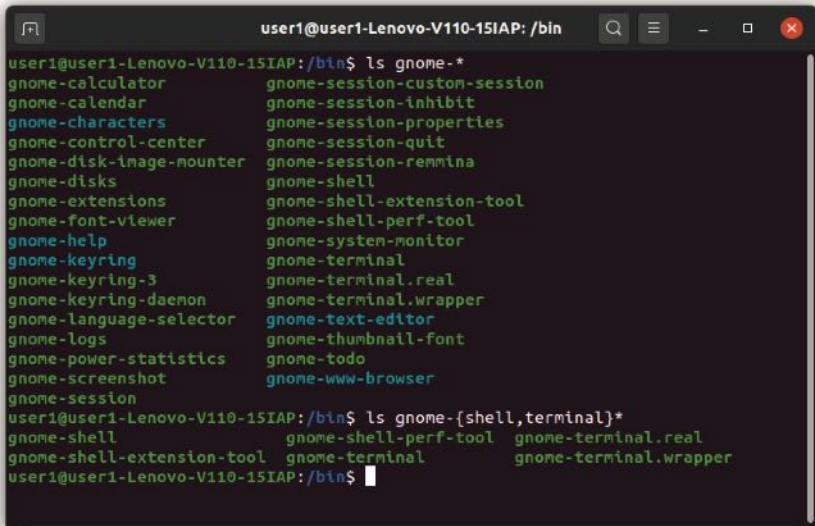
Abb. 7.32 Die Suche nach Dateien, in denen ein bestimmter Buchstabe nicht vorhanden sein soll.

Geschweifte Klammern

Eine weitere Möglichkeit besteht darin, geschweifte Klammern zu verwenden. Darin ist es möglich, verschiedene Zeichenfolgen vorzugeben. Diese müssen durch ein Komma voneinander getrennt werden. Der Befehl bezieht sich dann auf alle Inhalte, die eine der genannten Zeichenfolgen an der entsprechenden Stelle aufweisen.

Um dies zu zeigen, lassen wir uns zunächst mit dem bereits bekannten Sternsymbol alle Inhalte anzeigen, die mit dem Ausdruck „gnome-“ beginnen: `ls gnome-*`. Dabei handelt es sich um Inhalte, die zur von uns verwendeten Desktopoberfläche gehören. In Abbildung 7.33 sehen wir, dass nach diesem Begriff weitere Ausdrücke folgen, die sich als Unterkategorien verstehen lassen. Beispielsweise sehen wir, dass bei mehreren Dateien der Begriff `terminal` oder der Ausdruck `shell` folgt. Darüber hinaus gibt es jedoch noch viele weitere Möglichkeiten.

Nun gehen wir jedoch davon aus, dass gerade die Bereiche Terminal und Shell für uns von Interesse sind. Mit den geschweiften Klammern können wir die Suche auf diese beiden Zeichenfolgen beschränken: `ls gnome-{shell,terminal}*`. Auch das Ergebnis dieses Befehls ist in der unten stehenden Abbildung zu sehen.



```

user1@user1-Lenovo-V110-15IAP: /bin
user1@user1-Lenovo-V110-15IAP:/bin$ ls gnome-*
gnome-calculator      gnome-session-custom-session
gnome-calendar         gnome-session-inhibit
gnome-characters       gnome-session-properties
gnome-control-center   gnome-session-quit
gnome-disk-image-mounter gnome-session-remmina
gnome-disks            gnome-shell
gnome-extensions       gnome-shell-extension-tool
gnome-font-viewer      gnome-shell-perf-tool
gnome-help             gnome-system-monitor
gnome-keyring          gnome-terminal
gnome-keyring-3        gnome-terminal.real
gnome-keyring-daemon   gnome-terminal.wrapper
gnome-language-selector gnome-text-editor
gnome-logs             gnome-thumbnail-font
gnome-power-statistics gnome-todo
gnome-screenshot       gnome-www-browser
gnome-session
user1@user1-Lenovo-V110-15IAP:/bin$ ls gnome-{shell,terminal}*
gnome-shell      gnome-shell-perf-tool  gnome-terminal.real
gnome-shell-extension-tool  gnome-terminal      gnome-terminal.wrapper
user1@user1-Lenovo-V110-15IAP:/bin$
  
```

Abb. 7.33 Der Einsatz der geschweiften Klammern für die Angabe verschiedener Zeichenfolgen.

7.7 Pipes und Filter

Zu den wesentlichen Eigenschaften aller unixoiden Betriebssysteme zählt, dass diese Pipes unterstützen. Diese Funktionsweise wurde bereits 1973 zum UNIX-Betriebssystem hinzugefügt. Verantwortlich hierfür war der Informatiker Douglas McIlroy. Pipes bieten die Möglichkeit, zwei verschiedene Prozesse miteinander zu verbinden. Zwar haben mittlerweile auch die meisten anderen gängigen Betriebssysteme Pipes übernommen. Unter UNIX und allen UNIX-artigen Betriebssystemen

ist dieses Werkzeug jedoch ausgesprochen mächtig und kommt daher häufig zur Verwendung.

Was ist eine Pipe?

Die meisten Computerprogramme nutzen eine Eingabe und eine Ausgabe. Wenn es sich hierbei um Konsolenprogramme handelt, erfolgen die Eingabe und die Ausgabe meistens ebenfalls über die Konsole – also über die Tastatur und den Monitor. Es bestehen jedoch noch einige weitere Möglichkeiten. Eingaben und Ausgaben können beispielsweise über Dateien oder Netzwerke erfolgen.

7

Die Programme können hierbei den Standard-Input (stdin) und den Standard-Output (stdout) verwenden. Zu diesem Zweck erzeugt die Shell sowohl einen stdin-Stream als auch einen stdout-Stream. Wenn wir keine anderen Vorgaben machen, sind diese ebenfalls mit der Shell verbunden und lesen die entsprechenden Tastatureingaben ein oder geben die Ergebnisse auf dem Monitor aus. Da es sich bei den einzelnen Shell-Befehlen ebenfalls um Programme handelt, können diese ebenfalls die entsprechenden Streams nutzen.

Es wurde gesagt, dass die Ein- und die Ausgabe in der Regel über die Shell erfolgen. Das ist aber nicht immer so. Wir können den Datenstrom auch umleiten. Ein Beispiel hierfür stellt die Verwendung einer Pipe dar. Eine Pipe dient dazu, mehrere Programme miteinander zu verbinden. Zu diesem Zweck leitet sie die Streams um und verknüpft sie. Das bedeutet, dass der Output-Stream des ersten Programms zum Input-Stream des zweiten Programms wird.

Zwei Prozesse miteinander verbinden

Um Pipes besser zu verstehen, soll deren Funktionsweise nun an einigen Beispielen verdeutlicht werden. Zu diesem Zweck benötigen wir zunächst eine kleine Beispieldatei, die viele Textzeilen enthält. Zu diesem Zweck öffnen wir den Texteditor und geben einen passenden Inhalt ein. Um das folgende Beispiel nachzustellen, sollten mindestens 50 Zeilen

vorhanden sein. Etwa die Hälfte davon sollte den Buchstaben a enthalten, die andere Hälfte hingegen nicht. Außerdem sollten die Zeilen nicht alphabetisch geordnet sein.

Um die Funktionsweise der Pipes zu demonstrieren, benötigen wir außerdem einige neue Befehle. Wir beginnen mit dem Kommando `more`. Falls wir daran den Namen einer Datei anschließen, gibt es deren Inhalte aus. Wenn wir die eben erstellte Datei unter dem Namen `bsp1.txt` abgespeichert haben, führt das zu folgendem Befehl: `more bsp1.txt`. Der Inhalt der Datei wird dabei zum Input-Stream. Der Output erfolgt hingegen direkt über die Konsole. Der Befehl `more` führt dazu, dass nicht alle Inhalte auf einmal ausgegeben werden. Das würde bei langen Texten dazu führen, dass man nach der Ausgabe wieder nach oben scrollen muss, um die ersten Zeilen zu betrachten. Durch die Verwendung des `more`-Befehls werden hingegen zunächst nur die Zeilen ausgegeben, die in das aktuelle Fenster passen. Durch die Betätigung einer beliebigen Taste kann man dann den nächsten Bereich abrufen – so wie dies in Abbildung 7.34 zu sehen ist.

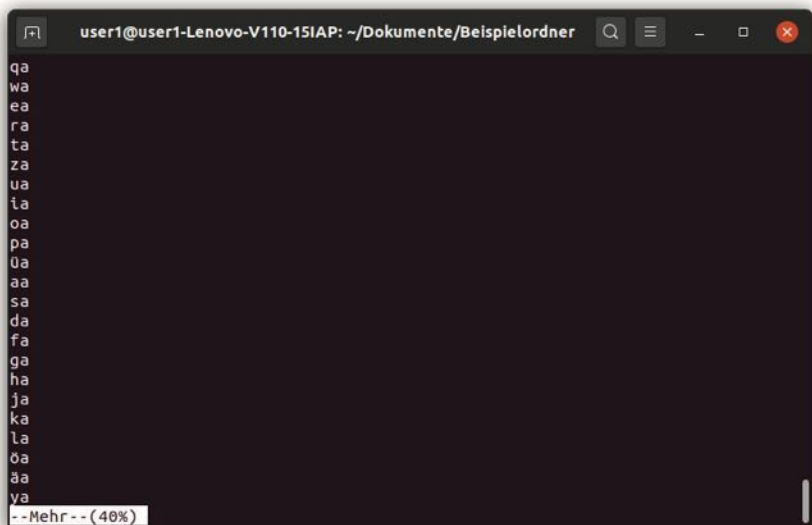
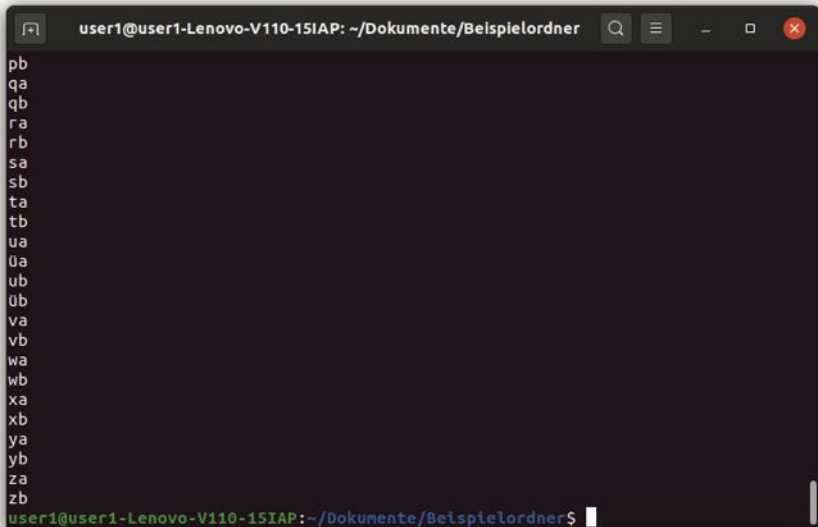


Abb.7.34 Die stückweise Ausgabe der Inhalte

Außerdem verwenden wir den Befehl `sort`. Dieser dient dazu, einzelne Zeilen alphabetisch zu ordnen. Dieser Befehl zählt in den Bereich der Filter – obwohl er streng genommen die Ergebnisse nicht filtert, sondern sortiert. Pipes kommen sehr häufig gemeinsam mit Filtern zum Einsatz. Daher werden diese beiden Bereiche oftmals auch gemeinsam beschrieben – so wie dies auch in diesem Kapitel der Fall ist. Mit dem folgenden Befehl geben wir nun den Inhalt unserer Beispieldatei sortiert aus: `sort bsp1.txt`.

A terminal window with a dark background. The title bar reads 'user1@user1-Lenovo-V110-15IAP: ~/Dokumente/Beispielordner'. The terminal displays a list of two-letter combinations sorted alphabetically: pb, qa, qb, ra, rb, sa, sb, ta, tb, ua, qa, ub, vb, va, vb, wa, wb, xa, xb, ya, yb, za, zb. The prompt 'user1@user1-Lenovo-V110-15IAP: ~/Dokumente/Beispielordner\$' is visible at the bottom.

```
user1@user1-Lenovo-V110-15IAP: ~/Dokumente/Beispielordner
pb
qa
qb
ra
rb
sa
sb
ta
tb
ua
qa
ub
vb
va
vb
wa
wb
xa
xb
ya
yb
za
zb
user1@user1-Lenovo-V110-15IAP: ~/Dokumente/Beispielordner$
```

Abb. 7.35 Die sortierten Zeilen der Beispieldatei

In Abbildung 7.35 ist zu sehen, dass die Inhalte nun sortiert sind. Allerdings werden sie alle auf einmal ausgegeben. Das macht es schwierig, den Anfang zu betrachten. Nun ist es jedoch auch möglich, die beiden Befehle miteinander zu verbinden. Dazu verwenden wir zunächst den `sort`-Befehl. Danach fügen wir einen senkrechten Strich (`|`) ein. Anschließend folgt der Befehl `more`. Das führt dazu, dass der Output-Stream des `sort`-Befehls zum Input-Stream des `more`-Befehls wird. In Abbildung 7.36 ist zu sehen, dass dadurch die sortierte Liste Stück

für Stück ausgegeben wird. Der komplette Befehl sieht dann so aus:
`sort bsp1.txt | more.`

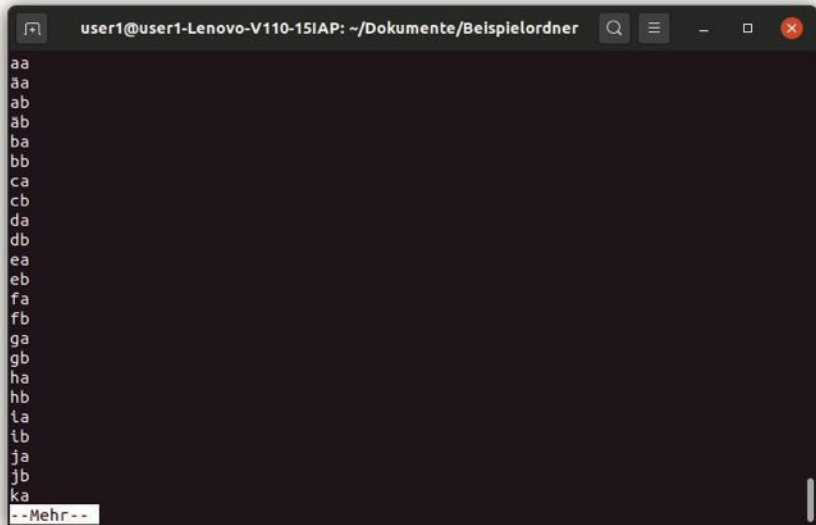
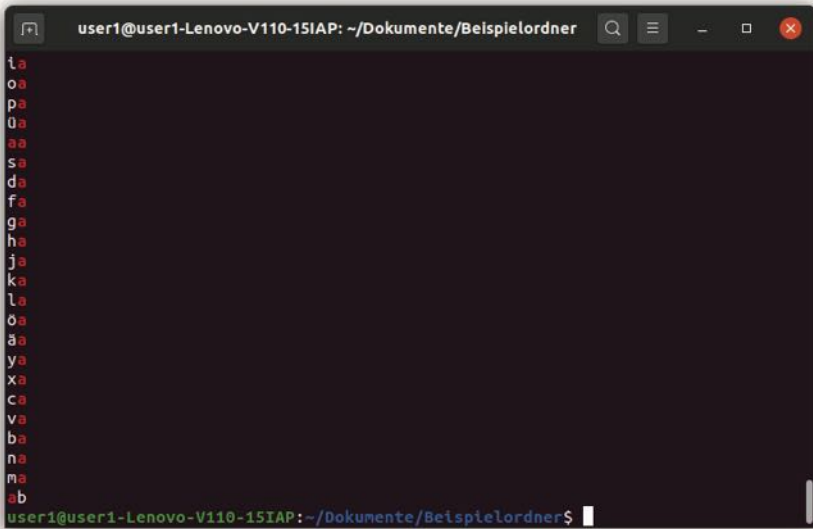


Abb. 7.36 Die Kombination aus `sort-` und `more-`Befehl

Da die Verwendung von Pipes sehr wichtig für die Arbeit mit Linux ist, stellen wir nun noch ein weiteres Beispiel vor. Dazu verwenden wir den Befehl `grep`. Dabei handelt es sich um einen weiteren Filterbefehl. Nach dem Ausdruck `grep` geben wir zunächst eine Zeichenfolge an. Der Befehl gibt daraufhin nur die Zeilen aus, die diese Zeichenfolge enthalten. Wenn wir dieses Kommando für sich alleine verwenden, müssen wir hinzufügen, welche Daten es filtern soll. Dabei müssen wir eine Datei angeben. Um dies auszuprobieren, filtern wir zunächst einmal aus der Beispieldatei, die wir im vorigen Beispiel verwendet haben, alle Zeilen heraus, die den Buchstaben 'a' enthalten: `grep a bsp1.txt`. Das Ergebnis ist in Abbildung 7.37 zu sehen.

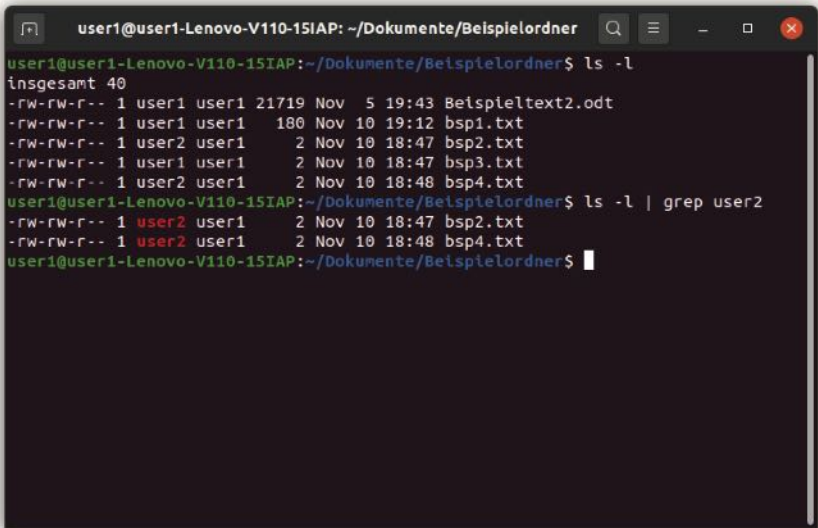


```
user1@user1-Lenovo-V110-15IAP: ~/Dokumente/Beispielordner
ta
oa
pa
ua
aa
sa
da
fa
ga
ha
ja
ka
la
oa
aa
ya
xa
ca
va
ba
na
ma
ab
user1@user1-Lenovo-V110-15IAP:~/Dokumente/Beispielordner$
```

Abb. 7.37 Der Befehl `grep` filtert Zeilen nach einer bestimmten Zeichenfolge.

Für unser Beispiel ist es nun notwendig, in den aktuellen Ordner noch einige weitere Dateien einzufügen. Der Inhalt ist hierbei nicht von Bedeutung. Allerdings sollten zwei der Dateien dem Anwender `user2` gehören. Diese Aufgabe erledigen Sie mit dem Befehl `chown` – so wie dies in Kapitel 7.4 vorgestellt wurde. Im ersten Schritt geben wir die Dateien mit dem zugehörigen Besitzer über den Befehl `ls -l` aus, um zu kontrollieren, ob unsere Änderungen erfolgreich waren.

Für unser Beispiel wollen wir nun jedoch nur die Dateien anzeigen lassen, die `user2` gehören. Zu diesem Zweck verbinden wir die Befehle `ls` und `grep` über eine pipe: `ls -l | grep user2`.

A terminal window titled 'user1@user1-Lenovo-V110-15IAP: ~/Dokumente/Beispielordner'. The first command is 'ls -l', which lists files with permissions, owner, group, size, date, and filename. The output shows four files: 'Beispieltext2.odt', 'bsp1.txt', 'bsp2.txt', and 'bsp3.txt'. The second command is 'ls -l | grep user2', which filters the output to show only files owned by 'user2'. The output shows 'bsp2.txt' and 'bsp4.txt'.

```
user1@user1-Lenovo-V110-15IAP: ~/Dokumente/Beispielordner$ ls -l
insgesamt 40
-rw-rw-r-- 1 user1 user1 21719 Nov  5 19:43 Beispieltext2.odt
-rw-rw-r-- 1 user1 user1 180 Nov 10 19:12 bsp1.txt
-rw-rw-r-- 1 user2 user1  2 Nov 10 18:47 bsp2.txt
-rw-rw-r-- 1 user1 user1  2 Nov 10 18:47 bsp3.txt
-rw-rw-r-- 1 user2 user1  2 Nov 10 18:48 bsp4.txt
user1@user1-Lenovo-V110-15IAP:~/Dokumente/Beispielordner$ ls -l | grep user2
-rw-rw-r-- 1 user2 user1  2 Nov 10 18:47 bsp2.txt
-rw-rw-r-- 1 user2 user1  2 Nov 10 18:48 bsp4.txt
user1@user1-Lenovo-V110-15IAP:~/Dokumente/Beispielordner$
```

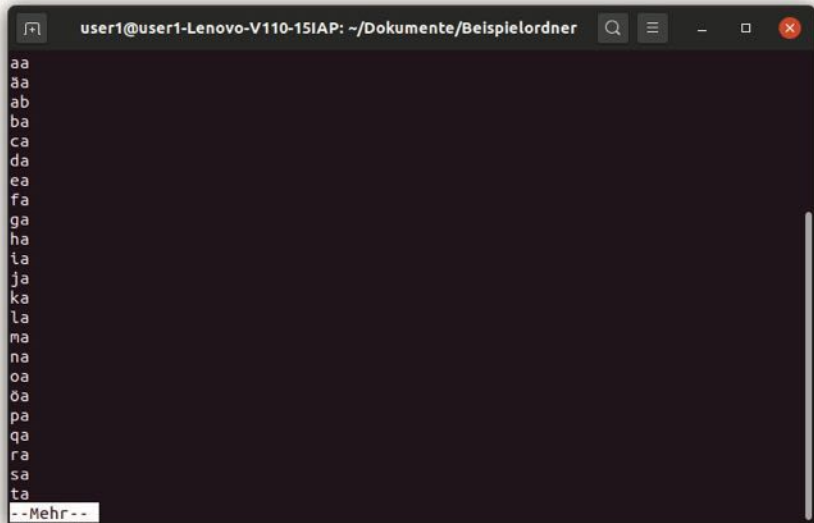
Abb. 7.38 Die Verbindung der Befehle `ls` und `grep`

In Abbildung 7.38 ist zu sehen, dass hierbei die Ergebnisse des `ls`-Befehls so gefiltert werden, dass nur die Zeilen wiedergegeben werden, die den Ausdruck „user2“ enthalten. Das ist möglich, da durch die Pipe der Output des `ls`-Befehls zum Input des `grep`-Befehls wird.

Drei oder mehr Befehle miteinander verbinden

Pipes ermöglichen es nicht nur, eine Verbindung zwischen zwei Befehlen herzustellen. Wir können damit auch drei oder beliebig viele weitere Kommandos zusammenfügen. Das soll nun an einem weiteren Beispiel gezeigt werden. Zu diesem Zweck verbinden wir die Befehle `grep`, `sort` und `more`. Diese wenden wir wieder auf die Datei aus dem ersten Beispiel an. Zunächst filtern wir alle Zeilen heraus, die den Buchstaben ‘a’ enthalten. Daraufhin sortieren wir diese alphabetisch. Abschließend geben wir sie Stück für Stück mit dem `more`-Befehl aus: `grep a bsp1.txt | sort | more`. In Abbildung 7.39 ist zu sehen,

dass wir auf diese Weise alle drei Aufgaben in einer Befehlszeile zusammenfassen können.



7

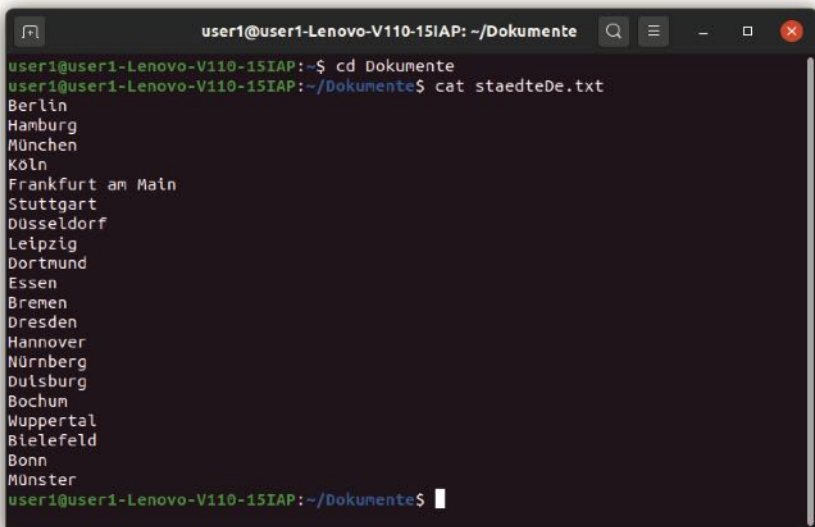
Abb. 7.39 Die Verbindung der Befehle `grep`, `sort` und `more` über eine Pipe.

7.8 Die Befehle cat, head und tail: Inhalte von Dateien anzeigen

Eine wichtige Aufgabe bei der Arbeit mit der Shell besteht darin, die Inhalte von Dateien anzuzeigen und zu bearbeiten. In Kapitel 7.2 haben wir zu diesem Zweck bereits den Befehl `cat` kennengelernt. Indem wir an diesen den Namen einer Datei angehängt haben, war es möglich, deren Inhalt anzeigen zu lassen. Darüber hinaus bietet dieser Befehl jedoch noch viele weitere Möglichkeiten. Hinzu kommen die Kommandos `head` und `tail`, mit denen sich die Darstellung der Inhalte an verschiedene Anforderungen anpassen lässt.

Der Befehl `cat`: Dateien ausgeben, zusammenfügen und kopieren

Um die Arbeit mit dem `cat`-Befehl vorzustellen, ist es wieder notwendig, eine kleine Textdatei zu gestalten. Zu diesem Zweck erstellen wir mit `gedit` eine Liste der 20 größten Städte in Deutschland. Diese schreiben wir jeweils in eine eigene Zeile und speichern sie als Datei mit dem Namen `staedteDe.txt` ab. Wie wir diese Inhalte mit dem `cat`-Befehl ausgeben können, haben wir bereits in Kapitel 7.2 gelernt. Als Wiederholung wechseln wir zunächst in den Ordner, in dem wir die Datei abgespeichert haben und führen daraufhin den folgenden Befehl aus: `cat staedteDe.txt`. Das Ergebnis ist in Abbildung 7.40 zu sehen.



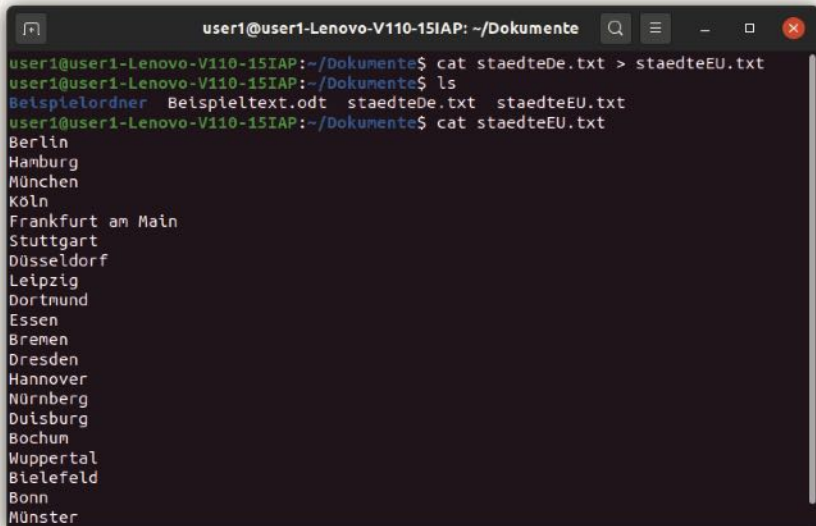
```
user1@user1-Lenovo-V110-15IAP: ~/Dokumente
user1@user1-Lenovo-V110-15IAP:~$ cd Dokumente
user1@user1-Lenovo-V110-15IAP:~/Dokumente$ cat staedteDe.txt
Berlin
Hamburg
München
Köln
Frankfurt am Main
Stuttgart
Düsseldorf
Leipzig
Dortmund
Essen
Bremen
Dresden
Hannover
Nürnberg
Duisburg
Bochum
Wuppertal
Bielefeld
Bonn
Münster
user1@user1-Lenovo-V110-15IAP:~/Dokumente$
```

Abb. 7.40 Die Ausgabe der Inhalte der Datei

Darüber hinaus kommt der Befehl `cat` häufig zum Einsatz, um Inhalte zu kopieren. Sie lassen sich dabei ganz einfach in eine neue Datei schreiben. Dazu müssen wir lediglich das Größer-Zeichen (`>`) nach dem Namen der Datei angeben und daraufhin einen weiteren Dateinamen hinzufügen. Der `cat`-Befehl erstellt daraufhin automatisch eine Datei mit der angegebenen Bezeichnung und führt die entspre-

chenden Inhalte darin ein. Um das zu demonstrieren, wollen wir eine weitere Datei für die Städte in der EU erstellen. Im ersten Schritt erstellen wir diese Datei und fügen die Einträge aus unserer Liste für die Städte in Deutschland darin ein. Der Befehl kann wie folgt aussehen: `cat staedteDe.txt > staedteEU.txt`. Wenn wir danach den `ls`-Befehl ausführen, erkennen wir, dass eine neue Datei mit der entsprechenden Bezeichnung entstanden ist. Wenn wir diese wieder mit dem `cat`-Befehl ausgeben, sehen wir außerdem, dass diese genau den gleichen Inhalt wie die Datei `staedteDe.txt` hat. Das ist in Abbildung 7.41 zu erkennen.

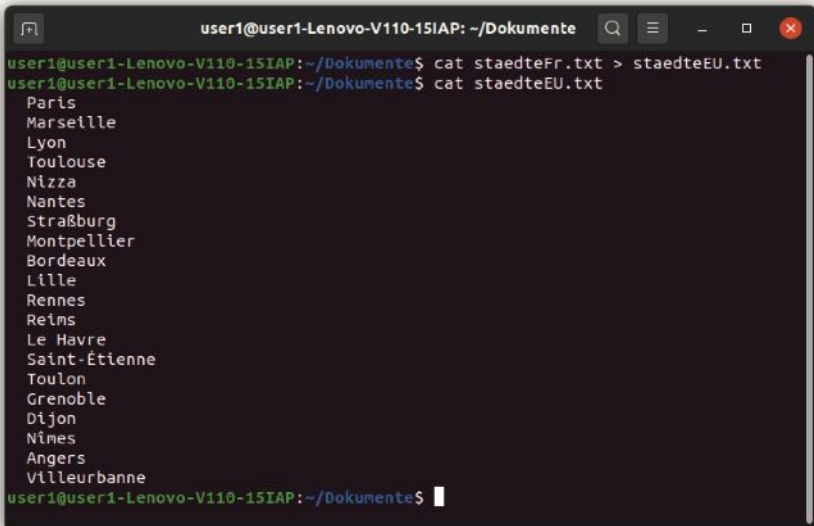
7



```
user1@user1-Lenovo-V110-15IAP: ~/Dokumente
user1@user1-Lenovo-V110-15IAP:~/Dokumente$ cat staedteDe.txt > staedteEU.txt
user1@user1-Lenovo-V110-15IAP:~/Dokumente$ ls
Beispielordner  Beispieltext.odt  staedteDe.txt  staedteEU.txt
user1@user1-Lenovo-V110-15IAP:~/Dokumente$ cat staedteEU.txt
Berlin
Hamburg
München
Köln
Frankfurt am Main
Stuttgart
Düsseldorf
Leipzig
Dortmund
Essen
Bremen
Dresden
Hannover
Nürnberg
Duisburg
Bochum
Wuppertal
Bielefeld
Bonn
Münster
```

Abb. 7.41 Eine Datei mit dem `cat`-Befehl kopieren

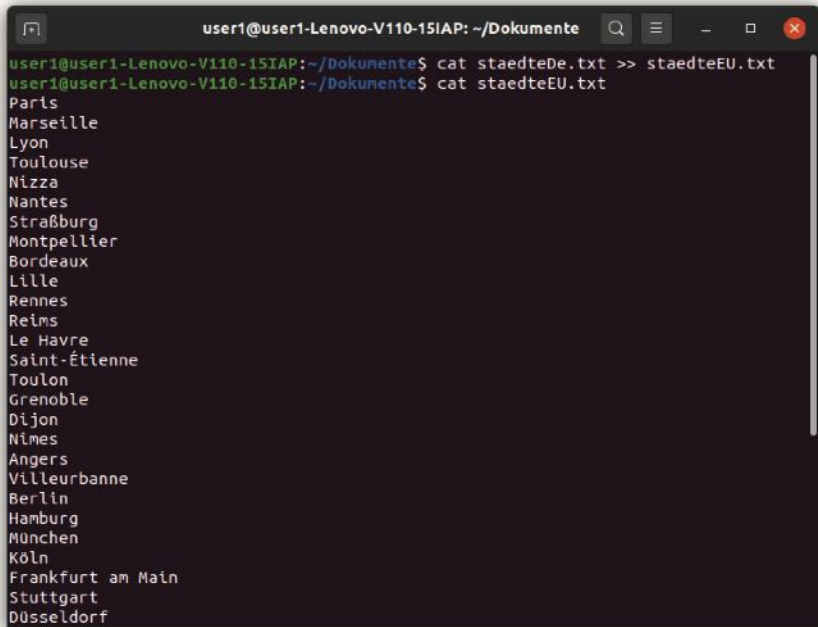
Um mit der Auflistung der Städte in der EU fortzufahren, erstellen wir nun zunächst manuell eine Datei mit den 20 größten Städten in Frankreich. Dieser geben wir den Namen `staedteFr.txt`. Nun fügen wir diese nach dem gleichen Muster in die Datei `staedteEU.txt` ein: `cat staedteFr.txt > staedteEU.txt`.



```
user1@user1-Lenovo-V110-15IAP: ~/Dokumente
user1@user1-Lenovo-V110-15IAP:~/Dokumente$ cat staedteFr.txt > staedteEU.txt
user1@user1-Lenovo-V110-15IAP:~/Dokumente$ cat staedteEU.txt
Paris
Marseille
Lyon
Toulouse
Nizza
Nantes
Straßburg
Montpellier
Bordeaux
Lille
Rennes
Reims
Le Havre
Saint-Étienne
Toulon
Grenoble
Dijon
Nîmes
Angers
Villeurbanne
user1@user1-Lenovo-V110-15IAP:~/Dokumente$
```

Abb. 7.42 Die Datei enthält jetzt nur noch die französischen Städte.

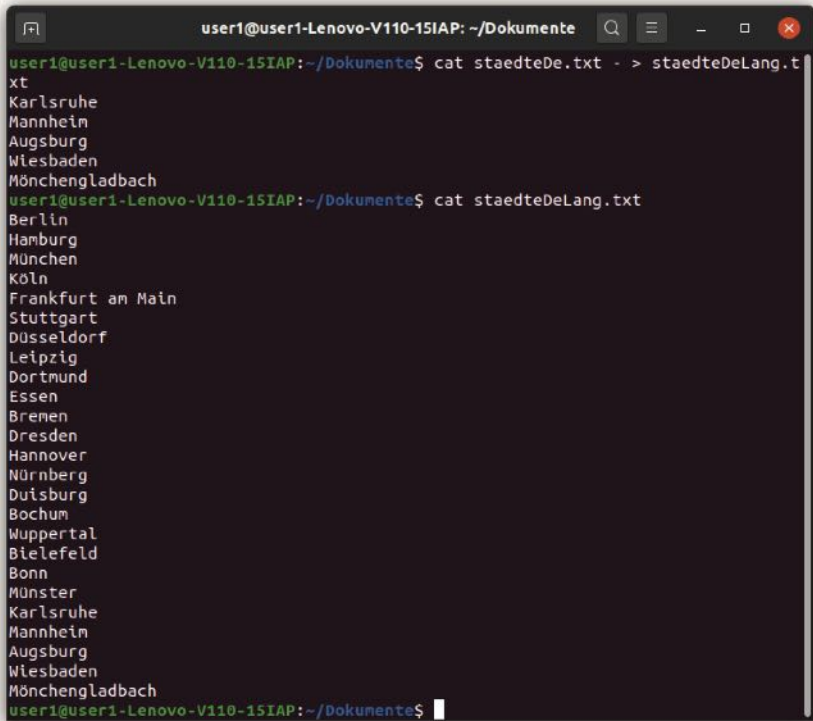
In Abbildung 7.42 ist zu sehen, dass die Datei nun jedoch nur noch die französischen Städte enthält. Die deutschen Städte sind hingegen verschwunden. Das lässt sich jedoch leicht ändern. Wenn wir statt des einfachen Größerzeichens ein doppeltes Größerzeichen verwenden, fügt der `cat`-Befehl die neuen Inhalte an den bestehenden Text an. Auf diese Weise können wir nun wieder die deutschen Städte in die Datei einfügen: `cat staedteDe.txt >> staedteEU.txt`



```
user1@user1-Lenovo-V110-15IAP: ~/Dokumente
user1@user1-Lenovo-V110-15IAP:~/Dokumente$ cat staedteDe.txt >> staedteEU.txt
user1@user1-Lenovo-V110-15IAP:~/Dokumente$ cat staedteEU.txt
Paris
Marseille
Lyon
Toulouse
Nizza
Nantes
Straßburg
Montpellier
Bordeaux
Lille
Rennes
Reims
Le Havre
Saint-Étienne
Toulon
Grenoble
Dijon
Nîmes
Angers
Villeurbanne
Berlin
Hamburg
München
Köln
Frankfurt am Main
Stuttgart
Düsseldorf
```

Abb. 7.43 Die Inhalte zu einer bestehenden Datei hinzufügen

Eine weitere Möglichkeit besteht darin, weitere Inhalte direkt über die Konsole einzugeben. Zu diesem Zweck erstellen wir eine weitere Datei, die ebenfalls die größten Städte in Deutschland enthalten soll. Wir wollen die Liste hierbei jedoch auf 25 Einträge erweitern. Dazu verwenden wir die bisherige Datei und fügen fünf weitere Städte über die Konsole ein. Zu diesem Zweck ist es lediglich notwendig, nach der Angabe der Ausgangsdatei ein Minuszeichen hinzuzufügen. Dieses muss von Leerzeichen umgeben sein: `cat staedteDe.txt - > staedteDeLang.txt`.



```

user1@user1-Lenovo-V110-15IAP: ~/Dokumente
user1@user1-Lenovo-V110-15IAP:~/Dokumente$ cat staedteDe.txt - > staedteDeLang.txt
xt
Karlsruhe
Mannheim
Augsburg
Wiesbaden
Mönchengladbach
user1@user1-Lenovo-V110-15IAP:~/Dokumente$ cat staedteDeLang.txt
Berlin
Hamburg
München
Köln
Frankfurt am Main
Stuttgart
Düsseldorf
Leipzig
Dortmund
Essen
Bremen
Dresden
Hannover
Nürnberg
Duisburg
Bochum
Wuppertal
Bielefeld
Bonn
Münster
Karlsruhe
Mannheim
Augsburg
Wiesbaden
Mönchengladbach
user1@user1-Lenovo-V110-15IAP:~/Dokumente$

```

Abb. 7.44 Die erweiterte Liste mit den Städten

Den entsprechenden `cat`-Befehl müssen wir zunächst mit der Eingabetaste bestätigen. Danach können wir die gewünschten Inhalte hinzufügen. Dabei stellt es auch kein Problem dar, Zeilenumbrüche per Eingabetaste hinzuzufügen. Dadurch wird die Eingabe nicht beendet. Wenn die Liste fertiggestellt ist, müssen wir nochmals die Eingabetaste betätigen und daraufhin die Tastenkombination `Strg + d` drücken. Daraufhin wird der Befehl abgeschlossen und die neue Datei erstellt. Abbildung 7.44 zeigt, dass in diesem Fall die neuen Inhalte am Ende hinzugefügt werden. Wenn Sie sie am Anfang einfügen möchten, müssen Sie das Minuszeichen `-` eben-

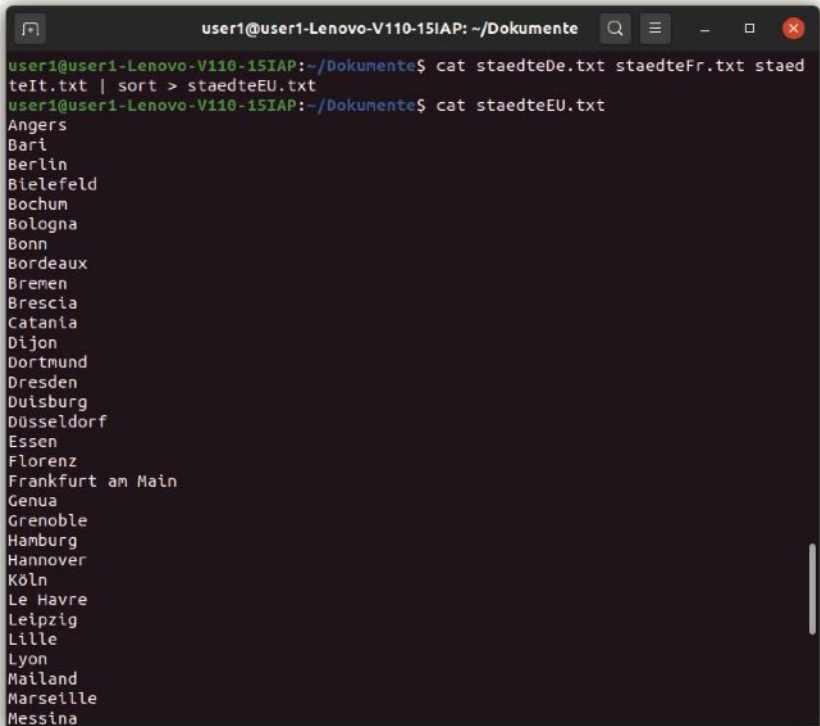
falls von Leerzeichen umgeben – vor dem Dateinamen einfügen:

```
cat - staedteDe.txt > staedteDeLang.txt.
```

Mehrere Dateien zusammenfügen

Es ist nicht nur möglich, die Inhalte verschiedener Dateien einzeln zu einer neuen Datei hinzuzufügen. Diese Aufgabe lässt sich auch in einem einzigen Arbeitsschritt erledigen. Dazu ist es lediglich notwendig, die Ausgangsdateien durch ein Leerzeichen voneinander getrennt anzugeben. Um dies zu demonstrieren, erstellen wir nun noch eine weitere Datei mit den größten Städten Italiens. Die drei Ausgangsdateien fügen wir dann zur Datei `staedteEU.txt` zusammen. Da wir hierbei ein einfaches Größerzeichen verwenden, überschreiben wir dabei die bisherigen Inhalte: `cat staedteDe.txt staedteFr.txt staedteIt.txt > staedteEU.txt.`

Hierbei ist es auch häufig sinnvoll, eine Pipe zu verwenden. Auf diese Weise ist es beispielsweise möglich, die Städte gleichzeitig alphabetisch zu ordnen. Dazu müssen wir hinter den Ausgangsdateien einfach nach einem senkrechten Strich den `sort`-Befehl hinzufügen: `cat staedteDe.txt staedteFr.txt staedteIt.txt | sort > staedteEU.txt.` In Abbildung 7.45 ist zu sehen, dass die neue Datei nun die größten Städte Deutschlands, Frankreichs und Italiens alphabetisch geordnet enthält.



```
user1@user1-Lenovo-V110-15IAP: ~/Dokumente
user1@user1-Lenovo-V110-15IAP$ cat staedteDe.txt staedteFr.txt staed
teit.txt | sort > staedteEU.txt
user1@user1-Lenovo-V110-15IAP$ cat staedteEU.txt
Angers
Bari
Berlin
Bielefeld
Bochum
Bologna
Bonn
Bordeaux
Bremen
Brescia
Catania
Dijon
Dortmund
Dresden
Duisburg
Düsseldorf
Essen
Florenz
Frankfurt am Main
Genua
Grenoble
Hamburg
Hannover
Köln
Le Havre
Leipzig
Lille
Lyon
Malland
Marseille
Messina
```

Abb 7.45 Die Städte in alphabetischer Reihenfolge

Teile einer Datei mit den Befehlen head und tail ausgeben

Wenn wir mit dem `cat`-Befehl eine Datei ausgeben, werden stets alle Inhalte angezeigt. Bereits in unseren bisherigen Beispielen waren die Einträge so umfangreich, dass es nicht möglich war, sie komplett in einem Terminal-Fenster darzustellen. Viele Dateien sind jedoch noch wesentlich umfangreicher. Das beeinträchtigt die Übersichtlichkeit.

Um dieses Problem zu lösen, haben wir bereits den `more`-Befehl kennengelernt, der die Inhalte Stück für Stück präsentiert. Oftmals ist es jedoch sinnvoll, lediglich einen Teilbereich darzustellen. Hierfür kön-

nen wir den `head`- und den `tail`-Befehl verwenden. Diese sind sehr ähnlich in ihrer Anwendung. Der einzige Unterschied besteht darin, dass der `head`-Befehl die Inhalte darstellt, die sich am Anfang der Datei befinden, während der `tail`-Befehl am Ende beginnt.

Das probieren wir nun aus, indem wir unsere Liste mit den EU-Städten mit dem `head`-Befehl ausgeben: `head staedteEU.txt`. In Abbildung 7.46 ist zu sehen, dass nun die ersten zehn Zeilen der Datei erscheinen.

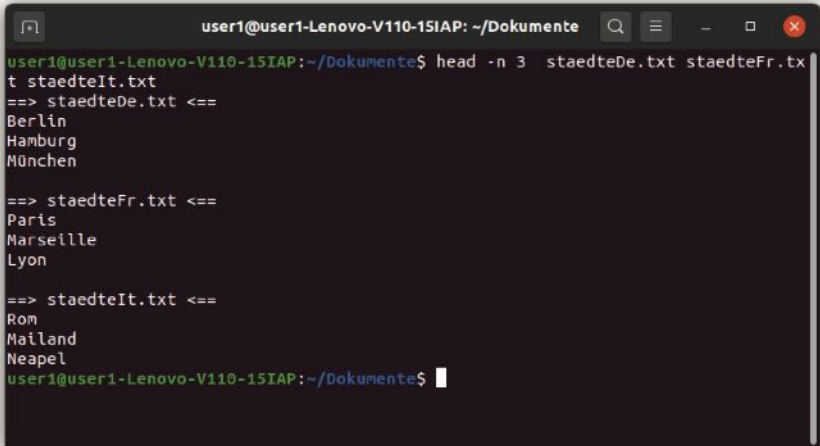
A screenshot of a terminal window with a dark background. The window title is 'user1@user1-Lenovo-V110-15IAP: ~/Dokumente'. The prompt is 'user1@user1-Lenovo-V110-15IAP:~/Dokumente\$'. The command 'head staedteEU.txt' has been entered, and the output shows the first ten lines of the file: 'Angers', 'Bart', 'Berlin', 'Bielefeld', 'Bochum', 'Bologna', 'Bonn', 'Bordeaux', 'Bremen', and 'Brescia'. The prompt is now 'user1@user1-Lenovo-V110-15IAP:~/Dokumente\$' with a cursor.

Abb. 7.46 Die Ausgabe mit dem `head`-Befehl

Oftmals ist es jedoch sinnvoll, nur weniger Einträge anzuzeigen. Dazu können wir die Flag `-n` verwenden. Danach folgt eine Zahl. Damit geben wir vor, wie viele Einträge angezeigt werden sollen. Wenn wir beispielsweise nur drei Zeilen darstellen möchten, erledigen wir dies mit dem folgenden Befehl: `head -n 3 staedteEU.txt`.

Diese Option ist insbesondere dann interessant, wenn wir die Anfänge mehrerer Dateien ausgeben möchten. Auf diese Weise bleibt die Darstellung übersichtlich. Zu diesem Zweck ist es lediglich notwendig, die entsprechenden Dateien durch ein Leerzeichen voneinander getrennt einzufügen: `head -n 3 staedteDe.txt staedteFr.txt`

staedteIt.txt. In Abbildung 7.47 ist zu sehen, dass wir auf diese Weise die drei größten Städte der jeweiligen Länder ausgeben und dabei eine ansprechende und übersichtliche Darstellung erzielen.



```
user1@user1-Lenovo-V110-15IAP: ~/Dokumente
user1@user1-Lenovo-V110-15IAP:~/Dokumente$ head -n 3 staedteDe.txt staedteFr.txt staedteIt.txt
t staedteIt.txt
==> staedteDe.txt <==
Berlin
Hamburg
München

==> staedteFr.txt <==
Paris
Marseille
Lyon

==> staedteIt.txt <==
Rom
Mailand
Neapel
user1@user1-Lenovo-V110-15IAP:~/Dokumente$
```

Abb. 7.47 Die Auflistung der drei größten Städte der entsprechenden Länder

Der `tail`-Befehl funktioniert genau auf die gleiche Weise – lediglich mit dem Unterschied, dass er die letzten Zeilen der Datei darstellt. Hierfür sind jedoch keine weiteren Erklärungen notwendig. Sie können die Funktionsweise gerne selbst ausprobieren.

7.9 Nano Editor, VIM und weitere VI-Editoren über die Shell aufrufen

In Kapitel 7.8 haben wir bereits gelernt, Text-Dateien mit dem `cat`-Befehl anzuzeigen, zu kopieren und zu ergänzen. Die Möglichkeiten für die Bearbeitung der Inhalte sind dabei jedoch stark eingeschränkt. Häufig ist es allerdings notwendig, ganz individuelle Änderungen an den Dateien vorzunehmen. Um dies an einem Beispiel zu verdeutlichen, gehen wir davon aus, dass wir in unserer Datei mit den Städten nun auch eine Nummerierung einfügen möchten, um darzustellen,

welche Stadt an welcher Position der Liste steht. Mit dem `cat`-Befehl wäre es lediglich möglich, die bisherigen Inhalte zu löschen und die überarbeiteten Werte komplett neu einzugeben. Das wäre selbstverständlich mit einem erheblichen zusätzlichen Aufwand verbunden. Eine weitere Möglichkeit stellt es dar, die Datei mit einem Texteditor wie `gedit` zu öffnen. Mit diesem Werkzeug lassen sich die Änderungen zwar problemlos durchführen. Allerdings bringt es einen gewissen Umbruch und eine Wartezeit mit sich, ein Programm in einem neuen Fenster zu öffnen und zu diesem zu wechseln. Insbesondere wenn wir viele kleinere Korrekturen durchführen müssen und zwischendurch wieder das Terminal verwenden, schränkt das die Produktivität ein.

Aus diesem Grund sind für die Arbeit mit der Shell Editoren beliebt, die direkt im Terminal ablaufen. Diese werden im gleichen Fenster geöffnet. Dieser Vorgang nimmt nur Sekundenbruchteile in Anspruch. Außerdem müssen Sie den Arbeitsbereich dabei nicht wechseln. Das ermöglicht eine hohe Produktivität. Diese Editoren sollen in den folgenden Abschnitten vorgestellt werden.

Der Nano-Editor

Es gibt zahlreiche verschiedene Texteditoren, die sich direkt in der Konsole ausführen lassen. Einer der beliebtesten von ihnen trägt die Bezeichnung Nano Editor. Diese Software müssen wir zunächst installieren. Da wir in Kapitel 7.7 bereits gelernt haben, wie wir diese Aufgabe erledigen, sollte dies kein Problem mehr darstellen: `sudo apt install nano`.

Nachdem die Installation abgeschlossen ist, können wir die Datei mit unseren Städten im Nano-Editor öffnen. Dazu benötigen wir den Befehl `nano` – gefolgt vom entsprechenden Dateinamen: `nano staedteDe.txt`.

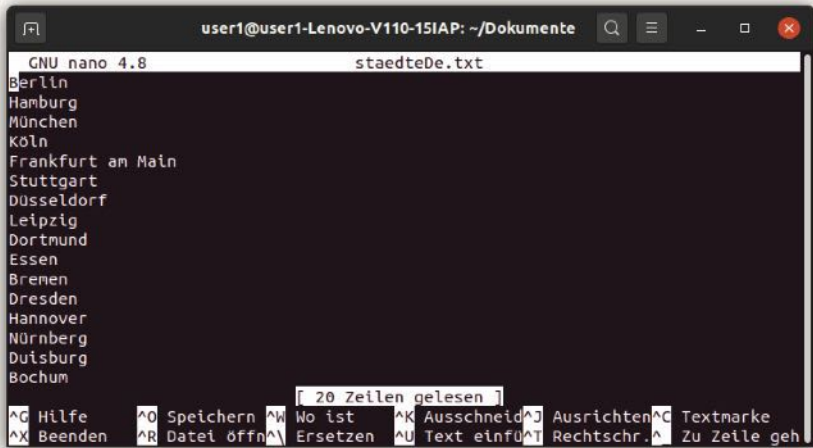


Abb. 7.48 Die Darstellung der Datei im Nano-Editor

Hier werden nun die bestehenden Inhalte angezeigt. Diese können wir genau auf die gleiche Weise verändern wie bei einem gewöhnlichen Texteditor. Der einzige Unterschied besteht darin, dass die Steuerungsmöglichkeiten mit der Maus eingeschränkt sind. Beispielsweise ist es nicht möglich, den Cursor per Mausklick in den gewünschten Bereich zu setzen. Daher ist es notwendig, hierfür die Pfeiltasten zu verwenden. Auf diese Weise können wir die gewünschten Nummerierungen jedoch schnell und einfach einfügen.

Im unteren Bereich sind nun verschiedene Aktionen angegeben, die wir mit der Datei durchführen können. Es ist beispielsweise möglich, sie zu speichern oder eine neue Datei zu öffnen. Außerdem erlaubt es der Texteditor, Textpassagen zu kopieren, auszuschneiden oder einzufügen. Es ist sogar möglich, bestimmte Zeichenfolgen zu suchen und durch einen anderen Ausdruck zu ersetzen. Sicherlich sind die Funktionen hierbei nicht so umfangreich wie bei gewöhnlichen Texteditoren. Doch reichen Sie für die meisten Aufgaben vollkommen aus. Die dafür erforderlichen Tastenkombinationen können Sie einfach im unteren Bereich des Fensters ablesen. Daher ist die Verwendung weitgehend

selbsterklärend. Weitere Informationen stehen außerdem über die Hilfe-Funktion (Strg + G) zur Verfügung.

In dem Bereich, in dem die Befehle aufgelistet sind, erscheint immer das Zirkumflex-Zeichen (^) und anschließend ein Buchstabe. Das Zirkumflexzeichen steht dabei für die Strg-Taste. Das bedeutet, dass Sie diese gleichzeitig mit dem anschließend genannten Zeichen betätigen müssen. Um die überarbeitete Datei abzuspeichern, müssen Sie beispielsweise Strg + O drücken. Abbildung 7.49 zeigt, dass Sie nun die Änderungen in der bisherigen Datei abspeichern oder dafür ein neues Dokument erstellen können.

7

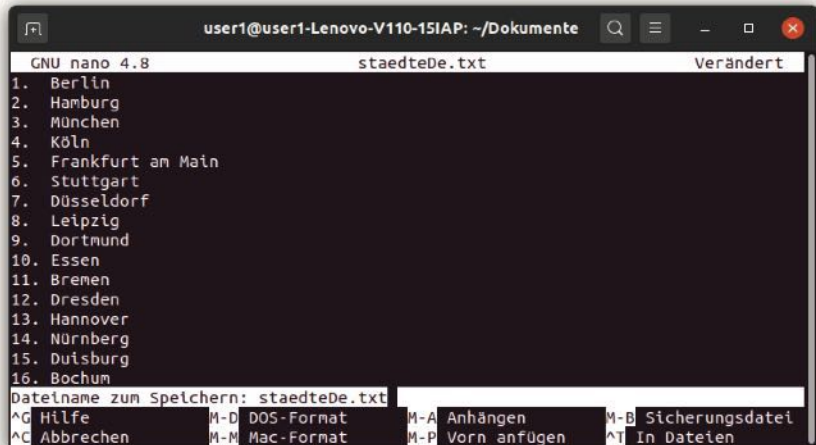


Abb. 7.49 Das Speichern einer Datei im Nano-Editor

Es ist auch möglich, den Nano-Editor mit verschiedenen Flags zu starten. Sehr beliebt ist beispielsweise die Flag `-c`: `nano -c staedteDe.txt`. Diese führt dazu, dass im Fenster stets die aktuelle Position des Cursors angegeben ist – für die Zeile, für die Spalte und für das Zeichen. Dabei ist außerdem stets die Gesamtzahl der Zeilen, Spalten und Zeichen angegeben sowie die relative Position. Diese Informationen sind insbesondere bei sehr umfangreichen Dateien hilfreich, um den Überblick zu bewahren.



Abb. 7.50 Die Angabe der Position des Cursors

Der VIM-Editor

Eine weitere Möglichkeit stellt der Editor VIM dar. Dieser bietet etwas umfangreichere Funktionen als der Nano-Editor. Allerdings ist die Anwendung deutlich weniger intuitiv. Daher ist für die Verwendung eine etwas umfangreichere Einarbeitung erforderlich – oder ein häufiges Konsultieren der Hilfe.

Wenn wir auch diesen Editor ausprobieren möchten, ist es notwendig, ihn zunächst zu installieren: `sudo apt install vim`. Danach können wir mit dem Befehl `vim` und der Angabe der gewünschten Textdatei den Editor öffnen: `vim staedteDe.txt`. Abbildung 7.51 zeigt die Oberfläche, die wir daraufhin erblicken.

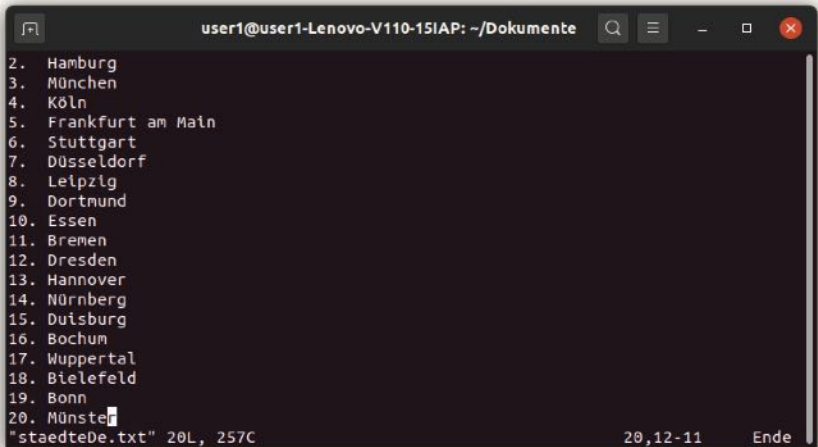


Abb. 7.51 Eine Datei mit dem VIM-Editor bearbeiten

Nun können Sie zwar den Cursor im Text bewegen. Wenn Sie versuchen, die Inhalte zu verändern, passiert aber zunächst einmal nichts. Im Gegensatz zum Nano-Editor sind auch keine Anweisungen enthalten, wie Sie vorgehen müssen. Das gestaltet den Einstieg etwas schwieriger. Mit einer kurzen Einarbeitung ist es aber dennoch schnell möglich, den Umgang mit diesem Werkzeug zu erlernen.

Um den Text zu bearbeiten, müssen Sie zunächst entweder die Einfg-Taste oder die i-Taste betätigen. Damit wechseln Sie den Modus des Editors und können die erforderlichen Anpassungen vornehmen. Das erkennen Sie daran, dass ganz unten im Fenster der Ausdruck -- EIN-FÜGEN -- erscheint.

Nachdem Sie die Anpassungen beendet haben, müssen Sie den Editor wieder in den ursprünglichen Modus zurückversetzen. Zu diesem Zweck dient die Esc-Taste. Nun haben Sie beispielsweise die Möglichkeit, die Änderungen abzuspeichern. Dazu müssen Sie den Ausdruck :w eingeben und mit der Eingabetaste bestätigen. Dabei ist es wichtig, darauf zu achten, dass dies nur funktioniert, wenn Sie sich im normalen Modus befinden. Sollte es zu Problemen kommen, kann es sinnvoll sein, nochmals die Esc-Taste zu betätigen, um dies sicherzustellen.

Um das Programm zu verlassen, müssen Sie – ebenfalls im normalen Modus – den Befehl :q eingeben. Dieser wird jedoch nur dann ausgeführt, wenn Sie die Änderungen zuvor gespeichert haben. Ist dies nicht der Fall, verhindert der Editor ein Beenden des Programms. In diesem Fall müssen Sie die Datei zunächst abspeichern. Wenn Sie das Programm verlassen möchten, ohne die Änderungen abzuspeichern, müssen Sie hingegen den Befehl :q! verwenden. Durch das Hinzufügen des Ausrufungszeichens ist es möglich, das Programm auch ohne vorheriges Speichern zu beenden.

Hierbei handelt es sich nur um die grundlegenden Befehle des VIM-Editors. Es gibt noch viele weitere Steuerungsmöglichkeiten. Diese hier alle aufzuführen, würde jedoch den Rahmen dieses Buchs sprengen. Wenn Ihnen dieser Editor zusagt, können Sie sich aber auch problemlos selbst in die Nutzung einarbeiten. Indem Sie den Begriff :h eingeben und

per Eingabetaste bestätigen, gelangen Sie zur Hilfe des Programms. Hier sind alle Befehle aufgeführt.

Beim VIM-Editor handelt es sich um eine Weiterentwicklung des ursprünglichen VI-Editors, der bereits 1976 erschien und der zu dieser Zeit sehr populär war. Dieser stellt zwar den am häufigsten verwendeten Abkömmling des VI-Editors dar, aber bei Weitem nicht den einzigen. Es gibt noch zahlreiche weitere Angebote, die teilweise starke Ähnlichkeiten zum VIM-Editor aufweisen – beispielsweise Elvis, vile, Extensible Versatile Editor oder Aquamacs Emacs. Bei Interesse können Sie diese selbst installieren und die Anwendung ausprobieren.

7.10 Informationen mit `help`, `man` und `info` abrufen

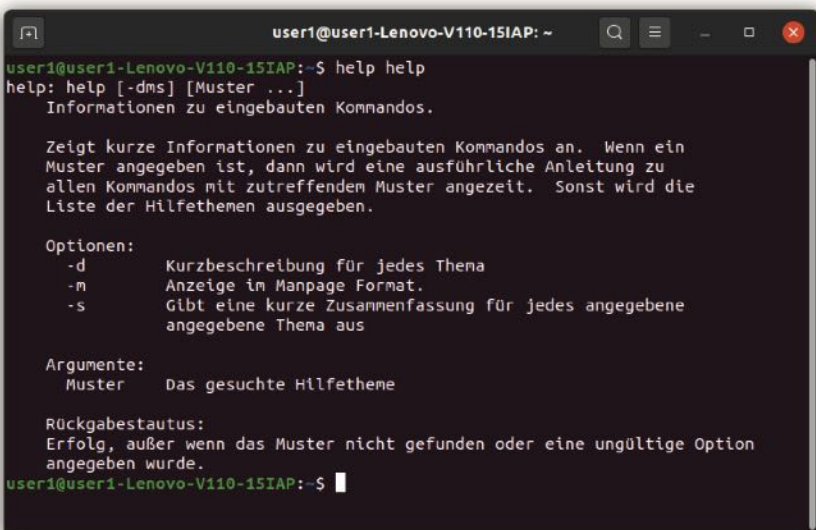
In den letzten Abschnitten haben wir bereits zahlreiche Shell-Befehle kennengelernt. Dennoch handelt es sich hierbei nur um einen kleinen Ausschnitt aus den vielfältigen Möglichkeiten für die Steuerung des Systems. Es kommt hinzu, dass auch die hier vorgestellten Kommandos noch zahlreiche weitere Optionen ermöglichen, die bislang nicht beschrieben wurden.

Da eine komplette Erklärung aller Möglichkeiten zu umfangreich wäre, um sie in diesem Buch darzustellen, stellt sich die Frage, auf welche Weise Sie Informationen zu den übrigen Shell-Befehlen erhalten. Das ist nicht nur notwendig, wenn Sie sich in weitere Funktionen einarbeiten möchten. Auch wenn Sie die Funktionsweise eines Kommandos vergessen haben sollten, ist es sehr wichtig, dass eine Möglichkeit bereitsteht, um die benötigten Informationen erneut abzurufen.

Eine sinnvolle Möglichkeit besteht sicherlich darin, eine Internetrecherche durchzuführen. Mit einer Suchmaschine ist es in der Regel schnell möglich, umfangreiche Informationen und passende Anwendungsbeispiele zu den verschiedenen Kommandos zu finden. Doch bietet auch die Shell selbst viele hilfreiche Informationen an. Hierbei sind insbesondere die Befehle `help`, `man` und `info` von Bedeutung.

Der Befehl help für alle Builtin-Kommandos

Eine Möglichkeit, um eine Hilfestellung bei der Nutzung der Shell zu erhalten, besteht in der Verwendung des Befehls `help`. Danach müssen wir das Kommando einfügen, über das wir weitere Informationen erhalten möchten. Um die Verwendung vorzustellen, rufen wir zunächst einmal die Hilfe zum `help`-Befehl selbst ab. Dafür verwenden wir das Kommando `help help`.



```
user1@user1-Lenovo-V110-15IAP: ~
user1@user1-Lenovo-V110-15IAP:~$ help help
help: help [-dms] [Muster ...]
    Informationen zu eingebauten Kommandos.

    Zeigt kurze Informationen zu eingebauten Kommandos an. Wenn ein
    Muster angegeben ist, dann wird eine ausführliche Anleitung zu
    allen Kommandos mit zutreffendem Muster angezeigt. Sonst wird die
    Liste der Hilfethemen ausgegeben.

    Optionen:
      -d      Kurzbeschreibung für jedes Thema
      -m      Anzeige im Manpage Format.
      -s      Gibt eine kurze Zusammenfassung für jedes angegebene
              angegebene Thema aus

    Argumente:
      Muster  Das gesuchte Hilfethema

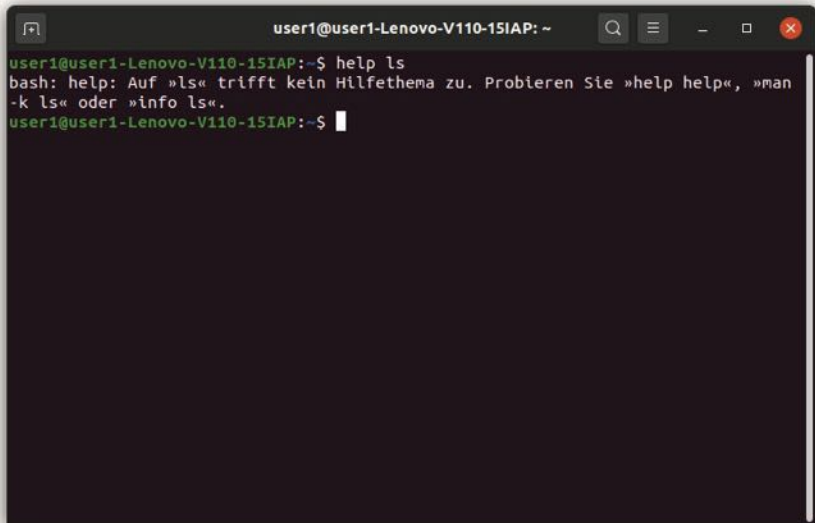
    Rückgabestatus:
    Erfolg, außer wenn das Muster nicht gefunden oder eine ungültige Option
    angegeben wurde.
user1@user1-Lenovo-V110-15IAP:~$
```

Abb. 7.52 Die Hilfe zum Befehl `help`

In der ersten Zeile erscheint eine Vorlage für die Anwendung dieses Kommandos. Darauf gehen wir jedoch erst später ein. Zunächst befassen wir uns mit der Beschreibung des `help`-Befehls. Hier wird angegeben, dass wir auf diese Weise Informationen zu eingebauten Kommandos erhalten. Das stellt einen wichtigen Aspekt bei der Verwendung des `help`-Befehls dar – und gleichzeitig eine erhebliche Einschränkung. Die Shell verfügt über einige eingebaute Befehle, die auch als *Builtins* bezeichnet werden. Diese decken jedoch nur die grundlegenden Funktionen ab. Die meisten Befehle sind hingegen im Rahmen von eigenen

Programmen implementiert. Von den Kommandos, die wir bislang in diesem Buch vorgestellt haben, handelte es sich nur bei wenigen Beispielen um eingebaute Kommandos. Daher ist der `help`-Befehl nur bei einer sehr geringen Zahl von Befehlen hilfreich.

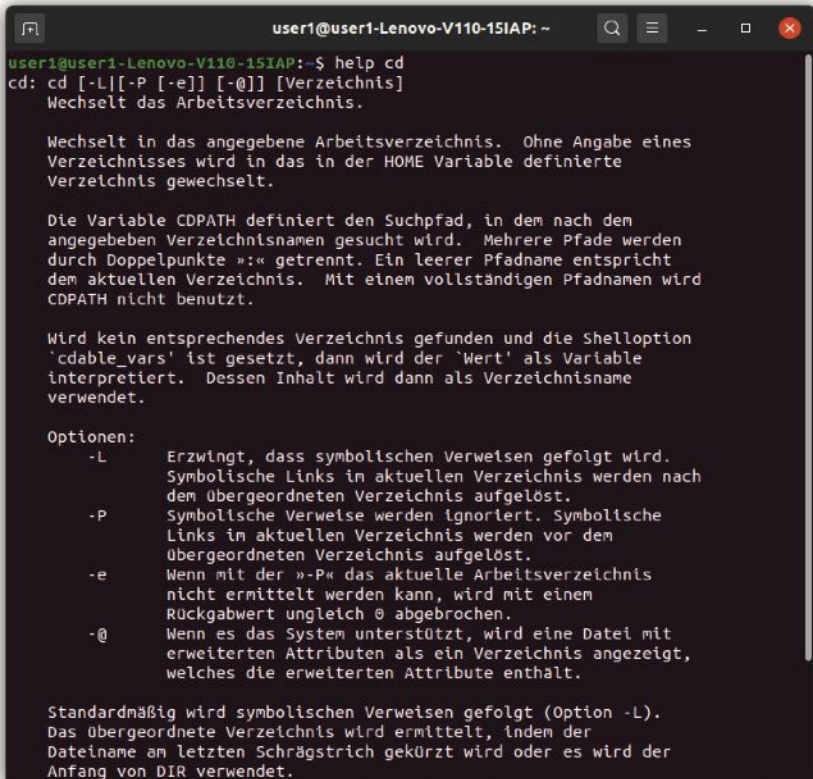
Das sollen die folgenden Beispiele zeigen. Wie wir bereits wissen, müssen wir für die Verwendung der Hilfe den Begriff `help` und anschließend das Kommando eingeben, über das wir uns informieren möchten. Das probieren wir nun mit einem Befehl aus, den wir bereits sehr häufig verwendet haben: `ls`.



```
user1@user1-Lenovo-V110-15IAP: ~  
user1@user1-Lenovo-V110-15IAP:~$ help ls  
bash: help: Auf »ls« trifft kein Hilfethema zu. Probieren Sie »help help«, »man  
-k ls« oder »info ls«.  
user1@user1-Lenovo-V110-15IAP:~$
```

Abb. 7.53 Die Nachricht beim Abruf der Hilfe für einen Befehl, der nicht in die Shell eingebaut ist.

In Abbildung 7.53 ist zu erkennen, dass dieses Kommando nicht in den Hilfsthemen aufgelistet ist. Das liegt daran, dass es sich hierbei nicht um ein Builtin handelt. Im nächsten Schritt probieren wir das Kommando dann noch einmal mit einem weiteren häufig verwendeten Behl aus: mit `cd`.



```

user1@user1-Lenovo-V110-15IAP: ~
user1@user1-Lenovo-V110-15IAP:~$ help cd
cd: cd [-L|[-P [-e]] [-@]] [Verzeichnis]
    Wechselt das Arbeitsverzeichnis.

    Wechselt in das angegebene Arbeitsverzeichnis. Ohne Angabe eines
    Verzeichnisses wird in das in der HOME Variable definierte
    Verzeichnis gewechselt.

    Die Variable CDPATH definiert den Suchpfad, in dem nach dem
    angegebenen Verzeichnisnamen gesucht wird. Mehrere Pfade werden
    durch Doppelpunkte »:« getrennt. Ein leerer Pfadname entspricht
    dem aktuellen Verzeichnis. Mit einem vollständigen Pfadnamen wird
    CDPATH nicht benutzt.

    Wird kein entsprechendes Verzeichnis gefunden und die Shelloption
    'cdable_vars' ist gesetzt, dann wird der 'Wert' als Variable
    interpretiert. Dessen Inhalt wird dann als Verzeichnisname
    verwendet.

    Optionen:
      -L      Erzwingt, dass symbolischen Verweisen gefolgt wird.
              Symbolische Links in aktuellen Verzeichnis werden nach
              dem übergeordneten Verzeichnis aufgelöst.
      -P      Symbolische Verweise werden ignoriert. Symbolische
              Links in aktuellen Verzeichnis werden vor dem
              übergeordneten Verzeichnis aufgelöst.
      -e      Wenn mit der »-P« das aktuelle Arbeitsverzeichnis
              nicht ermittelt werden kann, wird mit einem
              Rückgabewert ungleich 0 abgebrochen.
      -@      Wenn es das System unterstützt, wird eine Datei mit
              erweiterten Attributen als ein Verzeichnis angezeigt,
              welches die erweiterten Attribute enthält.

    Standardmäßig wird symbolischen Verweisen gefolgt (Option -L).
    Das übergeordnete Verzeichnis wird ermittelt, indem der
    Dateiname am letzten Schrägstrich gekürzt wird oder es wird der
    Anfang von DIR verwendet.
  
```

7

Abb. 7.54 Die Hilfeseite für den `cd`-Befehl

In Abbildung 7.54 wird deutlich, dass für den `cd`-Befehl ein Eintrag vorhanden ist. Das liegt daran, dass es sich hierbei um ein eingebautes Kommando handelt. Dabei werden das Muster für die Verwendung, eine kurze Beschreibung und die möglichen Optionen angegeben. Anhand dieser Informationen ist es problemlos möglich, den Befehl korrekt zu verwenden.

Nun stellt sich noch die Frage, wie wir herausfinden können, bei welchen Kommandos es sich um Builtins handelt und bei welchen nicht. Hierzu betrachten wir nochmals Abbildung 7.53 – die Hilfeseite zum

help-Befehl. In der ersten Zeile ist hier die Struktur für diesen Befehl angegeben: `help [-dms] [Muster ...]`. Zunächst steht hier der Ausdruck `help`, mit dem wir diesen Befehl aufrufen. Danach folgen zwei Ergänzungen. Diese stehen jedoch in einer eckigen Klammer. Das bedeutet, dass sie optional sind. Daraus folgt, dass wir sie auch weglassen können. Bislang haben wir den `help`-Befehl stets mit einem weiteren Kommando kombiniert. Das entspricht dem Ausdruck `Muster` in der zweiten eckigen Klammer. Im Hilfetext ist aber auch zu sehen, dass wenn wir diesen Bereich weglassen „die Liste der Hilfetemen“ angezeigt wird. Das probieren wir nun aus, indem wir nur den Begriff `help` eingeben.

```

user1@user1-Lenovo-V110-15IAP: ~
(( Ausdruck ))
:
[ Argument... ]
[[ Ausdruck ]]
alias [-p] [Name[=Wert] ... ]
bg [Jobbezeichnung ...]
bind [-lpsvPSVX] [-m Tastaturtabelle]>
break [n]
builtin [Shell-Kommando [Argument ...>
caller [Ausdruck]
case Wort in [Muster [| Muster]...> K>
cd [-L][-P [-e]] [-@]] [Verzeichnis]
command [-pVv] Kommando [Argument ...>
compgen [-abcdefgjkuv] [-o Option] [>
complete [-abcdefgjkuv] [-pr] [-DE] >
comptopt [-o+o Option] [-DEI] [Name .>
continue [n]
coproc [Name] Kommando [Unleitungen]
declare [-aAffgIlrtux] [-p] [Name=W>
dirs [-clpv] [+N] [-N]
disown [-h] [-ar] [Jobbezeichnung ...>
echo [-neE] [Argument ...]
enable [-a] [-dnps] [-f Dateiname] [N>
eval [Argument ...]
exec [-cl] [-a Name] [Kommando [Argum>
exit [n]
export [-fn] [Name[=Wert] ...] oder e>
false
fc [-e Editor] [-lnr] [Anfang] [Ende]>
fg [Jobbezeichnung]
for Name [in Wortliste ... ] ; do Kom>
for (( Ausdruck1; Ausdruck2; Ausdruck>
function Name { Kommandos ; } oder Na>
getopts Optionen Variable [Argumente>
hash [-lr] [-p Pfadname] [-dt] [Name >
help [-dms] [Muster ...]
if Kommandos; then Kommandos; [ elif>
jobs [-lnprs] [Jobbezeichnung ...] o>
kill [-s Signalname | -n Signalnumme>
let Argument [Argument ...]
local [Option] Name[=Wert] ...
logout [n]
mapfile [-d Begrenzer] [-n Anzahl] [>
popd [-n] [-N | -N]
printf [-v var] Format [Argumente]
pushd [-n] [+N | -N | Verzeichnis]
pwd [-LP]
read [-ers] [-a Feld] [-d Begrenzer]>
readarray [-d Begrenzer] [-n Anzahl]>
readonly [-aAf] [Name[=Wert] ...] od>
return [n]
select Name [in Wortliste ... ] do >
set [-abefhknptuvxBCHP] [-o Option]>
shift [n]
shopt [-pqsu] [-o] [Optionsname ...]
source Dateiname [Argumente]
suspend [-f]
test [Ausdruck]
time [-p] Pipeline
times
trap [-lp] [[Argument] Signalbezeich>
true
type [-afptP] Name [Name ...]
typeset [-aAffgIlrtux] [-p] Name[=W>
ulimit [-SHabceflklmnpqrstuvxPT] [G>
umask [-p] [-S] [Modus]
unalias [-a] Name [Name ...]
unset [-f] [-v] [-n] [NAME ...]
until Kommandos; do Kommandos; done
variables - Namen und Bedeutung einl>
wait [-fn] [id ...]
while Kommandos; do Kommandos; done
{ Kommandos ; }

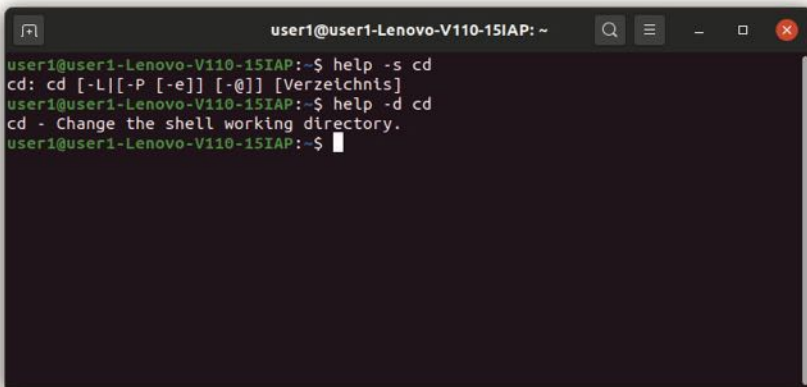
```

Abb. 7.55 Die Liste der Builtin-Kommandos

Auf diese Weise erscheint die Liste, die in Abbildung 7.55 zu sehen ist. Diese enthält alle vorhandenen Builtin-Befehle. Manche von ihnen haben wir bereits kennengelernt. Im weiteren Verlauf dieses Buchs werden wir uns außerdem noch mit einigen weiteren hier aufgeführten Befehle beschäftigen. Dennoch wird deutlich, dass es sich hierbei nur um einen sehr kleinen Ausschnitt der gängigen Shell-Befehle handelt. Deshalb bietet der `help`-Befehl in vielen Fällen keine Hilfe.

7

Dennoch wollen wir uns abschließend noch kurz mit den möglichen Flags befassen. Dabei sind insbesondere die Flags `-s` und `-d` von Bedeutung. Die Flag `-s` führt dazu, dass nur die Struktur des Kommandos mit allen möglichen Optionen aufgeführt wird. Die Flag `-d` hat hingegen zur Folge, dass nur eine kurze Beschreibung angegeben wird, die meistens nur aus einer Zeile besteht. Beide Optionen dienen dazu, sich einen schnellen Überblick über die Funktionsweise zu verschaffen. Beide Möglichkeiten sind in Abbildung 7.56 dargestellt.



```

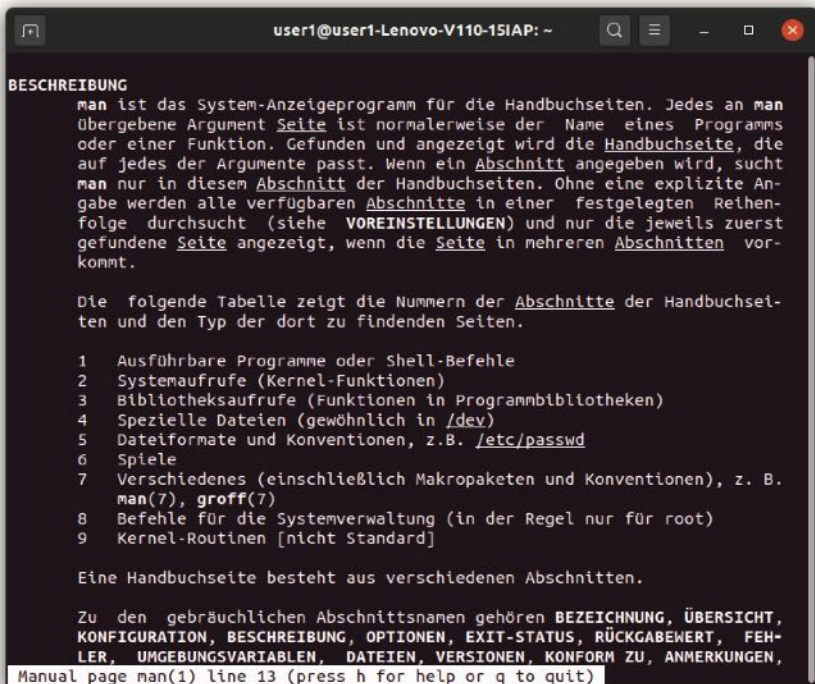
user1@user1-Lenovo-V110-15IAP: ~
user1@user1-Lenovo-V110-15IAP:~$ help -s cd
cd: cd [-L|[-P [-e]] [-@]] [Verzeichnis]
user1@user1-Lenovo-V110-15IAP:~$ help -d cd
cd - Change the shell working directory.
user1@user1-Lenovo-V110-15IAP:~$

```

Abb. 7.56 Der `help`-Befehl mit den Flags `-s` und `-d`

Der Befehl man für weitere Kommandos

Wir haben festgestellt, dass der Befehl `help` nur bei sehr wenigen Kommandos hilfreich ist. In den übrigen Fällen kann uns der Begriff `man` wertvolle Informationen geben. Dieser steht für Manpages. Er führt uns zu einer umfangreichen Dokumentation für viele Bereiche des Linux-Betriebssystems. Für den Einstieg rufen wir den Handbucheintrag zum `man`-Befehl auf: `man man`. Dabei entdecken wir zunächst Angaben zur Struktur dieses Kommandos. Wenn wir etwas nach unten scrollen, gelangen wir zum Bereich mit der Überschrift „Beschreibung“, der in Abbildung 7.57 zu sehen ist.



```

user1@user1-Lenovo-V110-15IAP: ~
BESCHREIBUNG
man ist das System-Anzeige-Programm für die Handbuchseiten. Jedes an man
übergebene Argument Seite ist normalerweise der Name eines Programms
oder einer Funktion. Gefunden und angezeigt wird die Handbuchseite, die
auf jedes der Argumente passt. Wenn ein Abschnitt angegeben wird, sucht
man nur in diesem Abschnitt der Handbuchseiten. Ohne eine explizite An-
gabe werden alle verfügbaren Abschnitte in einer festgelegten Reihen-
folge durchsucht (siehe VOREINSTELLUNGEN) und nur die jeweils zuerst
gefundene Seite angezeigt, wenn die Seite in mehreren Abschnitten vor-
kommt.

Die folgende Tabelle zeigt die Nummern der Abschnitte der Handbuchset-
ten und den Typ der dort zu findenden Seiten.

1  Ausführbare Programme oder Shell-Befehle
2  Systemaufrufe (Kernel-Funktionen)
3  Bibliotheksaufrufe (Funktionen in Programmbibliotheken)
4  Spezielle Dateien (gewöhnlich in /dev)
5  Dateiformate und Konventionen, z.B. /etc/passwd
6  Spiele
7  Verschiedenes (einschließlich Makropaketen und Konventionen), z. B.
   man(7), groff(7)
8  Befehle für die Systemverwaltung (in der Regel nur für root)
9  Kernel-Routinen [nicht Standard]

Eine Handbuchseite besteht aus verschiedenen Abschnitten.

Zu den gebräuchlichen Abschnittsnamen gehören BEZEICHNUNG, ÜBERSICHT,
KONFIGURATION, BESCHREIBUNG, OPTIONEN, EXIT-STATUS, RÜCKGABEWERT, FEH-
LER, UMGEBUNGSVARIABLEN, DATEIEN, VERSIONEN, KONFORM ZU, ANMERKUNGEN,
Manual page man(1) line 13 (press h for help or q to quit)
  
```

Abb. 7.57 Die Beschreibung zum Ausdruck `man`

Hierbei ist insbesondere die Liste mit den Themen von Interesse, zu denen Handbuch-Einträge vorhanden sind. An erster Stelle stehen hierbei ausführbare Programme und Shell-Befehle. Danach sind jedoch noch viele weitere Bereiche aufgelistet – von Systemaufrufen bis hin zu Kernel-Routinen. Daran wird deutlich, dass wir mit dem Befehl `man` Informationen zu vielfältigen Funktionen des Betriebssystems abrufen können.

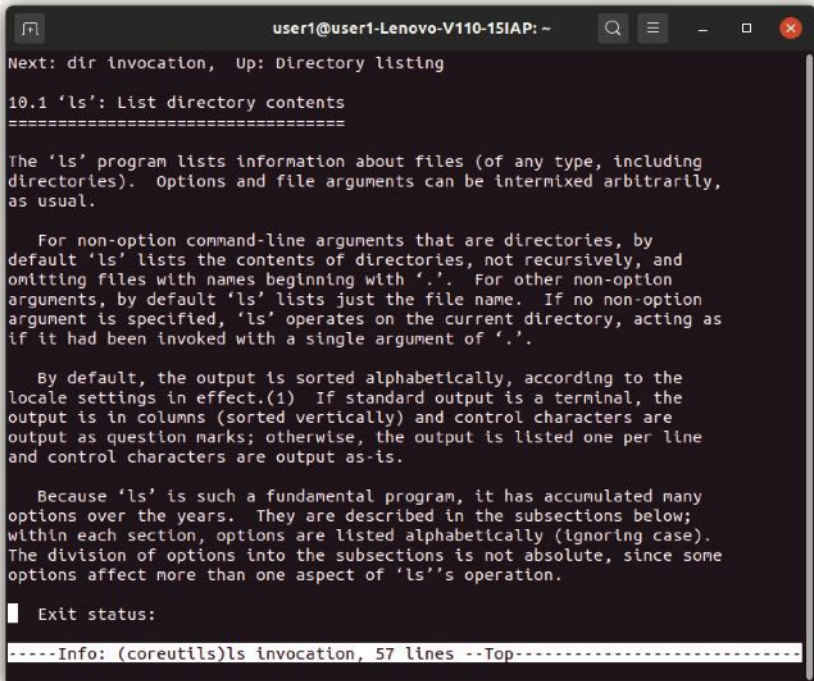
Die Funktionsweise soll nun an einem weiteren Beispiel verdeutlicht werden. Dazu verlassen wir zunächst diesen Eintrag, indem wir den Befehl `q` eingeben. Auf diese Weise kehren wir wieder zur Shell zurück. Nun geben wir den Begriff `man ls` ein. Dadurch gelangen wir zu einem recht umfangreichen Eintrag. Wir erkennen nicht nur die Struktur und die Funktionsweise des Kommandos. Darüber hinaus erhalten wir Zugang zu einer langen Liste mit möglichen Flags. Daran wird deutlich, wie vielfältig die Optionen sind, die wir bei diesem Befehl vorgeben können. Auf diese Weise können Sie sich bestens mit den möglichen Funktionsweisen vertraut machen.

7

Der Befehl `info` für umfassendere Informationen

Eine weitere Möglichkeit stellt der Befehl `info` dar. Auch dieser bietet weiterführende Informationen zu zahlreichen Shell-Befehlen und weiteren Bestandteilen des Betriebssystems. Die Verwendung ist dabei sehr ähnlich wie bei den beiden bereits vorgestellten Möglichkeiten: Sie müssen zunächst den Befehl `info` eingeben und danach das Kommando, zu dem Sie die Informationen abrufen möchten.

Das probieren wir nun für den Befehl `ls` aus, über den wir uns gerade in den Manpages informiert haben. Auf diese Weise ist es möglich, die Einträge zu vergleichen. Daher geben wir nun das Kommando `info ls` in die Shell ein.



```

user1@user1-Lenovo-V110-15IAP: ~
Next: dir invocation, Up: Directory listing

10.1 'ls': List directory contents
=====

The 'ls' program lists information about files (of any type, including
directories). Options and file arguments can be intermixed arbitrarily,
as usual.

    For non-option command-line arguments that are directories, by
    default 'ls' lists the contents of directories, not recursively, and
    omitting files with names beginning with '.'. For other non-option
    arguments, by default 'ls' lists just the file name. If no non-option
    argument is specified, 'ls' operates on the current directory, acting as
    if it had been invoked with a single argument of '.'.

    By default, the output is sorted alphabetically, according to the
    locale settings in effect.(1) If standard output is a terminal, the
    output is in columns (sorted vertically) and control characters are
    output as question marks; otherwise, the output is listed one per line
    and control characters are output as-is.

    Because 'ls' is such a fundamental program, it has accumulated many
    options over the years. They are described in the subsections below;
    within each section, options are listed alphabetically (ignoring case).
    The division of options into the subsections is not absolute, since some
    options affect more than one aspect of 'ls's operation.

Exit status:

-----Info: (coreutils)ls invocation, 57 lines --Top-----

```

Abb. 7.58 Weiterführende Informationen zum `ls`-Befehl.

Wenn Sie den Eintrag zum `ls`-Befehl betrachten, stellen Sie zunächst fest, dass dieser auf Englisch verfasst ist. Das stellt einen wichtigen Unterschied zu den Befehlen `help` und `man` dar, bei denen jeweils eine deutsche Übersetzung verfügbar ist. Darüber hinaus ist dieser Eintrag ausgesprochen umfangreich. Sie erhalten ausführliche Informationen zur Funktionsweise, zu den möglichen Optionen und zu Kombinationsmöglichkeiten mit anderen Kommandos – beispielsweise mit einer Pipe.

Auf diese Weise erhalten Sie zwar umfassende Kenntnisse zur Verwendung des Befehls. Allerdings fällt der Überblick dadurch etwas schwerer. Wenn Sie sich lediglich mit den wesentlichen Details vertraut machen möchten, ist daher der Befehl `man` vorzuziehen. Nur wenn Sie Wert auf

eine ausführliche Beschreibung legen, sollten Sie das Kommando `info` verwenden.

Wenn Sie mit dem `info`-Befehl arbeiten, ist es wichtig, darauf zu achten, dass dieser Links enthält. Dabei handelt es sich um die unterstrichenen Bereiche im Text. Diese verweisen auf verschiedene weiterführende Inhalte zum entsprechenden Thema. Diese beziehen sich jedoch alle auf das gleiche Dokument. Es handelt sich also lediglich um interne Links innerhalb dieser Dokumentation.

7

Wenn Sie einen dieser Links anklicken, passiert jedoch nichts. Die Steuerung ist nur per Tastatur möglich. Zu diesem Zweck ist ein Cursor vorhanden. Diesen können Sie mit den Pfeiltasten bewegen. Wenn der Cursor auf einem Link steht und Sie daraufhin die Eingabetaste betätigen, gelangen Sie zu den entsprechenden Inhalten.

7.11 Übungsaufgaben

Suchen Sie selbstständig nach Lösungen, um die folgenden Aufgaben über die Shell zu erledigen:

1. Ändern Sie den Namen des Ordners „Beispielordner“. Erstellen Sie diesen zunächst, falls er noch nicht besteht.
2. Öffnen Sie eine Datei (`bsp.txt`) über die Shell mit dem Texteditor `gedit`.
3. Geben Sie die Inhalte der Dateien `staedteFr.txt` und `staedteIt.txt` per `cat`-Befehl gemeinsam aus. Verbinden Sie dieses Kommando mit dem Befehl `grep` und lassen Sie auf diese Weise nur Einträge anzeigen, die ein großes B enthalten.

Lösungen:

1. `mv Beispielordner neuerName`
2. `gedit bsp.txt`
3. `cat staedteIt.txt staedteFr.txt | grep B`

Kapitel 8

Shell-Programme erstellen

In Kapitel 7 haben wir bereits die Nutzung der Shell kennengelernt. Dabei haben wir verschiedene Kommandos verwendet, um Dateien und Ordner zu verwalten, um Programme zu installieren und um viele weitere Aufgaben zu erledigen. Dabei handelte es sich jedoch bislang um einzelne Befehle, die jeweils eine ganz bestimmte Aktion zur Folge haben. Die gleichen Aufgaben lassen sich auch über eine grafische Benutzeroberfläche erledigen – sowohl unter Linux als auch unter Windows und macOS. Die Shell bietet jedoch noch zahlreiche weitere Möglichkeiten.

Beispielsweise können wir hier Variablen definieren und diesen einen Wert zuweisen. Außerdem können wir Bedingungen aufstellen und damit eine bestimmte Aktion verbinden. Das bedeutet, dass diese nur ausgeführt wird, wenn der Wert einer Variable einer Vorgabe entspricht, die wir selbst aufstellen können. Außerdem ist es möglich, bestimmte Aufgaben in einer Schleife zu wiederholen. Das entspricht den Aufgaben, die ein gewöhnliches Computerprogramm durchführt. Wenn Sie mit Linux arbeiten, können Sie die entsprechenden Befehle jedoch direkt in die Shell eingeben. Das ist ausgesprochen praktisch. Auf diese Weise können Sie kleine Programme erstellen – sogenannte Shell Scripts. Diese erlauben es, vielfältige Aufgaben auszuführen. Da sie direkt in der Shell ablaufen, ist es hierbei auch möglich, alle bereits bekannten Befehle zur Steuerung des Systems zu verwenden. Auf diese Weise ist es möglich, hierfür automatische *Routinen* zu erstellen.

8.1 Was ist ein Shell-Script?

In der Einleitung zu diesem Kapitel wurde bereits ein grober Überblick darüber gegeben, was ein Shell Script ist. Um dieses Konzept zu verstehen, sind jedoch noch einige weitere Erklärungen erforderlich. Bevor wir mit dem praktischen Teil dieses Kapitels beginnen, soll daher genau

vorgestellt werden, was ein Shell Script ist und welche Anwendungsmöglichkeiten es bietet.

Dazu müssen wir uns zunächst mit den gewöhnlichen Abläufen beim Erstellen eines Computerprogramms befassen. Hierfür kommen viele verschiedene Programmiersprachen zum Einsatz – beispielsweise C, C++, Python, Java oder PHP. Diese weisen unterschiedliche Befehle auf und eignen sich für verschiedene Anwendungen. Dabei besteht jedoch eine Gemeinsamkeit: Die Befehle, die die Programmiersprachen verwenden, lassen sich nicht direkt auf dem Prozessor ausführen. Zum einen arbeitet dieser ausschließlich mit binären Kommandos. Die entsprechenden Programmiersprachen verwenden hingegen Befehle, die aus Buchstaben, Zahlen und weiteren Zeichen bestehen. Zum anderen lässt sich ein Befehl einer höheren Programmiersprache normalerweise nicht direkt in einen Prozessor-Befehl übertragen. Hierfür ist es häufig notwendig, eine größere Anzahl an Kommandos auszuführen.

Die Befehle, die der Prozessor ausführen kann, werden als Maschinensprache bezeichnet. Um ein Programm, das in einer Programmiersprache geschrieben ist, ausführen zu können, ist es zunächst notwendig, dieses in die Maschinensprache zu übertragen. Hierfür kommen verschiedene Vorgehensweisen infrage. Zum einen ist es möglich, einen Compiler zu verwenden. Dieser übernimmt den Übersetzungsprozess und speichert dessen Ergebnis in einer eigenen Datei ab. Hierbei handelt es sich dann um eine ausführbare Datei. Diese lässt sich beliebig oft ausführen, ohne dass es notwendig ist, den Übersetzungsprozess erneut durchzuführen. Eine weitere Möglichkeit besteht darin, einen Interpreter zu verwenden. Dieser führt den entsprechenden Übersetzungsprozess bei jeder Ausführung erneut aus. Das bedeutet, dass keine ausführbare Datei entsteht. Das Programm wird nur als Code abgespeichert. In diesem Fall spricht man von einer Scriptsprache.

Die Shell verfügt über die Funktion eines solchen Interpreters. Das bedeutet, dass sie in der Lage dazu ist, die entsprechenden Kommandos eines Computerprogramms in die Maschinensprache zu übersetzen und auszuführen. Bei solchen Programmen handelt es sich um Shell Scripts. Deren Besonderheit besteht darin, dass sie direkt in der Shell

ausgeführt werden, ohne dass dafür ein weiterer Interpreter erforderlich ist.

Ein Shell Script ist daher nichts anderes als ein kleines Computerprogramm. Wie bei jedem Programm in einer anderen Programmiersprache ist es auch hierbei notwendig, den Code in die Maschinensprache zu übertragen. Die Besonderheit bei einem Shell Script besteht lediglich darin, dass diesen Übersetzungsprozess die Shell selbst übernimmt.

Daraus folgt, dass Sie das Shell Script in einer Programmiersprache verfassen müssen. Jede Sprache verfügt über ihre eigenen Kommandos und Syntax-Regeln. Für jede von ihnen ist ein Compiler oder ein Interpreter erforderlich, der die entsprechenden Regeln überprüft, die Befehle in die Maschinensprache überträgt und damit eine Ausführung ermöglicht. Im Rahmen dieses Kapitels stellen wir Ihnen die grundlegenden Regeln und Kommandos vor, die Sie für das Shell Script benötigen. Das bedeutet, dass Sie hierbei einige Programmierkenntnisse erwerben.

Viele Leser fragen sich an dieser Stelle sicherlich, um welche Programmiersprache es sich dabei handelt. Hierbei sei zunächst angemerkt, dass es in diesem Fall nicht üblich ist, die Programmiersprache zu benennen. Man spricht lediglich von Shell Scripts. Deshalb sind die Sprachen, die hierfür zum Einsatz kommen, im Gegensatz zu den oben aufgeführten Beispielen nicht allzu geläufig. Wenn man sie dennoch nennen will, kommen hierfür die Bezeichnungen für die jeweiligen Shells zum Einsatz – wie etwa `bash`, `sh`, `csh` oder `ksh`. In diesem Buch befassen wir uns ausschließlich mit `bash`. Allerdings sind die Unterschiede zu den übrigen Sprachen nur gering. Die meisten Beispiele lassen sich auch mit den weiteren aufgeführten Sprachen ohne Anpassungen umsetzen.

Bei der Gestaltung von Shell Scripts gibt es noch eine weitere wichtige Besonderheit: Für die meisten Computerprogramme müssen wir eine eigene Datei für den Code erstellen. Das ist bei der Gestaltung von Shell Scripts selbstverständlich ebenfalls möglich. Allerdings gibt es hierbei noch eine weitere Möglichkeit: Sie können den Programmcode auch direkt in die Shell eintippen. Das ist besonders schnell und einfach und eignet sich daher bestens zum Ausprobieren kleiner Code-Beispiele. Aller-

dings bleiben die Programme auf diese Weise nicht dauerhaft erhalten. Daher eignet sich diese Möglichkeit ausschließlich, um das Programmieren zu üben oder um einmalige Aktionen durchzuführen. Wenn Sie jedoch eine Anwendung programmieren, die Sie dauerhaft nutzen möchten, ist es notwendig, hierfür eine eigene Datei zu erstellen.

8.2 Die ersten Shell Scripts erstellen

Die wohl einfachste Aufgabe, die ein Computerprogramm erledigen kann, besteht darin, einen einfachen Text auszugeben. Daher wählen wir diese Funktion für unser erstes Programmbeispiel aus. Hierzu kommt der Befehl `echo` zum Einsatz. Der Text, der ausgegeben wird, steht in Anführungszeichen. Unser erstes Programm soll den folgenden Text ausgeben: „Mein erstes Shell Script!“. Das führt zu folgendem Befehl:

```
1 echo "Mein erstes Shell Script!"
```

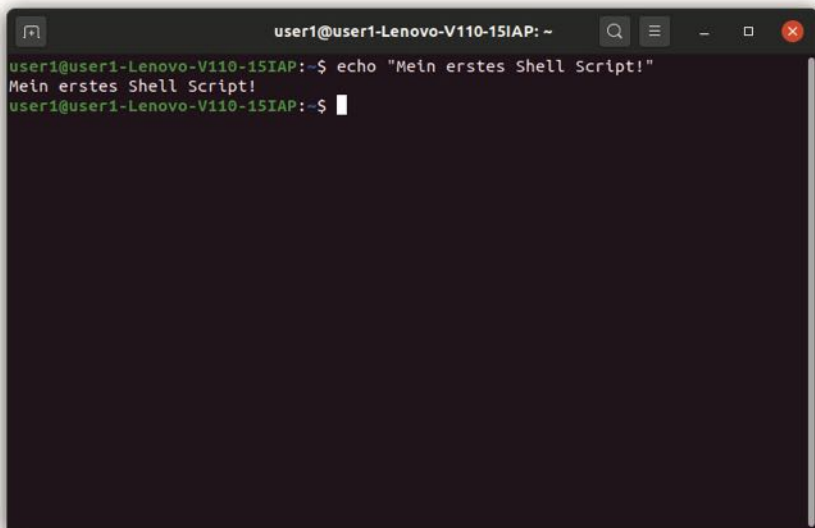
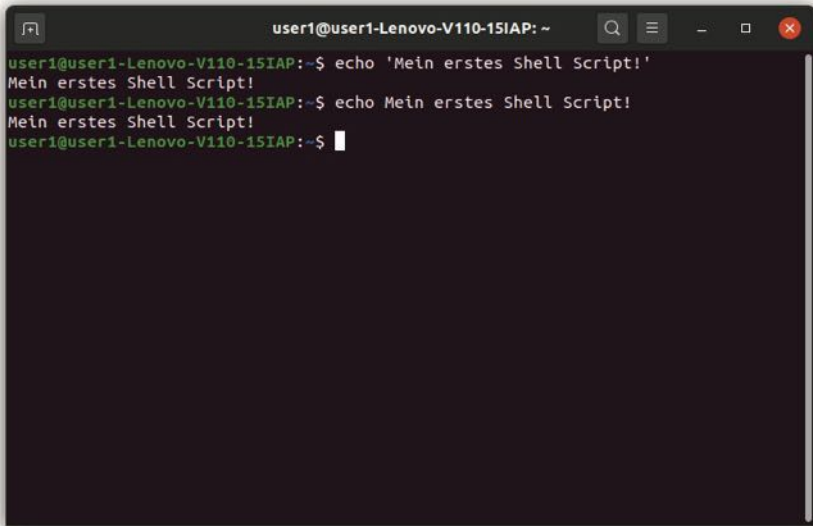
A screenshot of a terminal window with a dark background. The title bar at the top reads 'user1@user1-Lenovo-V110-15IAP: ~'. The terminal shows a prompt 'user1@user1-Lenovo-V110-15IAP:~\$' followed by the command 'echo "Mein erstes Shell Script!"'. The output 'Mein erstes Shell Script!' is displayed on the next line. The prompt 'user1@user1-Lenovo-V110-15IAP:~\$' appears again on the third line, followed by a cursor. The terminal window has standard window controls (minimize, maximize, close) in the top right corner.

Abb. 8.1 Die Textausgabe über die Shell

In Abbildung 8.1 ist zu sehen, dass unser Programm den Text, den wir im `echo`-Befehl angegeben haben, in der nächsten Zeile des Kommandozeileninterpreters ausgibt. Das ist die typische Funktionsweise bei einem Konsolenprogramm – auch bei anderen Programmiersprachen wie C, Python oder Java. Die Interaktion mit dem Anwender findet hierbei über den Kommandozeileninterpreter statt. Dieser dient der Ausgabe und der Aufnahme von Eingaben durch den Anwender. Das stellt einen erheblichen Unterschied zu den meisten modernen Computerprogrammen dar, die mit einer grafischen Benutzeroberfläche arbeiten. Zwar ist es auch möglich, solche fensterbasierten Programme mit einem Shell Script zu erstellen. Allerdings sind hierfür externe Frameworks und fortgeschrittene Programmierkenntnisse erforderlich. Außerdem stellt das Shell Script hierfür nicht das optimale Werkzeug dar. Aus diesen Gründen arbeiten wir in diesem Buch ausschließlich mit Konsolenprogrammen.

In unserem ersten Beispiel haben wir den Text in doppelte Anführungszeichen gestellt. Für die Ausgabe gibt es aber noch weitere Alternativen. Beispielsweise ist es möglich, auch einfache Anführungszeichen zu verwenden. Eine große Besonderheit bei Shell Scripts besteht darin, dass es auch erlaubt ist, auf die Anführungszeichen komplett zu verzichten. In fast allen übrigen gängigen Programmiersprachen ist es zwingend notwendig, Textausgaben durch Anführungszeichen kenntlich zu machen. Abbildung 8.2 zeigt, dass Shell Scripts hierbei jedoch eine Ausnahme darstellen und dass die Ausgabe auch ohne Anführungszeichen funktioniert.



```
user1@user1-Lenovo-V110-15IAP: ~  
user1@user1-Lenovo-V110-15IAP:~$ echo 'Mein erstes Shell Script!'  
Mein erstes Shell Script!  
user1@user1-Lenovo-V110-15IAP:~$ echo Mein erstes Shell Script!  
Mein erstes Shell Script!  
user1@user1-Lenovo-V110-15IAP:~$
```

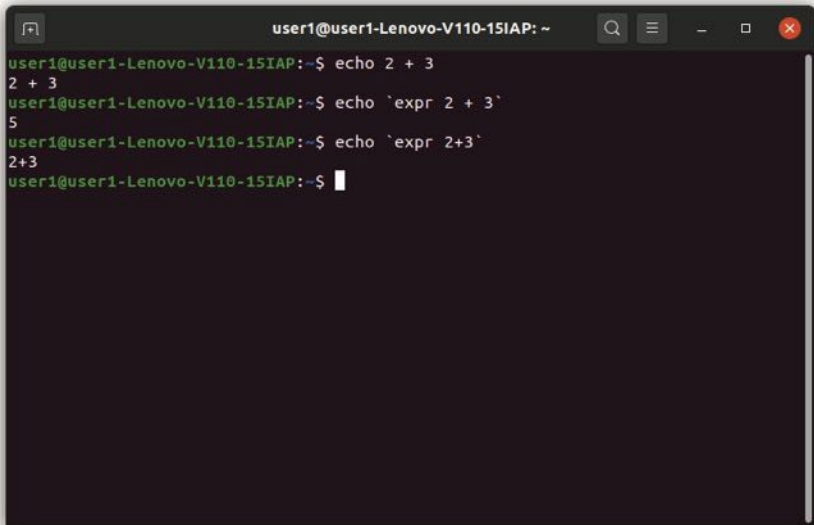
Abb. 8.2 Die Ausgabe mit einfachen Anführungszeichen und ganz ohne Anführungszeichen

Einfache Berechnungen durchführen

Wenn Sie bereits mit anderen Programmiersprachen gearbeitet haben, dann wissen Sie sicherlich, dass es bei der Ausgabe auch möglich ist, einfache Berechnungen durchzuführen. Auch das probieren wir nun aus, indem wir eine Summe bilden: `echo 2 + 3`. In Abbildung 8.3 ist zu sehen, dass wir auf diese Weise jedoch nicht ans Ziel kommen. Das Programm gibt hierbei genau die Zeichen aus, die wir auch eingegeben haben – also `2 + 3`. Auch dieses Verhalten stellt einen wesentlichen Unterschied zu den meisten übrigen Programmiersprachen dar, die bei einem entsprechenden Ausdruck das Ergebnis direkt berechnen.

Doch ist es auch in einem Shell Script möglich, eine entsprechende Berechnung durchzuführen. Hierfür ist es jedoch notwendig, vor der Berechnung den Ausdruck `expr` einzufügen. Der komplette Ausdruck – also der Begriff `expr` sowie der Rechenausdruck – muss in sogenann-

ten backticks stehen. Diese werden auch als backquotes oder im deutschen Zeichensatz als Gravis bezeichnet. Diese sind nicht mit den einfachen Anführungszeichen zu verwechseln, die wir zuvor für unsere Zeichenkette verwendet haben. Die backticks erreichen Sie bei gedrückter Umschalttaste über das Akzentzeichen, das sich bei einer deutschen Tastaturbelegung links neben dem Eszett (ß) befindet. Anschließend müssen Sie die Space-Taste drücken. In Abbildung 8.3 ist zu sehen, dass auf diese Weise die Berechnung durchgeführt wird. Der entsprechende Befehl sieht dann wie folgt aus: `echo `expr 2 + 3``



```
user1@user1-Lenovo-V110-15IAP: ~  
user1@user1-Lenovo-V110-15IAP:~$ echo 2 + 3  
2 + 3  
user1@user1-Lenovo-V110-15IAP:~$ echo `expr 2 + 3`  
5  
user1@user1-Lenovo-V110-15IAP:~$ echo `expr 2+3`  
2+3  
user1@user1-Lenovo-V110-15IAP:~$
```

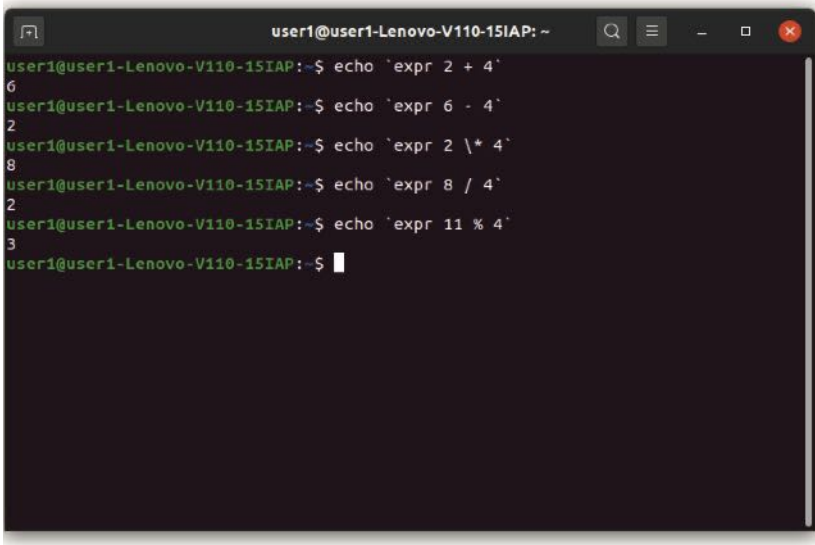
Abb. 8.3 Die Durchführung der Berechnung sowie mögliche Fehlerquellen

In der obigen Abbildung ist noch ein weiterer Ausdruck zu sehen. Dieser ist beinahe identisch zum eben dargestellten Befehl. Der einzige Unterschied besteht darin, dass wir hierbei die Leerzeichen vor und nach dem Pluszeichen entfernt haben. Auch das kann für Leser, die bereits über Programmiererfahrung in anderen Sprachen verfügen, für Verwirrung sorgen. In fast allen gängigen Programmiersprachen macht es keinerlei Unterschied, ob wir diese Leerzeichen verwenden oder nicht.

In unserem Shell Script ist dies jedoch von großer Bedeutung. Wenn wir die Leerzeichen weglassen, wird wieder nur der Rechenausdruck ausgegeben, nicht jedoch das Ergebnis.

Unser Programm erlaubt es nicht nur, Additionen durchzuführen. Auch alle übrigen Grundrechenarten sind hierbei möglich. Hinzu kommt der sogenannte Modulo-Operator. Dieser gibt den Rest einer ganzzahligen Division an. Wenn Sie sich nochmals an die Grundschule zurückerinnern, haben Sie hierbei bei Divisionen sicherlich nicht mit Kommazahlen gearbeitet. Wenn Sie beispielsweise das Ergebnis aus $11 \div 4$ berechnen mussten, haben Sie als Ergebnis 2 Rest 3 angegeben. Der Modulo-Operator berechnet genau diesen Rest. Hierfür kommt das Prozentzeichen zum Einsatz. Wenn Sie den Ausdruck $11 \% 4$ in Ihrem Programm berechnen, ist dessen Ergebnis daher 3. Die folgende Tabelle gibt einen Überblick über die Rechenoperationen und die Zeichen, die hierfür jeweils zum Einsatz kommen. Abbildung 8.4 stellt diese Beispiele dar.

Operation	Zeichen	Beispiel
Addition	+	$2 + 4 = 6$
Subtraktion	-	$6 - 4 = 2$
Multiplikation	*	$2 \backslash * 4 = 8$
Division	/	$8 / 4 = 2$
Modulo	%	$11 \% 4 = 3$

A screenshot of a terminal window titled 'user1@user1-Lenovo-V110-15IAP: ~'. The terminal shows a series of commands and their outputs. The commands are: 'echo `expr 2 + 4`', 'echo `expr 6 - 4`', 'echo `expr 2 \\* 4`', 'echo `expr 8 / 4`', and 'echo `expr 11 % 4`'. The outputs are 6, 2, 8, 2, and 3 respectively. The prompt 'user1@user1-Lenovo-V110-15IAP:~\$' is visible at the end of each line.

```
user1@user1-Lenovo-V110-15IAP:~$ echo `expr 2 + 4`  
6  
user1@user1-Lenovo-V110-15IAP:~$ echo `expr 6 - 4`  
2  
user1@user1-Lenovo-V110-15IAP:~$ echo `expr 2 \* 4`  
8  
user1@user1-Lenovo-V110-15IAP:~$ echo `expr 8 / 4`  
2  
user1@user1-Lenovo-V110-15IAP:~$ echo `expr 11 % 4`  
3  
user1@user1-Lenovo-V110-15IAP:~$
```

Abb. 8.4 Mathematische Operationen in der Shell

Das Programm in einer Datei speichern

Bei unseren bisherigen Beispielen haben wir die Befehle direkt in die Shell eingegeben. Man spricht in diesem Fall von einer interaktiven Anwendung. Diese ist zwar sehr praktisch, um die Befehle schnell auszuführen. Wenn wir jedoch ein Programm schreiben möchten, das öfters zum Einsatz kommt, ist es notwendig, dieses in einer separaten Datei zu verfassen. Auch das ist beim Erstellen eines Shell Scripts möglich.

Zu diesem Zweck benötigen wir einen Texteditor. In diesem Buch haben wir hierfür bereits zahlreiche Beispiele vorgestellt. Sie können beispielsweise den integrierten Texteditor gedit oder eine Software mit etwas umfangreicheren Funktionen wie etwa Atom verwenden. Auch die Editoren, die innerhalb der Shell ablaufen – wie etwa der Nano Editor oder VIM – eignen sich hierfür sehr gut. Die Auswahl bleibt hierbei Ihnen überlassen. Im folgenden Beispiel stellen wir die Erstellung in gedit vor. Doch sind für die Verwendung der übrigen Editoren nur minimale

Anpassungen bei der Vorgehensweise notwendig. Sehr wichtig ist es jedoch, darauf zu achten, kein Textverarbeitungsprogramm wie etwa den LibreOffice Writer zu verwenden. Damit wäre es nicht möglich, das Programm zu erstellen.

Um das Programm in einer eigenen Datei abzuspeichern, öffnen wir zunächst den Texteditor. Wenn wir ein Shell Script in einer eigenen Datei erstellen, ist es zu Beginn stets notwendig, den sogenannten Shebang anzugeben. Dabei handelt es sich um eine kurze Angabe, mit welchem Interpreter das Script ausgeführt werden soll. Der Shebang beginnt stets mit dem Rautezeichen und dem Ausrufezeichen. Danach folgt der Pfad zum Interpreter. Wir wollen unser Shellscrip mit bash ausführen. Der entsprechende Pfad hierfür lautet `/bin/bash`. Der komplette Shebang sieht dann so aus: `#!/bin/bash`. Es wurde bereits gesagt, dass wir in diesem Buch stets mit der Bash arbeiten. Wenn Sie aber einmal eine andere Shell verwenden möchten, können Sie dies ebenfalls über den Shebang vorgeben. Die übrigen installierten Shells erreichen Sie über die folgenden Ausdrücke:

- ▶ `#!/bin/sh`
- ▶ `#!/bin/rbash`
- ▶ `#!/bin/dash`

Nach dieser Angabe können Sie beliebige Kommandos einfügen. Im Gegensatz zu unseren bisherigen Beispielen ist es hierbei auch möglich, mehrere Befehle aneinanderzureihen. Dabei sind wir auch nicht auf den `echo`-Befehl beschränkt, den wir in den bisherigen Beispielen verwendet haben. Auch die Kommandos, die wir in Kapitel 7 für die Steuerung des Systems kennengelernt haben, lassen sich auf diese Weise ausführen. Das folgende Programm stellt die verschiedenen Möglichkeiten vor. Es gibt zunächst einen Text aus, führt daraufhin eine Berechnung durch und gibt anschließend den Inhalt des Ordners Dokumente mit dem `ls`-Befehl aus:

```
1 #!/bin/bash
2
3 echo "Programm in einer eigenen Datei"
```

8 Shell-Programme erstellen

```
4 echo 'expr 2 \* 3'
5 ls /home/user1/Dokumente
```

Bevor wir das Programm ausführen können, ist es notwendig, es abzuspeichern. Hierfür können wir einen beliebigen Namen auswählen. Als Endung müssen wir jedoch stets den Ausdruck `.sh` verwenden. Wir speichern unser erstes Beispielprogramm daher unter der Bezeichnung `beispiel1.sh` ab. In Abbildung 8.5 ist zu sehen, dass sich daraufhin die Darstellung im Texteditor ändert. Das liegt an der Funktion der Syntaxhervorhebung. Diese markiert die Schlüsselbegriffe einer Programmiersprache farbig, um den Code übersichtlicher zu gestalten.

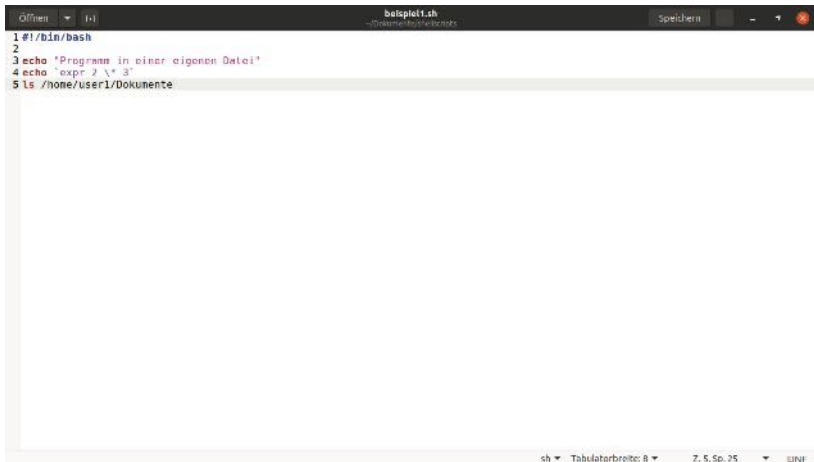


Abb. 8.5 Das Programm im Editor gedit mit Syntaxhervorhebung

Um das Programm aufzurufen, müssen wir zunächst wieder das Terminal öffnen. Hierbei bestehen nun zwei Möglichkeiten. Entweder wir rufen die Datei aus dem Verzeichnis auf, in dem wir uns gerade befinden. Hierfür ist es notwendig, den Pfad anzugeben. Wenn wir für unsere Datei im Ordner Dokumente ein Unterverzeichnis mit der Bezeichnung shellscripts erstellt haben, können wir das Programm mit dem relativen Pfad (`Dokumente/shellscripts/beispiel1.sh`) oder dem absoluten Pfad (`/home/user1/Dokumente/shellscripts/beispiel1.sh`) aufrufen. Eine weitere Möglichkeit besteht darin, zunächst mit dem `cd`-Befehl in den entsprechenden Ordner zu navigieren. Hier können

wir dann die Datei direkt aufrufen. Dazu müssen wir vor dem Dateinamen zunächst einen Punkt und danach einen Schrägstrich angeben: `./beispiel1.sh`.

In diesem Beispiel entscheiden wir uns für die letztgenannte Alternative und wechseln in den entsprechenden Ordner. Anschließend rufen wir die Datei auf. In Abbildung 8.6 ist jedoch zu sehen, dass die Ausführung nicht möglich ist. Das liegt daran, dass wir keine Berechtigung für diese Aktion haben. Das wird deutlich, wenn wir den Befehl `ls -l` ausführen. Da wir jedoch bereits wissen, wie das Recht zum Ausführen einer Datei hinzufügen können, sollte es kein Problem darstellen, die entsprechende Anpassung mit dem `chmod`-Befehl durchzuführen. Wir kontrollieren zunächst die Änderungen und probieren danach erneut, das Programm auszuführen. Abbildung 8.6 zeigt, dass dies nun möglich ist. Das Programm führt nacheinander alle Befehle aus, die wir in der entsprechenden Datei angegeben haben. Es ist wichtig, darauf zu achten, dass wir die Anpassung der Rechte bei jedem weiteren Shell Script, das wir erstellen, erneut ausführen müssen.

8

```

user1@user1-Lenovo-V110-15IAP: ~/Dokumente/shellscripts
user1@user1-Lenovo-V110-15IAP:~$ cd Dokumente/shellscripts
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$ ./beispiel1.sh
bash: ./beispiel1.sh: Keine Berechtigung
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$ ls -l
insgesamt 4
-rw-rw-r-- 1 user1 user1 96 Nov 17 17:30 beispiel1.sh
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$ chmod +x beispiel1.sh
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$ ls -l
insgesamt 4
-rwxrwxr-x 1 user1 user1 96 Nov 17 17:30 beispiel1.sh
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$ ./beispiel1.sh
Programm in einer eigenen Datei
6
Beispieltext.odt  staedteDeLang.txt  staedteEU.txt  staedteIt.txt
shellscripts    staedteDe.txt    staedteFr.txt
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$

```

Abb. 8.6 Die Anpassung der Rechte und die Ausführung der Datei

8.3 Variablen verwenden

Variablen bieten die Möglichkeit, einen Wert in einem Computerprogramm aufzunehmen und ihn zu einem späteren Zeitpunkt wieder zur Verfügung zu stellen. Dabei ist es auch möglich, den Inhalt zu verändern und Berechnungen oder andere Aktionen damit durchzuführen. *Variablen* stellen ein zentrales Element fast aller Computerprogramme dar. Ohne Variablen ist es nur möglich, ausgesprochen einfache Aktionen durchzuführen – beispielsweise feste Textausgaben wie im vorigen Abschnitt. Der Sinn eines Computerprogramms besteht jedoch in der Regel darin, die Aktionen und Abläufe an bestimmte Eingaben und Bedingungen anzupassen. Das bedeutet, dass die Werte hierbei nicht fest sind. Daraus folgt, dass wir Variablen benötigen, die unterschiedliche Werte aufnehmen können.

Wenn wir mit Variablen arbeiten, ist es zunächst notwendig, einen Namen dafür auszuwählen. Dabei muss es sich um eine eindeutige Bezeichnung handeln. Es ist nicht möglich, verschiedene Variablen mit dem gleichen Namen zu verwenden. Das ist wichtig, da wir bei jeder Aktion genau wissen müssen, auf welche Variable sie sich beziehen soll. Das ist nur bei der Verwendung eindeutiger Bezeichner möglich. Um den Namen für eine neue Variable vorzugeben, können wir eine beliebige Kombination aus Buchstaben, Ziffern und dem Unterstrich verwenden – allerdings unter der Bedingung, dass der Variablenname mit einem Buchstaben und nicht mit einer Ziffer beginnt. Dabei müssen wir darauf achten, dass der Variablenname noch nicht in Verwendung ist und dass dieser nicht identisch zu einem Shell-Befehl ist. Beispielsweise wäre es nicht möglich, eine Variable `mkdir` zu nennen, da dieser Ausdruck bereits mit einem anderen Befehl verknüpft ist.

Bei der Auswahl der Variablennamen ist es empfehlenswert, auf aussagekräftige Bezeichnungen zu achten. Wenn wir beispielsweise eine Variable mit der Bezeichnung `variable1` erstellen, wissen wir in diesem Moment sicherlich genau, welchen Wert wir hier aufnehmen. Wenn wir das Programm nach einigen Wochen erneut betrachten, haben wir dies jedoch wahrscheinlich bereits vergessen. Falls wir eine Änderung daran vornehmen wollen, ist daher eine umfangreiche Einarbeitung notwen-

dig. Daher ist es besser, Bezeichnungen wie `mitarbeiter`, `gehalt` oder `artikelnummer` zu verwenden, aus denen genau hervorgeht, welcher Inhalt in der Variable gespeichert ist. Es ist üblich, Variablennamen komplett klein zu schreiben. Wenn diese aus mehreren Wörtern bestehen sollen, können Sie diese mit einem Unterstrich verbinden. Leerzeichen sind hierbei nicht erlaubt. Dabei ist es wichtig, darauf zu achten, dass das Programm zwischen Groß- und Kleinschreibung unterscheidet. Daher handelt es sich bei `VARIABLE1`, `Variable1` und `variable1` jeweils um unterschiedliche Variablen

Bei Shell Scripts ist es im Gegensatz zu vielen anderen Programmiersprachen nicht notwendig, den Variablentyp anzugeben. Wir können hier sowohl ganze Zahlen als auch Fließkommazahlen oder Zeichenketten aufnehmen. Die gleiche Variable kann im Verlauf eines Programms sogar unterschiedliche Datentypen aufnehmen.

Um eine Variable zu definieren, müssen wir zunächst deren Namen angeben. Danach folgt ein Gleichheitszeichen. Anschließend müssen wir den Wert angeben, den die Variable aufnehmen soll. Auch hierbei weisen Shell Scripts eine Besonderheit auf, die in kaum einer anderen Programmiersprache zu finden ist: Weder vor noch nach dem Gleichheitszeichen darf ein Leerzeichen stehen. In den meisten Programmiersprachen spielt dies keine Rolle. Um eine Variable zu definieren, können wir beispielsweise den folgenden Befehl verwenden:

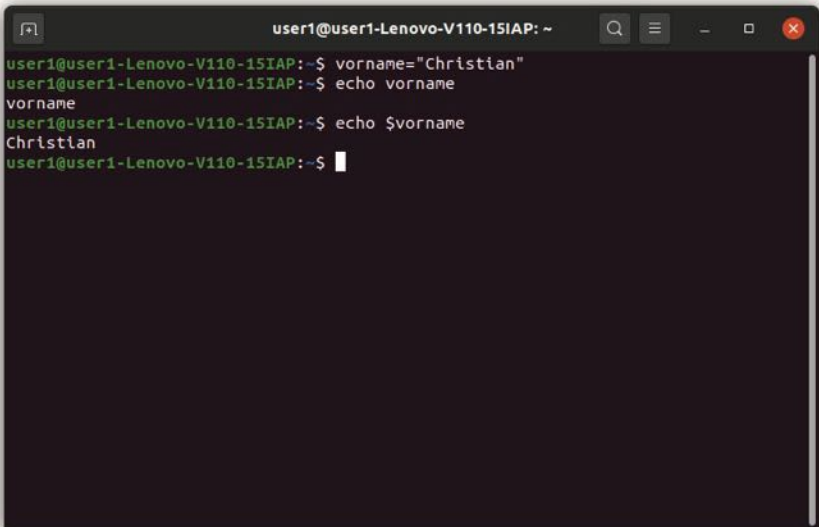
```
1 vorname="Christian"
```

Durch dieses Kommando wird die Zeichenkette „Christian“ in der Variable `vorname` gespeichert. Auf diese Weise steht der entsprechende Inhalt auch im weiteren Verlauf des Programms zur Verfügung. Nun probieren wir aus, den Inhalt der Variable über den `echo`-Befehl auszugeben. Dazu stellen wir einfach den Variablennamen hinter das entsprechende Kommando, so wie dies in vielen anderen Programmiersprachen üblich ist. Dabei gibt das Programm jedoch lediglich den Variablennamen aus – nicht jedoch ihren Inhalt. Das liegt daran, dass es im Shell Script nicht notwendig ist, Zeichenketten durch die Verwendung von Anführungszeichen kenntlich zu machen. Daher geht das

8 Shell-Programme erstellen

Programm davon aus, dass es sich beim Ausdruck `vorname` um eine gewöhnliche Zeichenkette handelt und gibt diese aus. Um deutlich zu machen, dass wir den Inhalt der Variable ausgeben möchten, müssen wir den Bezeichner als Variablennamen kenntlich machen. Dazu stellen wir das Dollarzeichen voran. Der entsprechende Ausdruck sieht dann so aus:

```
1 echo $vorname
```

A screenshot of a terminal window titled "user1@user1-Lenovo-V110-15IAP: ~". The terminal shows a sequence of commands and their outputs. First, the command `vorname="Christian"` is entered, followed by `echo vorname` which outputs `vorname`. Then, `echo $vorname` is entered, which outputs `Christian`. The prompt `user1@user1-Lenovo-V110-15IAP:~$` is visible at the end of the last line.

```
user1@user1-Lenovo-V110-15IAP:~$ vorname="Christian"
user1@user1-Lenovo-V110-15IAP:~$ echo vorname
vorname
user1@user1-Lenovo-V110-15IAP:~$ echo $vorname
Christian
user1@user1-Lenovo-V110-15IAP:~$
```

Abb. 8.7 Die Definition und die Ausgabe einer Variable

In Abbildung 8.7 ist zu sehen, dass wir bei einer interaktiven Anwendung die entsprechenden Befehle einfach nacheinander in das Terminal eingeben können. Die Shell registriert die Eingabe und speichert die entsprechende Variable ab. Ihr Wert bleibt auch bei der Eingabe eines neuen Kommandos verfügbar – so lange, bis wir das Terminal schließen. Wenn wir hingegen eine separate Datei für das Programm erstellen, ist die Variable nur innerhalb dieses Programms verfügbar. Wenn die Ausführung abgeschlossen ist, ist deren Wert innerhalb der

Shell nicht definiert. Das zeigt Abbildung 8.8. Beim Abruf der Variable erscheint eine leere Zeile, da der entsprechende Wert nicht definiert ist – obwohl wir ihn zuvor in unserem Shell Script vorgegeben haben. Hierfür erstellen wir ein Programm, das genau wie gerade vorgestellt eine Variable definiert und dann ausgibt:

```
1 #!/bin/bash
2
3 vorname="Christian"
4 echo $vorname
```

8

```
user1@user1-Lenovo-V110-15IAP: ~/Dokumente/shellscripts
user1@user1-Lenovo-V110-15IAP:~$ cd Dokumente/shellscripts
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$ chmod +x beispiel2.sh
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$ ./beispiel2.sh
Christian
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$ echo $vorname

user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$
```

Abb. 8.8 Die Variablen in separaten Programmen werden nicht in der Shell gespeichert

Der Wert einer Variable kann sich im Laufe des Programms verändern. Dazu ist es lediglich notwendig, ihr über das Gleichheitszeichen einen neuen Wert zuzuweisen. Das zeigt das folgende Programm. Dieses erstellt eine Variable mit einem Vornamen und gibt diesen aus. Danach

8 Shell-Programme erstellen

speichert es einen anderen Namen in der Variable ab und gibt diese erneut aus:

```
1  #!/bin/bash
2
3  vorname="Christian"
4  echo $vorname
5  vorname="Nina"
6  echo $vorname
```

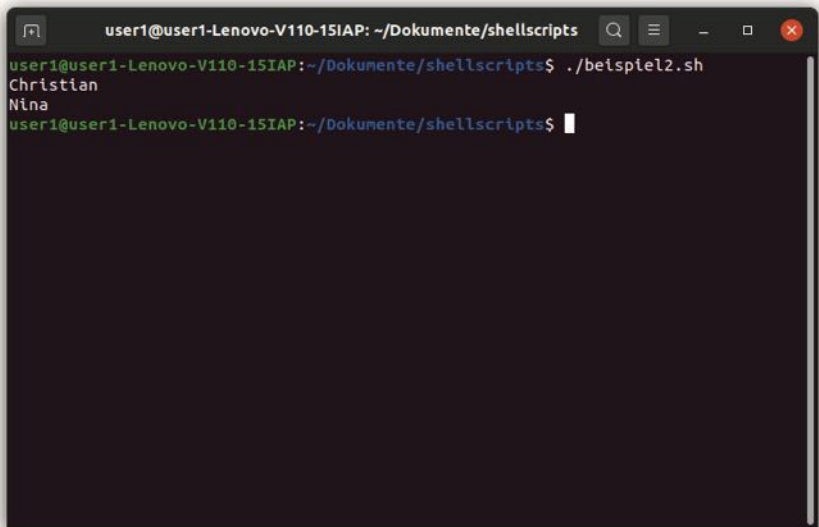


Abb. 8.9 Die Ausgabe der gleichen Variable nach der Änderung ihres Werts

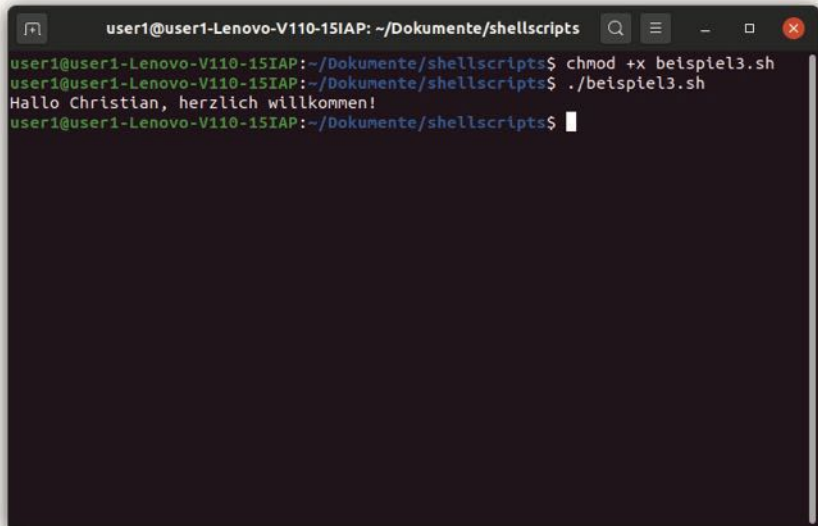
Dieses Programm verwendet zweimal einen vollkommen identischen `echo`-Befehl. Durch die Veränderung des Variablenwerts kommt es dabei jedoch zu unterschiedlichen Ausgaben. Das ist in Abbildung 8.9 zu sehen.

Oftmals ist es notwendig, Variablen gemeinsam mit einem anderen Text auszugeben. Auch das stellt kein Problem dar. Dazu ist es lediglich notwendig, den gewünschten Ausgabetext in den `echo`-Befehl einzu-

geben. Die Variable fügen wir dann unter Verwendung des Dollarzeichens in diesen Text ein. Das Programm ersetzt diesen bei der Ausgabe dann durch den Wert der Variable. Das ist in Abbildung 8.10 zu sehen. Der Code hierfür sieht wie folgt aus:

```
1 #!/bin/bash
2
3 vorname="Christian"
4 echo "Hallo $vorname, herzlich willkommen!"
```

8

A screenshot of a terminal window titled "user1@user1-Lenovo-V110-15IAP: ~/Dokumente/shellscripts". The terminal shows the following commands and output:

```
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$ chmod +x beispiel3.sh
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$ ./beispiel3.sh
Hallo Christian, herzlich willkommen!
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$
```

Abb. 8.10 Die Kombination von festem Text mit einer Variable

Variablen können auch Zahlen aufnehmen. Wenn wir die Zahlenwerte lediglich zuweisen und ausgeben, ist die Verwendung genau die gleiche wie bei Texten. Allerdings ist es mit Zahlen auch möglich, Berechnungen durchzuführen. Dazu benötigen wir wieder den Ausdruck `expr`. Beispielsweise können wir mit dem folgenden Befehl der Variable `summe` das Ergebnis des Rechenausdrucks $2 + 5$ zuweisen:

```
1 summe='expr 2 + 5'
```

8 Shell-Programme erstellen

Wir können den Wert einer Variable auch für weitere Berechnungen verwenden. Dabei ist es wichtig, darauf zu achten, dass wir beim Zugriff auf ihren Wert stets das Dollarzeichen verwenden müssen. Beispielsweise können wir mit dem folgenden Befehl der Variable `produkt` den dreifachen Wert der Variable `summe` zuweisen:

```
1 produkt='expr $summe \'* 3'
```

Es ist auch möglich, den bisherigen Wert einer Variable in die Berechnung einzubeziehen. Häufig ist es beispielsweise notwendig, den Wert einer Variable um 1 zu erhöhen. Dazu müssen wir für die Berechnung genau diesen Variablennamen angeben und den Wert 1 hinzu addieren:

```
1 summe='expr $summe + 1'
```

Wenn wir diese Befehle in einem Programm zusammenführen und dabei nach jeder Berechnung die Werte ausgeben, führt das zum unten dargestellten Code. Das Ergebnis ist in Abbildung 8.11 zu sehen.

```
1 #!/bin/bash
2
3 summe='expr 2 + 5'
4 echo "Wert der Variable summe: $summe"
5 produkt='expr $summe \'* 3'
6 echo "Wert der Variable produkt: $produkt"
7 summe='expr $summe + 1'
8 echo "Wert der Variable summe nach der Veränderung: $summe"
```

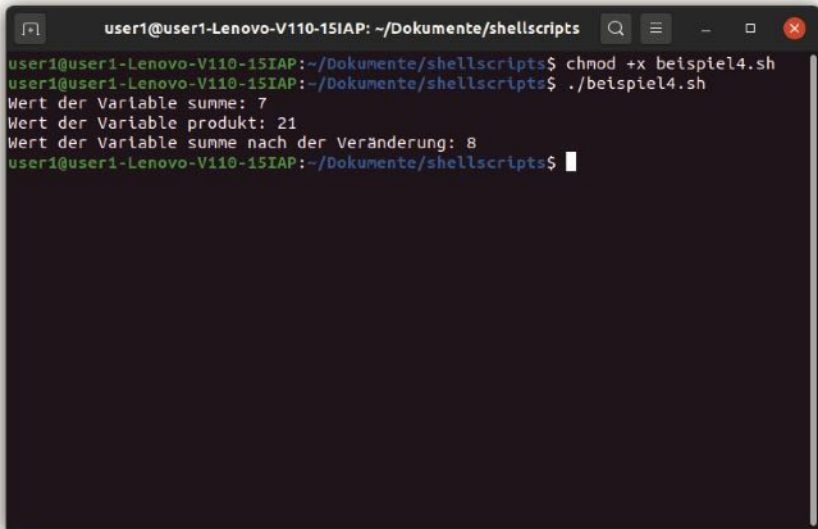
A terminal window titled 'user1@user1-Lenovo-V110-15IAP: ~/Dokumente/shellscripts'. The prompt is 'user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts\$'. The first command is 'chmod +x beispiel4.sh'. The second command is './beispiel4.sh'. The output shows: 'Wert der Variable summe: 7', 'Wert der Variable produkt: 21', and 'Wert der Variable summe nach der Veränderung: 8'. The prompt returns to 'user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts\$'.

Abb. 8.11 Verschiedene Berechnungen mit Variablen durchführen

8.4 Arrays

Die Variablen, die wir in Kapitel 8.3 kennengelernt haben, bieten lediglich die Möglichkeit, einen einzelnen Wert aufzunehmen. In der Praxis kommt es jedoch häufig vor, dass mehrere zusammengehörige Werte auftreten, die gemeinsam verwaltet werden sollen. Als Beispiel hierfür können wir die Mitarbeiter eines Unternehmens heranziehen. Hierfür wäre es nun selbstverständlich möglich, separate Variablen zu erstellen:

```
1 mitarbeiterR1="Gerda Mayer"
2 mitarbeiter2="Carsten Frei"
3 mitarbeiter3="Sabrina Gruber"
4 mitarbeiter4="Oliver Hauser"
```

Bereits bei dieser kurzen Zusammenstellung wird deutlich, dass der Aufwand für diese Art der Datenverwaltung sehr groß ist. Häufig sind

die Datensätze außerdem noch deutlich umfangreicher. Viele Unternehmen haben mehrere Hundert Mitarbeiter oder Kunden. Diese alle in einzelnen Variablen abzuspeichern, würde einen enormen Aufwand mit sich bringen. Deshalb ist es sinnvoll, diese in einem Array zusammenzufassen. Dieses kann man sich ähnlich wie eine Liste vorstellen. Es besteht aus mehreren Feldern, die jeweils einen eigenen Wert aufnehmen können. Das reduziert den Aufwand beim Umgang mit diesen Werten deutlich.

Bei der Verwendung einzelner Variablen kommt ein weiteres Problem hinzu. Häufig ist es notwendig, eine bestimmte Aktion auf alle Elemente eines Datensatzes anzuwenden. Beispielsweise kann es erforderlich sein, dass das Programm eine Liste mit allen Mitarbeitern ausgibt. Wenn wir hierfür einzelne Variablen verwenden, ist der Programmieraufwand nicht nur enorm. Darüber hinaus führt das zu sehr starren Strukturen. In diesem Fall müssen wir für jede Variable einen eigenen `echo`-Befehl erstellen. Nun kommt es jedoch häufig vor, dass ein neuer Mitarbeiter eingestellt wird oder dass eine Arbeitskraft den Betrieb verlässt. In diesem Fall wäre es jedes Mal notwendig, den Programmcode zu überarbeiten. Dabei gibt es auch Daten, bei denen die Fluktuation noch deutlich größer ist als bei den Mitarbeitern eines Unternehmens – beispielsweise bei der Verwaltung der Bestellungen. In diesem Fall wäre es vollkommen unmöglich, den Programmcode für jede neue Bestellung anzupassen. Doch auch in diesem Fall bieten Arrays zahlreiche Vorteile. Diese erlauben es, die Größe des Datensatzes variabel zu gestalten und bestimmte Aktionen auf alle Inhalte anzuwenden. Hierfür kommen Schleifen zum Einsatz. Diese werden wir noch in Kapitel 8.6 kennenlernen. Damit ist es beispielsweise möglich, alle Felder auszugeben – unabhängig von deren Anzahl. Daher sind keine Anpassungen am Programmcode erforderlich, wenn ein neuer Wert hinzukommt oder entfernt wird.

Nachdem die Vorteile und die Einsatzmöglichkeiten von Arrays vorgestellt wurden, lernen wir im nächsten Schritt, wie wir diese erstellen können. Hierfür bestehen unterschiedliche Möglichkeiten. In jedem Fall ist es notwendig, genau wie bei Variablen einen Namen für das Array vorzugeben. Dabei können wir wieder beliebige Buchstaben, Ziffern

sowie den Unterstrich verwenden. Genau wie bei Variablennamen gilt jedoch auch hierbei die Vorgabe, dass an erster Stelle keine Zahl stehen darf. Nach dem Namen, den wir gewählt haben, folgt nun jedoch eine eckige Klammer. Darin steht eine Zahl, die als Indexnummer bezeichnet wird. Diese gibt das Feld des entsprechenden Werts an. Dabei ist es wichtig, darauf zu achten, dass das erste Feld stets den Index 0 erhält. Nun können wir jedes Feld einzeln mit Werten füllen:

```
1 mitarbeiter[0]="Gerda Mayer"  
2 mitarbeiter[1]="Carsten Frei"  
3 mitarbeiter[2]="Sabrina Gruber"  
4 mitarbeiter[3]="Oliver Hauser"
```

8

Hierbei erstellen wir ein Array mit der Bezeichnung `mitarbeiter`. Das erste Feld erhält dabei den Eintrag „Gerda Mayer“, das zweite Feld den Eintrag „Carsten Frei“ und so weiter. Hierbei erscheint der Arbeitsaufwand bislang der gleiche. Dennoch erzeugen wir auf diese Weise bereits eine zusammengehörige Struktur, die alle Mitarbeiter erfasst.

Es gibt jedoch noch eine weitere Möglichkeit, um ein Array zu erstellen. Hierbei ist es ebenfalls notwendig, zunächst den Arraynamen zu nennen. Danach werden die Inhalte jedoch nicht einzeln vorgegeben. Stattdessen müssen Sie eine Klammer setzen und darin die Werte für alle Felder nacheinander eintragen – jeweils durch ein Leerzeichen voneinander getrennt:

```
1 mitarbeiter=("Gerda Mayer" "Carsten Frei" "Sabrina Gruber"  
2 "Oliver Hauser")
```

Auf diese Weise entsteht genau das gleiche Array wie im letzten Beispiel. Der Aufwand für die Erstellung ist jedoch wesentlich geringer.

Um mit Arrays zu arbeiten, ist es auch notwendig, auf die Inhalte zuzugreifen. Hierfür können wir den Inhalt eines einzelnen Feldes verwenden. Zu diesem Zweck ist es notwendig, den Arraynamen mit der gewünschten Indexnummer darzustellen. Dieser Ausdruck muss in einer geschweiften Klammer stehen. Davor steht das Dollarzeichen. Wenn wir beispielsweise den Inhalt des ersten Feldes unseres Arrays für die Mitarbeiter ausgeben wollen, ist dies mit folgendem Befehl möglich:

8 Shell-Programme erstellen

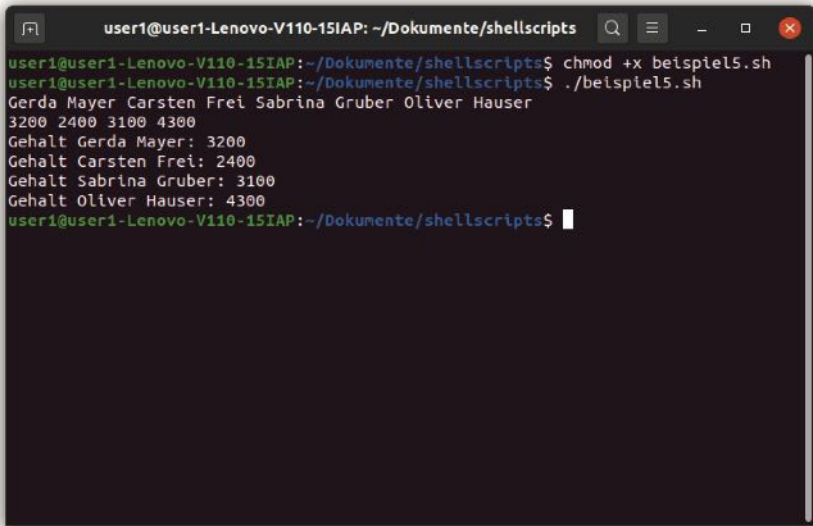
```
1 echo ${mitarbeiter[0]}
```

Genau nach dem gleichen Muster können wir auch auf alle übrigen Felder zugreifen. Manchmal ist es jedoch auch sinnvoll, das Array komplett auszugeben. In diesem Fall geben wir in der eckigen Klammer keine Indexnummer an, sondern entweder das Stern- oder das @-Zeichen. Zwischen diesen Möglichkeiten besteht keinerlei Unterschied. Die folgenden beiden Ausdrücke sind demnach vollkommen gleichbedeutend:

```
1 echo ${mitarbeiter[@]}
2 echo ${mitarbeiter[*]}
```

Beide Befehle geben das Array in einer Zeile aus. Die einzelnen Einträge sind dabei durch ein Leerzeichen voneinander getrennt. Zum Abschluss erstellen wir noch ein kleines Beispielsprogramm, das alle vorgestellten Möglichkeiten präsentiert. Dieses gestaltet ein Array für die Mitarbeiter und ein weiteres Array für deren Gehälter. Für die Erstellung verwenden wir unterschiedliche Vorgehensweisen. Daraufhin geben wir beide Arrays komplett aus und verwenden auch hierfür die beiden unterschiedlichen Methoden, die wir kennengelernt haben. Zum Abschluss greifen wir einzeln auf die Felder zu und geben dabei jeweils den Mitarbeiter und sein Gehalt aus. Das Ergebnis ist in Abbildung 8.12 zu sehen.

```
1  #!/bin/bash
2
3  mitarbeiter[0]="Gerda Mayer"
4  mitarbeiter[1]="Carsten Frei"
5  mitarbeiter[2]="Sabrina Gruber"
6  mitarbeiter[3]="Oliver Hauser"
7
8  gehalt=(3200 2400 3100 4300)
9
10 echo ${mitarbeiter[@]}
11 echo ${gehalt[*]}
12
13 echo "Gehalt ${mitarbeiter[0]}: ${gehalt[0]}"
14 echo "Gehalt ${mitarbeiter[1]}: ${gehalt[1]}"
15 echo "Gehalt ${mitarbeiter[2]}: ${gehalt[2]}"
16 echo "Gehalt ${mitarbeiter[3]}: ${gehalt[3]}"
```



```

user1@user1-Lenovo-V110-15IAP: ~/Dokumente/shellscripts
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$ chmod +x beispiel5.sh
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$ ./beispiel5.sh
Gerda Mayer Carsten Frei Sabrina Gruber Oliver Hauser
3200 2400 3100 4300
Gehalt Gerda Mayer: 3200
Gehalt Carsten Frei: 2400
Gehalt Sabrina Gruber: 3100
Gehalt Oliver Hauser: 4300
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$

```

Abb. 8.12 Die Ausgabe der Array-Inhalte

8.5 Einnahmen durch den Anwender aufnehmen

Ein Computerprogramm soll in der Regel nicht immer genau nach dem gleichen Muster ablaufen. Stattdessen soll eine Interaktion mit dem Anwender möglich sein. Auf diese Weise kann er den Ablauf steuern, verschiedene Werte eingeben und auf vielfältige weitere Arten Einfluss auf das Programm nehmen. Zu diesem Zweck ist es notwendig, Eingaben durch den Anwender aufzunehmen. Mit diesem Thema befassen wir uns in den nächsten Absätzen.

Um eine Eingabe durch den Anwender aufzunehmen, kommt der Befehl `read` zum Einsatz. An diesen fügen wir den Namen einer Variable an. In dieser nehmen wir den Wert auf, den der Anwender eingibt. Anschließend können wir den Inhalt der Variable genau auf die gleiche Weise verwenden, wie wir dies in den letzten Abschnitten kennengelernt haben.

8 Shell-Programme erstellen

Um dies an einem Beispiel darzustellen, gestalten wir nun zunächst ein Programm, das mit einer Textausgabe beginnt, die den Anwender zur Eingabe seines Namens auffordert. Diesen Wert nehmen wir dann über den `read`-Befehl in der Variable `name` auf. Anschließend gestalten wir damit eine personalisierte Begrüßung:

```
1  #!/bin/bash
2
3  echo "Geben Sie Ihren Namen ein: "
4  read name
5  echo "Hallo $name!"
```

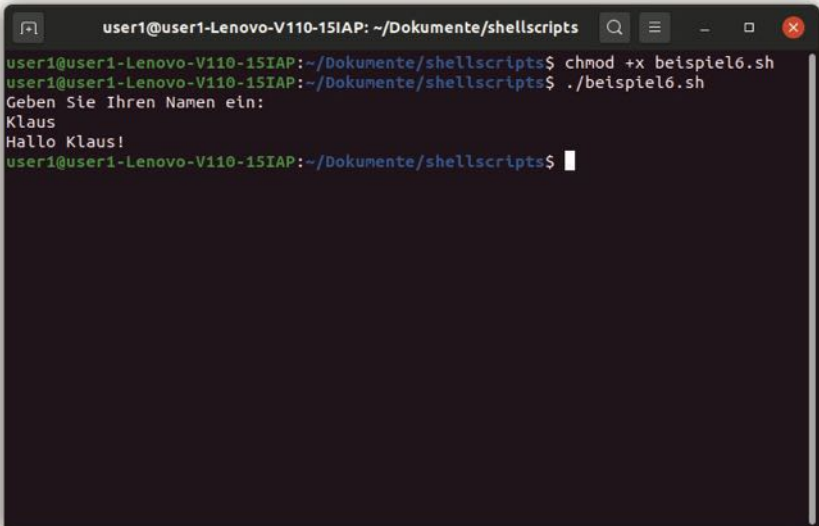


Abb. 8.13 Die Verarbeitung der Eingabe durch den Anwender

Wenn wir hinter dem `read`-Befehl eine Variable angeben, wird der eingegebene Wert automatisch darin gespeichert. Allerdings ist es nicht zwingend notwendig, diese Angabe zu machen. Wir können den `read`-Befehl auch ohne weitere Zusätze verwenden. In diesem Fall speichert das Programm den eingegebenen Wert automatisch in einer Variable mit der Bezeichnung `REPLY`. Auf diese können wir dann im weiteren

Programmverlauf ebenfalls zugreifen. Das folgende Programm weist genau die gleiche Funktionsweise wie das letzte Beispiel auf. Die Ausgabe ist daher vollkommen identisch. Allerdings verwenden wir darin den `read`-Befehl ohne Zusätze:

```
1 #!/bin/bash
2
3 echo "Geben Sie Ihren Namen ein: "
4 read
5 echo "Hallo $REPLY!"
```

8

Diese Vorgehensweise kann jedoch zu Problemen führen. Wenn das Programm mehrere Eingaben enthält, erhält die Variable `REPLY` dabei immer wieder einen neuen Wert. Daher ist es damit nicht möglich, die einzelnen Eingaben parallel zu speichern. Die Variable enthält immer nur den letzten Eingabewert.

8.6 Verzweigungen

Im letzten Abschnitt wurde bereits gesagt, dass es ein wichtiges Merkmal der meisten Computerprogramme darstellt, dass deren Ablauf nicht immer vollkommen gleich ist. In Kapitel 8.5 haben wir Eingaben durch den Anwender aufgenommen und auf diese Weise unterschiedliche Ausgaben erzeugt. Ein weiterer Schritt für einen individuellen Ablauf besteht nun darin, auch die verwendeten Befehle an den Variablenwerten auszurichten. Zu diesem Zweck kommen Verzweigungen zum Einsatz. Diese erlauben es, verschiedene Abläufe vorzugeben und - je nach den Werten bestimmter Variablen - eine dieser Möglichkeiten auszuwählen.

Das if-Statement

Sehr häufig kommt hierfür das `if`-Statement zum Einsatz, das auch als `if`-Abfrage bezeichnet wird. Der englische Ausdruck `if` bedeutet übersetzt wenn oder falls. Daran wird bereits deutlich, dass wir hiermit eine Bedingung aufstellen können. Daraufhin gestalten wir einige Befehle, die nur ausgeführt werden, wenn die Bedingung zutrifft.

8 Shell-Programme erstellen

Das soll nun an einem Beispiel verdeutlicht werden. Dazu gestalten wir ein Programm, das zunächst den Anwender nach seinem Namen fragt. Der Zugriff soll jedoch nur für einen ganz bestimmten Nutzer erlaubt sein. Daher überprüfen wir, ob es sich beim eingegebenen Namen um den autorisierten Nutzer handelt. Trifft dies zu, geben wir eine Begrüßung aus.

Zu diesem Zweck gestalten wir zunächst eine Aufforderung zur Eingabe des Namens und nehmen den Wert in einer Variable auf. Da die hierfür notwendigen Befehle bereits bekannt sind, sind hierfür keine weiteren Erklärungen erforderlich:

```
1 echo "Geben Sie Ihren Namen ein: "  
2 read name
```

Danach beginnen wir mit unserem `if`-Statement. Dieses führen wir mit dem Begriff `if` ein. Danach stellen wir die Bedingung in einer eckigen Klammer auf. In diesem Beispiel wollen wir überprüfen, ob der eingegebene Name mit unserem Anwender, der Tim heißt, übereinstimmt. Für den Vergleich verwenden wir das Gleichheitszeichen:

```
1 if [ $name = "Tim" ]
```

Hierbei ist es wichtig, darauf zu achten, dass nach der öffnenden und vor der schließenden eckigen Klammer jeweils ein Leerzeichen stehen muss. Anschließend folgt der Begriff `then`. Danach können wir die Befehle eingeben, die das Programm ausführen soll, wenn die Bedingung zutrifft. Um diesen Bereich abzuschließen, kommt der Ausdruck `fi` zum Einsatz. Alle Befehle, die darauf folgen, sind dann wieder unabhängig von der aufgestellten Bedingung. Unser erstes `if`-Statement sieht dann wie folgt aus:

```
1 #!/bin/bash  
2  
3 echo "Geben Sie Ihren Namen ein: "  
4 read name  
5 if [ $name = "Tim" ]  
6 then  
7     echo "Herzlich willkommen!"  
8 fi
```

```

user1@user1-Lenovo-V110-15IAP: ~/Dokumente/shellscripts
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$ chmod +x beispiel7.sh
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$ ./beispiel7.sh
Geben Sie Ihren Namen ein:
Tim
Herzlich willkommen!
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$ ./beispiel7.sh
Geben Sie Ihren Namen ein:
Tom
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$

```

Abb. 8.14 Die Ausführung des `echo`-Befehls hängt von der Eingabe des Anwenders ab

Abbildung 8.14 zeigt, dass bei der Eingabe des Namens „Tim“ die Begrüßung erscheint. Gibt der Anwender hingegen einen anderen Namen ein, wird der Befehl nicht ausgegeben.

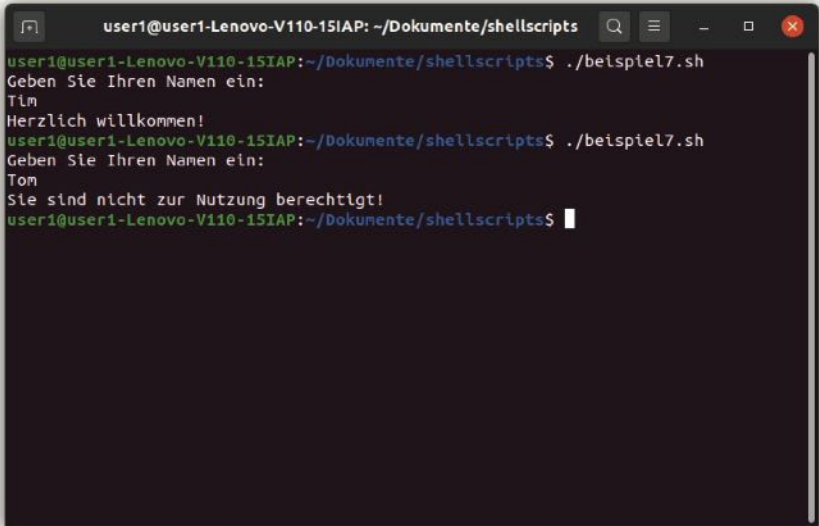
Das `if`-Statement erlaubt es nicht nur, einen Wert auf Gleichheit zu überprüfen. Darüber hinaus ist es möglich, herauszufinden ob eine Ungleichheit vorliegt. In diesem Fall ist die Bedingung erfüllt, wenn die beiden Werte unterschiedlich sind. Diese Funktion nutzen wir nun, um auch eine Meldung auszugeben, falls der Anwender einen Namen eingibt, der nicht mit unserem Nutzer übereinstimmt. In diesem Fall geben wir eine Nachricht aus, dass er das Programm nicht verwenden darf. Um auf Ungleichheit zu prüfen, müssen wir vor dem Gleichheitszeichen ein Ausrufungszeichen setzen:

```

1 if [ $name != "Tim" ]
2 then
3     echo "Sie sind nicht zur Nutzung berechtigt!"

```

Wenn wir unser Programm um diese Befehle erweitern, gibt es sowohl bei der Eingabe des richtigen als auch des falschen Namens eine passende Nachricht aus. Das ist in Abbildung 8.15 zu sehen.



```
user1@user1-Lenovo-V110-15IAP: ~/Dokumente/shellscripts
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$ ./beispiel7.sh
Geben Sie Ihren Namen ein:
Tim
Herzlich willkommen!
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$ ./beispiel7.sh
Geben Sie Ihren Namen ein:
Tom
Sie sind nicht zur Nutzung berechtigt!
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$
```

Abb. 8.15 Nun erscheint auch bei einem falschen Namen eine passende Ausgabe

Es gibt noch einige weitere Operatoren, mit denen wir Vergleiche durchführen können. Um auf Gleichheit zu überprüfen, können wir beispielsweise das doppelte Gleichheitszeichen verwenden. Die Funktionsweise ist dabei identisch zur Verwendung des einfachen Gleichheitszeichens. Darüber hinaus gibt es den Operator `-eq`, mit dem es möglich ist, zu überprüfen, ob zwei Werte gleich sind. Dabei ist es jedoch wichtig zu beachten, dass wir diesen nur auf ganze Zahlen und nicht auf Zeichenketten anwenden können. Das Gleiche gilt für den Operator `-ne`, der auf Ungleichheit prüft.

Darüber hinaus können wir überprüfen, ob ein Zahlenwert größer oder kleiner als ein anderer ist. Folgende Vergleichsoperatoren können wir für Zahlen verwenden:

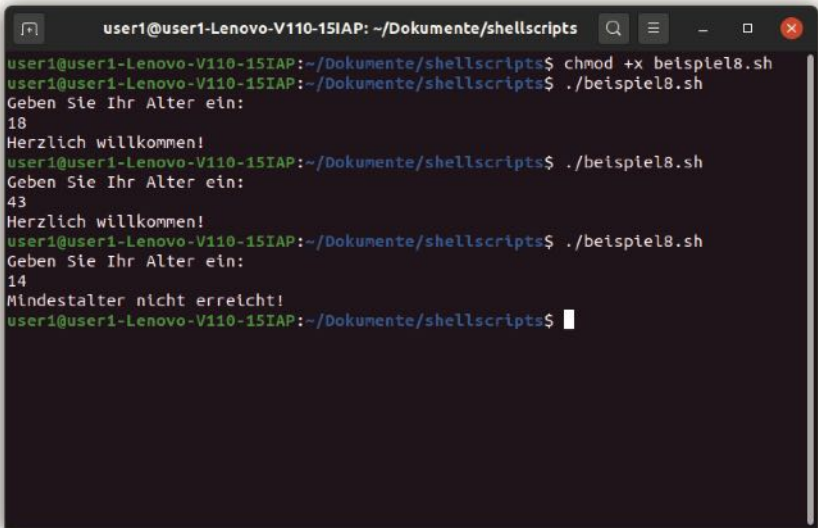
-eq	Trifft zu wenn beide Werte gleich sind.
-ne	Trifft zu, wenn beide Werte ungleich sind.
-gt	Trifft zu, wenn der erste Wert größer als der zweite ist.
-lt	Trifft zu, wenn der erste Wert kleiner als der zweite ist.
-ge	Trifft zu, wenn der erste Wert größer als der zweite oder gleich groß wie dieser ist.
-le	Trifft zu, wenn der erste Wert kleiner als der zweite oder gleich groß wie dieser ist.

Auch hierfür erstellen wir ein kurzes Beispiel. Darin fragen wir einen Anwender nach seinem Namen. Wenn er 18 Jahre oder älter ist, heißen wir ihn willkommen. Ist er hingegen jünger, weisen wir ihn darauf hin, dass er zur Verwendung des Programms nicht berechtigt ist:

```

1  #!/bin/bash
2
3  echo "Geben Sie Ihr Alter ein: "
4  read alter
5  if [ $alter -ge 18 ]
6  then
7      echo "Herzlich willkommen!"
8  fi
9  if [ $alter -lt 18 ]
10 then
11     echo "Mindestalter nicht erreicht!"
12 fi

```

```
user1@user1-Lenovo-V110-15IAP: ~/Dokumente/shellscripts
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$ chmod +x beispiel8.sh
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$ ./beispiel8.sh
Geben Sie Ihr Alter ein:
18
Herzlich willkommen!
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$ ./beispiel8.sh
Geben Sie Ihr Alter ein:
43
Herzlich willkommen!
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$ ./beispiel8.sh
Geben Sie Ihr Alter ein:
14
Mindestalter nicht erreicht!
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$
```

Abb. 8.16 Die Ausgabe bei verschiedenen Altersangaben

Erweiterungen mit else und elif

In unserem Beispiel im letzten Abschnitt haben wir den Anwender nach seinem Namen gefragt. Wenn er den richtigen Namen eingegeben hat, haben wir eine Begrüßung ausgegeben. Wenn er hingegen einen anderen Namen verwendet hat, haben wir ihn darauf hingewiesen, dass die Nutzung nicht möglich ist.

Wenn wir diese beiden Bedingungen genau betrachten, stellen wir fest, dass die erste Bedingung genau das Gegenteil der zweiten Bedingung darstellt. Das hat zur Folge, dass immer genau eine dieser beiden Bedingungen zutrifft – je nach Eingabe die erste oder die zweite. Es ist jedoch nicht möglich, dass beide Bedingungen zutreffen oder dass keine von ihnen erfüllt ist.

Solche Verknüpfungen kommen in sehr vielen Programmen vor. In diesem Beispiel haben wir hierfür den Ungleichheitsoperator verwendet. Damit haben wir genau die gegenteilige Bedingung formuliert. Es gibt hierfür jedoch noch eine einfachere Alternative: das `else`-Statement. Dieses können wir an ein `if`-Statement anhängen. Die Bedingung des `else`-Statements ist immer dann erfüllt, wenn die Bedingung des zugehörigen `if`-Statements nicht zutrifft. Daher müssen wir diese nicht separat aufstellen. Das erleichtert den Arbeitsaufwand. Deshalb gestalten wir das entsprechende Programm nun mit dem `else`-Statement neu. Die Funktionsweise verändert sich dadurch jedoch nicht, sodass die Ausgabe identisch zu Abbildung 8.15 ist.

```
1  #!/bin/bash
2
3  echo "Geben Sie Ihren Namen ein: "
4  read name
5  if [ $name = "Tim" ]
6  then
7      echo "Herzlich willkommen!"
8  else
9      echo "Sie sind nicht zur Nutzung berechtigt!"
10 fi
```

Dieses Beispiel zeigt, dass sich der Code dadurch deutlich verkürzen lässt. Das `else`-Statement steht vor dem Begriff `fi`. Daher müssen wir es auch nicht separat beenden. Mit dem Begriff `fi` schließen wir den kompletten Block ab.

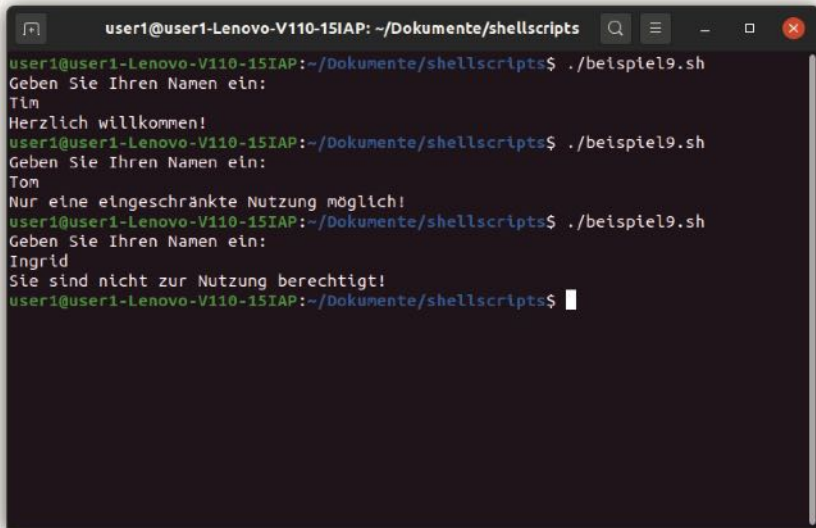
Darüber hinaus kann es vorkommen, dass mehrere Optionen möglich sind. Beispielsweise wäre es möglich, dass der Anwender Tim der Hauptanwender eines Programms ist und nur er über alle Rechte verfügt. Darüber hinaus darf auch der Anwender Tom das Programm nutzen – allerdings nur mit eingeschränkten Rechten. Alle übrigen Anwender werden abgewiesen.

Wenn wir in diesem Fall mit separaten `if`-Statements arbeiten würden, wären hierfür sehr komplizierte Ausdrücke notwendig, da die letzte Option nur angezeigt werden soll, wenn es sich beim Anwender weder um Tim noch um Tom handelt. Das ist zwar möglich, doch sind hierfür weitere

Operatoren notwendig mit denen wir verschiedene Bedingungen verbinden können. Das führt zu einem deutlich erhöhten Programmieraufwand.

Es ist jedoch auch möglich, alle Optionen zusammenzufassen. Dazu erstellen wir zunächst ein gewöhnliches `if`-Statement. Danach fügen wir den Begriff `elif` ein und fügen eine weitere Bedingung hinzu. Auf diese Weise lassen sich beliebig viele Optionen zu einem zusammengehörigen Block zusammenfassen. Abschließend ist es möglich, noch den Begriff `else` zu verwenden. Die zugehörige Aktion wird nur dann ausgeführt, wenn keine der zuvor genannten Bedingungen – sei es im `if`- oder in einem der `elif`-Statements – zutrifft.

```
1  #!/bin/bash
2
3  echo "Geben Sie Ihren Namen ein: "
4  read name
5  if [ $name = "Tim" ]
6  then
7      echo "Herzlich willkommen!"
8  elif [ $name = "Tom" ]
9  then
10     echo "Nur eine eingeschränkte Nutzung möglich!"
11 else
12     echo "Sie sind nicht zur Nutzung berechtigt!"
13 fi
```



```

user1@user1-Lenovo-V110-15IAP: ~/Dokumente/shellscripts
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$ ./beispiel9.sh
Geben Sie Ihren Namen ein:
Tim
Herzlich willkommen!
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$ ./beispiel9.sh
Geben Sie Ihren Namen ein:
Tom
Nur eine eingeschränkte Nutzung möglich!
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$ ./beispiel9.sh
Geben Sie Ihren Namen ein:
Ingrid
Sie sind nicht zur Nutzung berechtigt!
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$

```

Abb. 8.17 Die Ausführung bei verschiedenen Eingaben

Das case-Statement

Eine weitere Möglichkeit, um den Ablauf des Programms zu steuern, stellt das `case`-Statement dar. Dieses bietet keine neuen Funktionen an. Alle Aufgaben, die wir mit dem `case`-Statement erledigen können, lassen sich auch mit `if`, `elif` und `else` umsetzen. Allerdings ist die Schreibweise hierbei häufig kompakter und übersichtlicher.

Umgekehrt ist es jedoch nicht möglich, jedes `if`-Statement durch ein `case`-Statement zu ersetzen. Dieses bietet sich nur dann an, wenn eine klar abgegrenzte Menge an festen Optionen besteht. Es ist hierbei beispielsweise nicht möglich, Größenvergleiche durchzuführen. Wir überprüfen dabei immer nur, ob der Wert einer Variable einem bestimmten Wert entspricht.

Das `case`-Statement kommt beispielsweise häufig für die Programmsteuerung zum Einsatz. Viele Programme bieten mehrere Funktionen

an und der Anwender kann auswählen, welche davon er ausführen will. Das bedeutet, dass eine begrenzte Menge an festen Optionen vorliegt. Das `case`-Statement eignet sich hierfür hervorragend.

Um dies zu zeigen, erstellen wir ein Programm, das dem Anwender vier Auswahloptionen bietet. Um diese auszuwählen, kann er die Werte 1, 2, 3 oder 4 eingeben. In unserem Beispiel implementieren wir die entsprechenden Funktionen selbstverständlich noch nicht. Wir geben lediglich mit dem `case`-Statement aus, welche Funktion ausgewählt wurde.

Hierfür müssen wir zunächst die verschiedenen Optionen präsentieren und anschließend den Anwender dazu auffordern, eine davon auszuwählen. Die hierfür benötigten Befehle sollten bereits bekannt sein, sodass diese hier nicht nochmals erklärt werden müssen.

Danach leiten wir das `case`-Statement mit dem Begriff `case` ein. Anschließend nennen wir die Variable, die überprüft werden soll – in unserem Beispiel die Variable, in der wir die Eingabe des Anwenders aufgenommen haben. Danach folgt der Begriff `in`.

Im nächsten Schritt müssen wir die einzelnen Optionen auflisten. Dabei geben wir zunächst den Wert an, den die Variable für die entsprechende Option aufweisen muss. Danach folgt eine schließende Klammer. Wenn wir beispielsweise eine Option für die Eingabe des Werts 1 aufstellen möchten, müssen wir diese mit `1)` einleiten. Danach können wir die zugehörigen Befehle einfügen. Um deutlich zu machen, dass die Option beendet ist, verwenden wir einen doppelten Strichpunkt. Der komplette Code für die erste Option sieht dann wie folgt aus:

```
1) 1) echo "Sie haben Option 1 ausgewählt."  
2) ;;
```

Auf die gleiche Weise erstellen wir auch die übrigen Optionen. Auch beim `case`-Statement ist es sinnvoll, eine Aktion vorzugeben, die das Programm ausführen soll, wenn keiner der genannten Werte auftritt. Das entspricht dem Ausdruck `else` bei der Verwendung eines `if`-Sta-

tements. Hierfür verwenden wir als Vergleichswert das Sternsymbol. Wie wir bereits aus dem Kapitel über Wildcards wissen, steht dieses für einen beliebigen Ausdruck.

Dabei ist es wichtig, darauf zu achten, dass das `case`-Statement immer nur eine Option ausführt. Sobald es eine Übereinstimmung gefunden hat, führt es die zugehörigen Befehle aus und bricht die Überprüfung daraufhin ab. Das ist wichtig, da das Sternsymbol eigentlich auf alle Eingaben zutrifft. Wenn der Anwender beispielsweise den Wert 1 eingibt, ist sowohl die erste Option als auch die Option mit dem Sternsymbol erfüllt. Da die Ausführung jedoch nach der ersten Übereinstimmung abgebrochen wird, kommt es dennoch zu keinen Dopplungen. Um das komplette Statement abzuschließen, verwenden wir den Ausdruck `esac`. In Abbildung 8.18 ist das Programm bei der Eingabe unterschiedlicher Optionen zu sehen. Der komplette Code dafür sieht wie folgt aus:

```
1  #!/bin/bash
2
3  echo "Welche Funktion wollen Sie auswählen?"
4  echo "Funktion 1"
5  echo "Funktion 2"
6  echo "Funktion 3"
7  echo "Funktion 4"
8  read option
9  case $option in
10     1) echo "Sie haben Option 1 ausgewählt."
11         ;;
12     2) echo "Sie haben Option 2 ausgewählt."
13         ;;
14     3) echo "Sie haben Option 3 ausgewählt."
15         ;;
16     4) echo "Sie haben Option 4 ausgewählt."
17         ;;
18     *) echo "Sie haben einen ungültigen Wert eingegeben."
19         ;;
20 esac
```

```

user1@user1-Lenovo-V110-15IAP: ~/Dokumente/shellscripts
Welche Funktion wollen Sie auswählen?
Funktion 1
Funktion 2
Funktion 3
Funktion 4
3
Sie haben Option 3 ausgewählt.
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$ ./beispiel10.sh
Welche Funktion wollen Sie auswählen?
Funktion 1
Funktion 2
Funktion 3
Funktion 4
4
Sie haben Option 4 ausgewählt.
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$ ./beispiel10.sh
Welche Funktion wollen Sie auswählen?
Funktion 1
Funktion 2
Funktion 3
Funktion 4
12
Sie haben einen ungültigen Wert eingegeben.
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$

```

Abb. 8.18 Das case-Statement eignet sich für die Auswahl aus verschiedenen Optionen

8.7 Schleifen in der Shell ausführen

Ein Computer zeichnet sich dadurch aus, dass er einfache Befehle sehr oft wiederholen kann und dafür nur äußerst wenig Zeit benötigt. Das ist für viele Programme sehr hilfreich. Wenn wir all diese Befehle einzeln in das Programm einfügen, führt das zu einem enormen Aufwand. Daher ist es besser, mit Schleifen zu arbeiten. Diese ermöglichen eine automatische Wiederholung der Befehle. Für unsere Shell Scripts stehen mehrere Schleifen zur Auswahl.

Die while-Schleife

Besonders gebräuchlich ist die `while`-Schleife – nicht nur bei der Erstellung von Shell Scripts, sondern auch in fast allen weiteren Programmiersprachen. Diese ist sehr ähnlich wie ein einfaches `if`-Statement

aufgebaut. Zunächst ist es dabei notwendig, eine Bedingung aufzustellen. Danach folgt ein Bereich mit Befehlen, die an die Bedingung gebunden sind. Lediglich die Schlüsselwörter, die hierbei zum Einsatz kommen, sind unterschiedlich. Statt `if` verwenden wir `while`, statt `then do` und statt mit `fi` beenden wir den Ausdruck mit `done`.

Trotz dieser Ähnlichkeit in der Struktur kommt es hierbei zu einem wichtigen Unterschied in der Funktionsweise. Das `if`-Statement überprüft die Bedingung nur ein einziges Mal. Wenn sie erfüllt ist, führt es den zugehörigen Befehlsblock aus. Danach ist dieser Bereich beendet und das Programm fährt mit den darauf folgenden Befehlen fort. Bei der `while`-Schleife kommt es jedoch nach der Beendigung des Befehlsblocks zu einer erneuten Überprüfung der Bedingung. Wenn diese nach wie vor zutrifft, kommt es auch nochmals zu einer Ausführung der entsprechenden Befehle – so lange, bis die Bedingung nicht mehr zutrifft.

Das zeigen wir nun an einem kleinen Beispiel. Darin geben wir die Zahlen von 1 bis 10 aus. Dafür erstellen wir eine Variable mit der Bezeichnung `i`, die zu Beginn den Wert 1 erhält. In der Schleife geben wir diese aus und erhöhen sie bei jedem Durchgang um 1. Als Bedingung verwenden wir, dass die Variable kleiner als 11 sein muss. Das führt dazu, dass die Schleife abgebrochen wird, sobald sie die Aufgabe erledigt hat. In Abbildung 8.19 ist zu sehen, dass die Schleife nun genau die gewünschten Zahlen ausgibt.

```
1 #!/bin/bash
2
3 i=1
4 while [ $i -lt 11 ]
5 do
6     echo $i
7     i='expr $i + 1'
8 done
```

Anmerkung: Bei der Einführung der Variablen wurde gesagt, dass es hierbei wichtig ist, sprechende Bezeichner zu wählen. In diesem Beispiel verwendeten wir jedoch einen einzelnen Buchstaben: `i`. In diesem speziellen Fall ist dies aber dennoch sinnvoll, da es sich hierbei um die allgemein gängige Bezeichnung für den Zähler in einer Schleife han-

delt. Wenn ein Programmierer diesen Variablennamen sieht, weiß er daher sofort, wozu dieser dient. Sollte ein Programm mehrere Schleifen mit unterschiedlichen Zählern benötigen, ist es üblich, die folgenden Buchstaben des Alphabets zu verwenden: *i*, *j*, *k*...

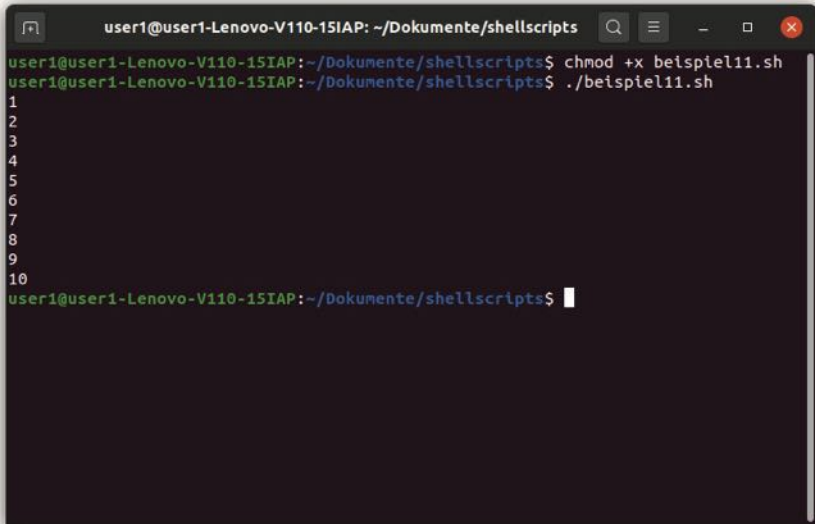
A terminal window titled 'user1@user1-Lenovo-V110-15IAP: ~/Dokumente/shellscripts'. The prompt is 'user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts\$'. The user enters 'chmod +x beispiel11.sh'. The prompt changes to 'user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts\$'. The user enters './beispiel11.sh'. The output shows a list of numbers from 1 to 10, each on a new line. The prompt returns to 'user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts\$'.

Abb. 8.19 Die Ausgabe der Zahlen mit einer Schleife

Die until-Schleife

Eine Schleife, die in kaum einer anderen Programmiersprache zu finden ist und die daher eine große Besonderheit der Shell Scripts darstellt, ist die *until*-Schleife. Diese ist beinahe genau gleich aufgebaut wie die *while*-Schleife. Der einzige strukturelle Unterschied besteht darin, dass statt des Begriffs *while* der Ausdruck *until* zum Einsatz kommt.

Das führt jedoch zu einer etwas anderen Funktionsweise. Während die *while*-Schleife so lange wiederholt wird, wie die Bedingung erfüllt ist,

wird die `until`-Schleife so lange wiederholt, bis die Bedingung erfüllt ist.

Hierbei sei angemerkt, dass der Einsatzbereich der `while`- und der `until`-Schleife genau der gleiche ist. Es ist stets möglich, sie gegenseitig zu ersetzen. Dazu ist es lediglich notwendig, das Gegenteil der Bedingung zu formulieren. In der `while`-Schleife im letzten Abschnitt haben wir beispielsweise die Bedingung aufgestellt, dass die Variable `i` kleiner als 11 sein muss (`[ $i -lt 11 ]`). Das Gegenteil hiervon wäre, dass die Variable `i` größer oder gleich 11 ist: `[ $i -ge 11 ]`. Wenn wir diese Bedingung für eine `until`-Schleife verwenden und den Rest des Programms unverändert lassen, erzielen wir daher genau das gleiche Ergebnis.

Dennoch wollen wir nun für die `until`-Schleife ein eigenes Beispiel entwickeln. Dazu stellen wir dem Anwender eine Rechenaufgabe. Die Eingabe soll so lange wiederholt werden, bis das Ergebnis korrekt ist. Das ist in Abbildung 8.20 zu sehen.

```
1 #!/bin/bash
2
3 ergebnis=0
4 until [ $ergebnis -eq 5 ]
5 do
6     echo "Was ist das Ergebnis aus 2 + 5?"
7     read ergebnis
8 done
```

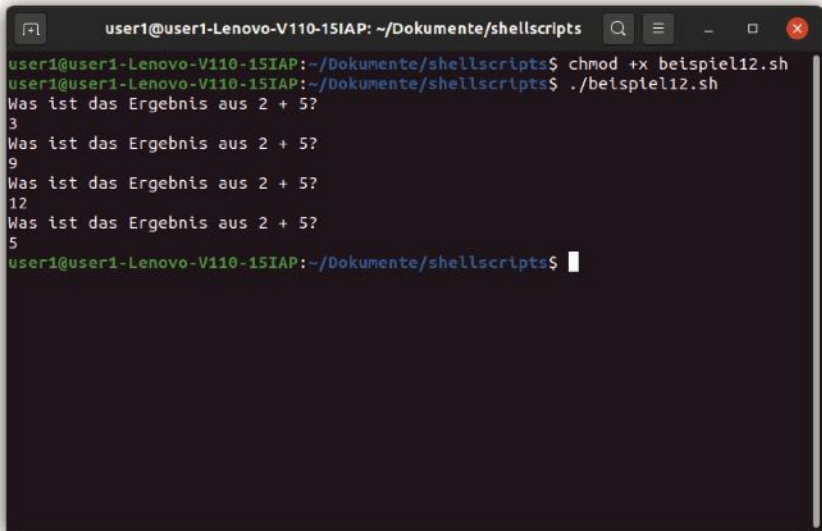
A terminal window titled 'user1@user1-Lenovo-V110-15IAP: ~/Dokumente/shellscripts'. The prompt is 'user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts\$'. The user enters 'chmod +x beispiel12.sh'. The prompt changes to 'user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts\$./beispiel12.sh'. The script then enters a loop, repeatedly asking 'Was ist das Ergebnis aus 2 + 5?' and printing the correct answer (3, 9, 12, 5) after each incorrect input. The terminal shows the following sequence of input and output:
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts\$ chmod +x beispiel12.sh
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts\$./beispiel12.sh
Was ist das Ergebnis aus 2 + 5?
3
Was ist das Ergebnis aus 2 + 5?
9
Was ist das Ergebnis aus 2 + 5?
12
Was ist das Ergebnis aus 2 + 5?
5
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts\$

Abb. 8.20 Die Eingabe wird so lange wiederholt, bis das Ergebnis korrekt ist.

Die for-Schleife

Eine weitere Möglichkeit stellt die `for`-Schleife dar. Diese eignet sich insbesondere für die Arbeit mit Arrays sowie für Schleifen, bei denen eine feste Zahl an Durchläufen erfolgen soll. All diese Aufgaben lassen sich auch mit einer `while`- oder `until`-Schleife durchführen. Allerdings führt die Verwendung der `for`-Schleife meistens zu einem kompakteren Code und bringt weniger Arbeit mit sich. Dabei ist es wichtig, darauf zu achten, dass der umgekehrte Weg nicht immer möglich ist: Nicht jede `while`- oder `until`-Schleife lässt sich durch eine `for`-Schleife ersetzen.

Eine `for`-Schleife leiten wir immer mit dem Begriff `for` ein. Danach müssen wir eine Variable angeben. Daran hängen wir den Ausdruck `in` an. Nun müssen wir eine bestimmte Menge an Werten angeben. Diese können wir beispielsweise direkt einfügen, indem wir sie einzeln nen-

nen. Eine andere Alternative besteht darin, ein Array anzugeben. Darüber hinaus besteht die Möglichkeit, eine Zahlenfolge zu erstellen. Das wird später noch vorgestellt.

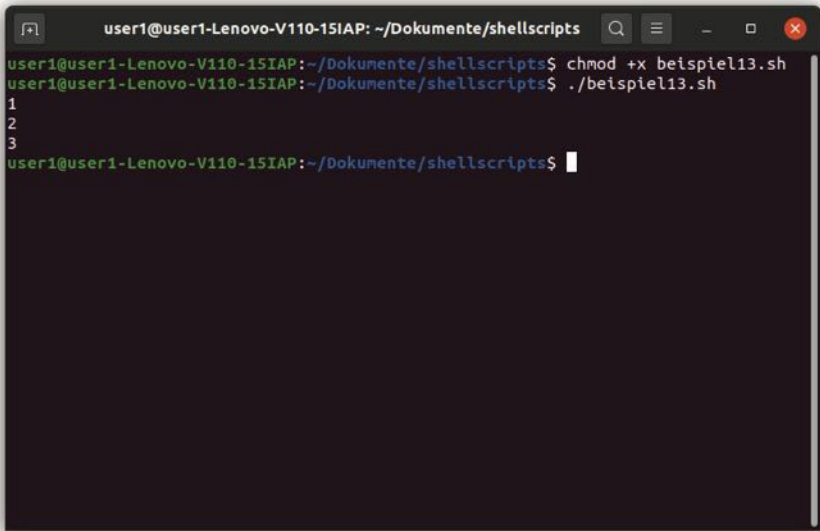
Die Funktionsweise der `for`-Schleife besteht darin, dass diese die nach dem Ausdruck `in` angegebene Menge Schritt für Schritt durchgeht. Die Variable, die wir zuvor definiert haben, nimmt dabei den entsprechenden Wert an. In der Schleife können wir dann darauf zugreifen. Nach dem ersten Durchlauf springt die Schleife dann zum nächsten Wert der Menge über. Diesen weist sie dann erneut der Variable zu. Auf diese Weise lässt sich die komplette Menge durchgehen. Dieser Vorgang wird als Iteration bezeichnet.

Das probieren wir nun mit einem sehr einfachen Beispiel aus. Dieses soll die Zahlen 1 bis 3 ausgeben. Hierfür erstellen wir die Menge auf direktem Wege, indem wir die drei Zahlen jeweils durch ein Leerzeichen voneinander getrennt angeben. Wir führen die Schleife daher wie folgt ein:

```
1 for zahl in 1 2 3
```

Den Körper der Schleife grenzen wir wieder mit den Begriffen `do` und `done` ein. Hier geben wir lediglich die Variable `zahl` aus. In Abbildung 8.21 ist zu sehen, dass diese bei jedem Durchlauf der Schleife ihren Wert verändert.

```
1 #!/bin/bash
2
3 for zahl in 1 2 3
4 do
5     echo $zahl
6 done
```



```

user1@user1-Lenovo-V110-15IAP: ~/Dokumente/shellscripts
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$ chmod +x beispiel13.sh
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$ ./beispiel13.sh
1
2
3
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$

```

Abb. 8.21 Die Ausgabe der `for`-Schleife

Im nächsten Beispiel wenden wir die `for`-Schleife auf ein Array an. Die Funktionsweise ist hierbei genau die gleiche. Die Variable, die wir zu Beginn des Arrays definieren, nimmt bei jedem Durchgang den Wert eines Arrayfelds an. Allerdings müssen wir darauf achten, nach dem Namen des Arrays eine eckige Klammer mit dem Sternsymbol oder dem `@`-Zeichen einzufügen, um auf alle Felder zugreifen zu können. Dieser Ausdruck muss in einer geschweiften Klammer stehen. Für das nächste Beispiel erstellen wir zunächst ein Array mit mehreren Namen. Danach geben wir mit einer `for`-Schleife für jeden von ihnen eine individuelle Begrüßung aus:

```

1  #!/bin/bash
2
3  namen=("Ines" "Michaela" "Christine" "Jörg" "Paul")
4
5  for name in ${namen[*]}
6  do
7      echo "Hallo $name!"
8  done

```

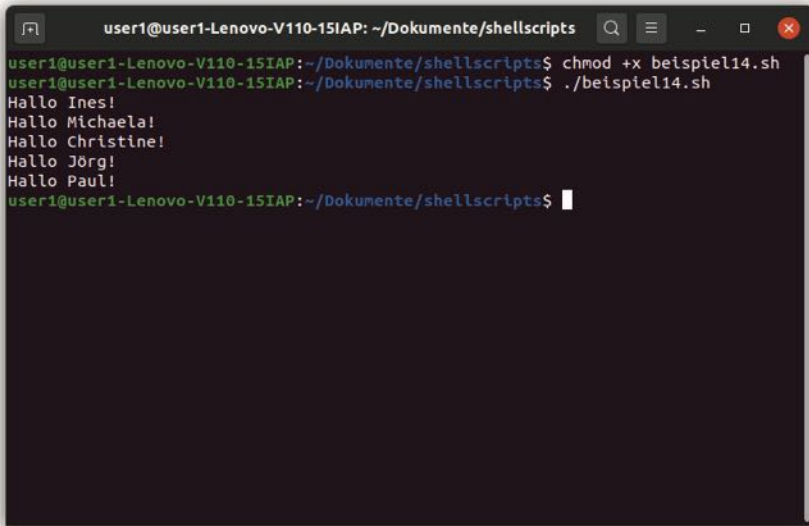
A terminal window titled 'user1@user1-Lenovo-V110-15IAP: ~/Dokumente/shellscripts'. The prompt is 'user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts\$'. The user enters 'chmod +x beispiel14.sh'. The prompt changes to 'user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts\$./beispiel14.sh'. The script outputs five lines: 'Hallo Ines!', 'Hallo Michaela!', 'Hallo Christine!', 'Hallo Jörg!', and 'Hallo Paul!'. The prompt returns to 'user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts\$'.

Abb. 8.22 Die Ausgabe der Inhalte des Arrays

In Abbildung 8.22 ist zu sehen, dass wir auf diese Weise das Array schnell und einfach ausgeben können. Im Vergleich zur Verwendung einer `while`-Schleife ist es hierbei insbesondere vorteilhaft, dass wir hierfür die Array-Länge nicht kennen müssen. Die Anzahl der Durchgänge wird dabei automatisch angepasst.

Wir haben bereits gesagt, dass sich die `for`-Schleife auch bestens für Aufgaben eignet, bei denen eine feste Anzahl an Durchläufen erforderlich ist. In unserem ersten Beispiel haben wir etwa den Inhalt dreimal ausgegeben. Allerdings sind bei vielen Schleifen deutlich mehr Durchläufe notwendig. Wenn wir hierbei alle Zahlen einzeln eingeben, bringt das einen enormen Aufwand mit sich. Doch gibt es auch eine deutlich einfachere Möglichkeit: Wir können eine Zahlenfolge erstellen. Dazu müssen wir lediglich den Ausgangswert, zwei Punkte und den Endwert eingeben. Dieser Ausdruck muss in einer geschweiften Klammer stehen. In unserer ersten `while`-Schleife haben wir beispielsweise die Werte von 1 bis 10 ausgegeben. Diese Aufgabe können wir mit einer

for-Schleife nun noch etwas schneller erledigen, ohne dass sich dabei die Ausgabe ändert:

```
1  #!/bin/bash
2
3  for zahl in {1..10}
4  do
5      echo $zahl
6  done
```

Die Befehle break und continue

Bislang haben wir für die Steuerung der Schleifen lediglich die Bedingung beziehungsweise eine Iteration verwendet. Es gibt aber noch weitere Möglichkeiten: die Befehle `break` und `continue`.

Der Befehl `break` führt dazu, dass die Schleife komplett beendet wird. Normalerweise steht er innerhalb einer `if`-Abfrage, da die Schleife sonst automatisch bereits beim ersten Durchgang beendet würde und daher überflüssig wäre.

Der `break`-Befehl bietet keine wirklich neuen Möglichkeiten für die Umsetzung unserer Programme. Durch die Anpassung der Bedingung und durch passende `if`-Statements kann die gewünschte Funktionsweise immer auch ohne den `break`-Befehl erzielt werden. Allerdings stellt er in vielen Fällen eine erhebliche Erleichterung dar.

Dafür gestalten wir nun ein weiteres Beispiel. Zu diesem Zweck erstellen wir wieder ein Array mit Namen. Darin suchen wir nun jedoch eine bestimmte Person. Das Ziel besteht dabei nicht darin, das komplette Array durchzugehen, sondern darin, herauszufinden, ob die entsprechende Person darin aufgeführt ist. In diesem Fall geben wir eine entsprechende Nachricht aus. Wenn wir den gewünschten Eintrag gefunden haben, ist es nun jedoch nicht mehr notwendig, den Rest des Arrays durchzugehen. Bei unserem kleinen Beispiel spielt dies sicherlich keine große Rolle. Wenn wir aber davon ausgehen, dass es auch Arrays mit Tausenden Einträgen gibt, können wir auf diese Weise die Effizienz unseres Programms steigern. Als Kontrolle für die Funktions-

weise geben wir außerdem eine kurze Nachricht aus, wenn wir beim entsprechenden Durchlauf die Person nicht gefunden haben:

```

1  #!/bin/bash
2
3  namen=("Ines" "Michaela" "Christine" "Jörg" "Paul")
4
5  for name in ${namen[*]}
6  do
7      if [ $name = "Christine" ]
8      then
9          echo "Person gefunden!"
10         break
11     fi
12     echo "Person nicht gefunden!"
13 done

```

8

```

user1@user1-Lenovo-V110-15IAP: ~/Dokumente/shellscripts
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$ chmod +x beispiel16.sh
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$ ./beispiel16.sh
Person nicht gefunden!
Person nicht gefunden!
Person gefunden!
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$

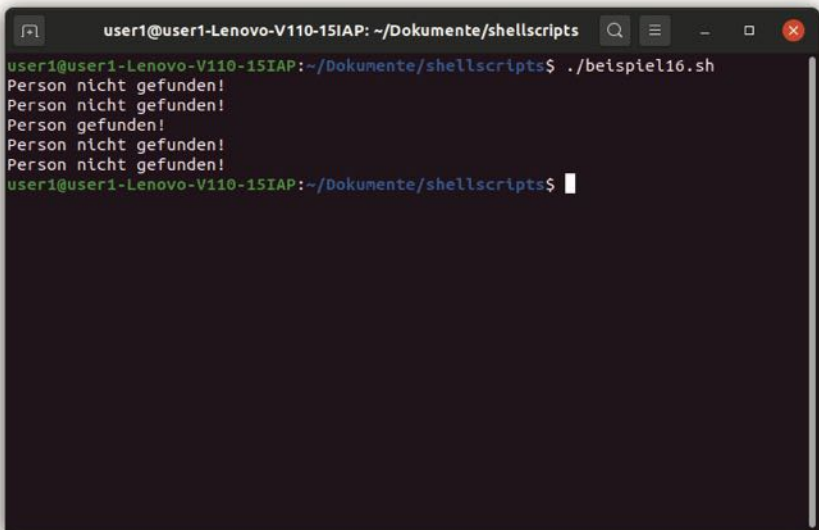
```

Abb. 8.23 Die Unterbrechung der Schleife mit dem `break`-Befehl

Abbildung 8.23 zeigt, dass unsere Schleife nun nur dreimal durchlaufen wird. Danach kommt es zu einem Abbruch, obwohl das Array eigentlich noch zwei weitere Felder aufweist.

Der `continue`-Befehl ermöglicht es ebenfalls, die Schleife zu steuern. Allerdings sorgt er nicht dafür, dass diese komplett abgebrochen wird. Stattdessen unterbricht er nur den aktuellen Durchlauf. Daraufhin springt er zum Beginn der Schleife zurück. Die Bedingung wird dann erneut überprüft. Sollte sie nach wie vor zutreffen, beginnt die Schleife den nächsten Durchgang.

Die Verwendung des `continue`-Befehls ist jedoch recht selten. Das liegt daran, dass sich dieser in der Regel problemlos durch eine Struktur mit `if` und `else` ersetzen lässt, die den meisten Programmierern geläufiger ist. Dennoch soll hierfür noch ein kurzes Beispiel aufgeführt werden. Dabei ersetzen wir den `break`-Befehl in unserem letzten Programm durch den Ausdruck `continue`. Das führt dazu, dass die Schleife das Array stets komplett durchgeht. Bei dem Durchlauf, bei dem eine Übereinstimmung gefunden wird, wird dieser jedoch vorzeitig beendet, sodass die Nachricht, dass die Person nicht gefunden wurde, nicht erscheint. In Abbildung 8.24 ist der Ablauf zu erkennen.

A terminal window titled 'user1@user1-Lenovo-V110-15IAP: ~/Dokumente/shellscrip...' shows the execution of a script. The prompt is 'user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscrip\$' and the command entered is './beispiel6.sh'. The output consists of five lines: 'Person nicht gefunden!', 'Person nicht gefunden!', 'Person gefunden!', 'Person nicht gefunden!', and 'Person nicht gefunden!'. The prompt returns to 'user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscrip\$' after the last line of output.

```
user1@user1-Lenovo-V110-15IAP: ~/Dokumente/shellscrip$ ./beispiel6.sh
Person nicht gefunden!
Person nicht gefunden!
Person gefunden!
Person nicht gefunden!
Person nicht gefunden!
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscrip$
```

Abb. 8.24 Bei der Verwendung des `continue`-Befehls wird das Array komplett durchlaufen

8.8 Funktionen in der Shell verwenden

In zahlreichen Computerprogrammen gibt es Abschnitte, die in genau der gleichen Form mehrere Male vorkommen. Diese jedes Mal neu zu erstellen, würde einen immensen Aufwand mit sich bringen. Aus diesem Grund ist es sinnvoll, mit *Funktionen* zu arbeiten. Darin ist es möglich, verschiedene Befehle anzugeben. Diese lassen sich dann aufrufen, indem Sie lediglich den Funktionsnamen nennen. Das sorgt insbesondere bei längeren Programmen häufig für eine erhebliche Arbeits erleichterung.

8

Darüber hinaus bietet die Verwendung von Funktionen noch zahlreiche weitere Vorteile. Beispielsweise gestaltet sie den Programmcode deutlich übersichtlicher. Damit lassen sich die Befehle zu funktionalen Einheiten zusammenfassen. Da es hierbei möglich ist, den Namen der Funktion frei zu wählen, stellt es kein Problem dar, deutlich zu machen, welche Aufgabe diese jeweils erledigen. Wenn im Programmcode der Name der Funktion erscheint, sieht man auf den ersten Blick, welchen Zweck diese erfüllt, ohne dass es dafür notwendig ist, sich durch die einzelnen Befehle zu arbeiten. Das ist insbesondere für spätere Wartungsarbeiten und Anpassungen sehr vorteilhaft.

Bei der Wartung eines Programms profitieren Sie außerdem davon, dass Sie Anpassungen hierbei nur ein einziges Mal innerhalb der Funktion vornehmen müssen. Sie wirken sich dann auf alle Programmbe reiche aus, die die Funktion aufrufen. Das reduziert den Wartungsauf wand deutlich.

Schließlich erleichtern Funktionen die Zusammenarbeit mehrerer Ent wickler. Wenn bei großen Projekten alle Programmierer am gleichen Code arbeiten, führt das fast immer zu Konflikten. Indem Sie das Pro gramm mithilfe von Funktionen strukturieren, erhält jeder Entwickler einen klar abgegrenzten Arbeitsbereich.

Um von diesen zahlreichen Vorteilen profitieren zu können, lernen Sie in diesem Abschnitt, wie Sie Funktionen erstellen und anwenden.

Funktionen erstellen und aufrufen

Um eine Funktion zu erstellen, ist es zunächst notwendig, einen passenden Namen dafür auszuwählen. Hierfür gelten genau die gleichen Regeln wie für Variablennamen. Es ist lediglich wichtig, darauf zu achten, dass diese ebenfalls einzigartig sein müssen. Das bedeutet, dass es nicht möglich ist, die gleiche Bezeichnung für eine Variable und für eine Funktion zu verwenden.

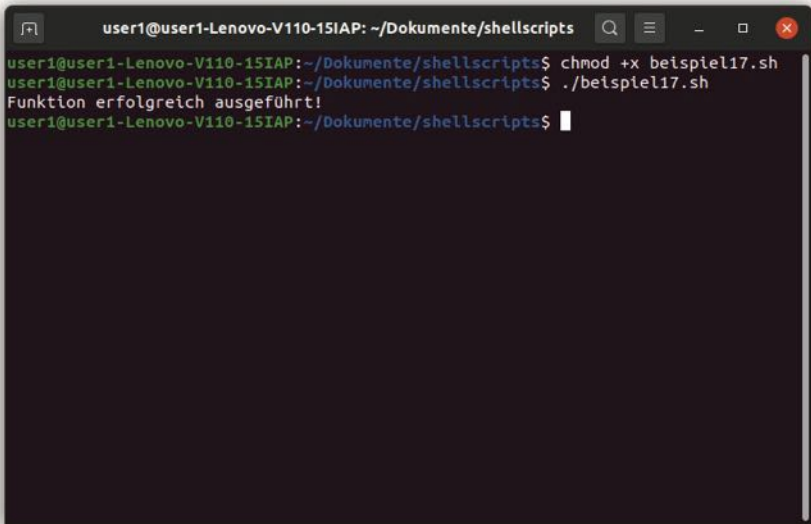
Die Funktion wird vor dem eigentlichen Programm definiert. Hierfür müssen wir zunächst ihren Namen angeben. Danach folgt eine Klammer. Diese lassen wir jedoch leer. Anschließend setzen wir eine geschweifte Klammer. Hier können wir nun alle Befehle einfügen, die die Funktion ausführen soll.

Die erste Funktion, die wir erstellen, soll lediglich einen kurzen Text ausgeben, um den Aufbau zu verdeutlichen. Daher nennen wir sie `text_ausgeben`. Danach setzen wir die leere Klammer und fügen in der geschweiften Klammer dann den Ausgabebefehl ein:

```
1 text_ausgeben () {  
2     echo "Funktion erfolgreich ausgeführt!"  
3 }
```

Nun können wir im eigentlichen Programm auf die Funktion zugreifen. Zu diesem Zweck ist es lediglich notwendig, ihren Namen zu nennen. In Abbildung 8.15 ist zu sehen, dass auf diese Weise der entsprechende Befehl ausgeführt wird. Der komplette Programmcode sieht dann so aus:

```
1 #!/bin/bash  
2  
3 text_ausgeben () {  
4     echo "Funktion erfolgreich ausgeführt!"  
5 }  
6  
7  
8 text_ausgeben
```

A terminal window titled 'user1@user1-Lenovo-V110-15IAP: ~/Dokumente/shellscrip...' with search, menu, and window control icons. The terminal shows the following commands and output:

```
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscrip$ chmod +x beispiel17.sh
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscrip$ ./beispiel17.sh
Funktion erfolgreich ausgeführt!
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscrip$
```

Abb. 8.25 Die Ausgabe der Funktion

Es gibt noch eine alternative Schreibweise für Funktionen. Dabei müssen wir vor dem Funktionsnamen den Begriff `function` angeben. In diesem Fall ist die leere Klammer optional. Die Funktionsweise verändert sich hierbei nicht. Wir können die eben vorgestellte Funktion daher auch auf die folgende Weise definieren:

```
1 function text_ausgeben {
2     echo "Funktion erfolgreich ausgeführt!"
3 }
```

Diese beiden Formen sind vollkommen identisch. Sie können daher selbst entscheiden, welche Sie vorziehen. In diesem Buch bleiben wir jedoch bei der zuerst vorgestellten Schreibweise.

Parameter verwenden

In unserem ersten Beispiel führte unsere Funktion bei jedem Aufruf genau den gleichen Befehl aus. Häufig ist es jedoch auch notwendig,

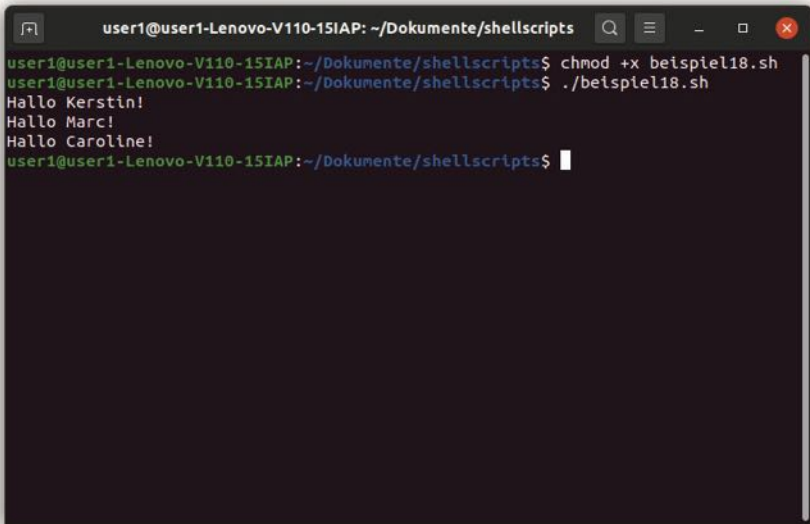
ihren Ablauf an verschiedene Werte des Programms anzupassen. Dazu können wir *Parameter* für die Funktion vorgeben. Diese werden auch als Übergabewerte oder Argumente bezeichnet.

Um dies an einem Beispiel darzustellen, gestalten wir wieder eine Funktion mit einer Textausgabe. Dabei handelt es sich nun jedoch um eine Begrüßungsformel. Diese soll personalisiert sein. Dafür verwenden wir Parameter. Das ermöglicht es, den Namen, der hier erscheinen soll, aus dem Hauptprogramm an die Funktion zu übermitteln.

Hierfür ist es lediglich notwendig, nach dem Funktionsaufruf im Hauptprogramm den Wert anzugeben, den wir übermitteln wollen – in diesem Fall also einen Namen. In der Funktion können wir auf diesen Wert dann zugreifen. Dafür kommt eine spezielle Variable zum Einsatz. Diese erhält den Namen `$1`. Es ist nicht notwendig, dass wir einen Wert für diese vorgeben. Das geschieht automatisch beim Aufruf der Funktion.

Um dies zu demonstrieren, erstellen wir nun die Funktion für die persönliche Begrüßung. Danach rufen wir sie im Hauptprogramm mit unterschiedlichen Namen auf. Das Ergebnis ist in Abbildung 8.26 zu sehen.

```
1  #!/bin/bash
2
3  begruessen () {
4      echo "Hallo $1!"
5  }
6
7
8  begruessen "Kerstin"
9  begruessen "Marc"
10 begruessen "Caroline"
```



```

user1@user1-Lenovo-V110-15IAP: ~/Dokumente/shellscripts
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$ chmod +x beispiel18.sh
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$ ./beispiel18.sh
Hallo Kerstin!
Hallo Marc!
Hallo Caroline!
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$

```

Abb. 8.26 Der Aufruf der Funktion mit unterschiedlichen Parametern

Es besteht auch die Möglichkeit, mehrere Parameter an die Funktion zu übergeben. Dazu müssen wir diese beim Aufruf der Funktion lediglich durch ein Leerzeichen trennen. In der Funktion können wir wieder mit speziellen Werten auf sie zugreifen. Der Wert, der an erster Stelle steht, ist über die Variable `$1` verfügbar. Den Wert, den wir als zweites übergeben haben, erreichen wir über die Variable `$2`. Dabei können wir beliebig viele Werte übergeben, auf die wir dann unter Verwendung ihrer jeweiligen Positionsnummer zugreifen können.

Auch das soll an einem kurzen Beispiel gezeigt werden. Dazu erstellen wir eine Funktion, die zwei Werte addieren soll. Dazu übergeben wir aus dem Hauptprogramm zwei Zahlen. Diese addieren wir dann in der Funktion und geben das Ergebnis aus:

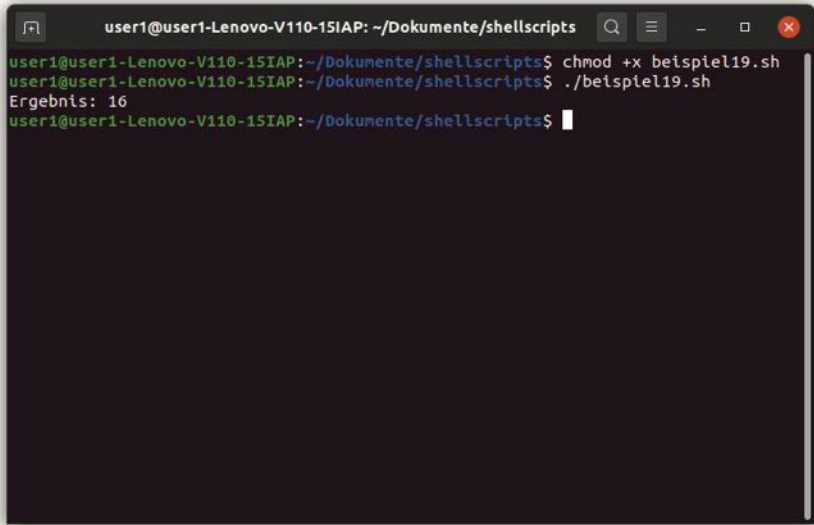
```

1  #!/bin/bash
2
3  addieren () {
4      echo "Ergebnis: 'expr $1 + $2'"

```

8 Shell-Programme erstellen

```
5 }  
6  
7  
8 addieren 7 9
```

A terminal window titled 'user1@user1-Lenovo-V110-15IAP: ~/Dokumente/shellscripts'. The prompt is 'user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts\$'. The user enters 'chmod +x beispiel19.sh'. The prompt changes to 'user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts\$'. The user enters './beispiel19.sh'. The terminal outputs 'Ergebnis: 16'. The prompt returns to 'user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts\$'.

```
user1@user1-Lenovo-V110-15IAP: ~/Dokumente/shellscripts  
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$ chmod +x beispiel19.sh  
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$ ./beispiel19.sh  
Ergebnis: 16  
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$
```

Abb. 8.27 Die Berechnung aus mehreren Parametern

Rückgabewerte

Häufig ist es auch sinnvoll, einen Wert an das Hauptprogramm zurückzugeben. Wenn wir beispielsweise nochmals die Funktion `addieren` betrachten, stellen wir fest, dass diese das Ergebnis nur einmalig ausgibt. Bei vielen Berechnungen ist es jedoch sinnvoll, den entsprechenden Wert zu speichern, damit er auch später noch zur Verfügung steht. Wenn die Funktion jedoch einmal beendet ist, gehen alle darin gespeicherten Werte verloren. Um das Ergebnis dauerhaft verfügbar zu machen, müssen wir es daher im Hauptprogramm aufnehmen. Zu diesem Zweck verwenden wir Rückgabewerte.

Für die Rückgabe eines Werts können wir das `return`-Statement verwenden. Leser, die bereits Erfahrungen mit anderen Programmiersprachen gesammelt haben, kennen dieses sicherlich. Allerdings sei darauf verwiesen, dass die Funktionsweise in einem Shell Script hierbei erhebliche Unterschiede zu den meisten anderen Programmiersprachen aufweist.

Wenn wir mit dem `return`-Statement arbeiten, müssen wir den entsprechenden Begriff ganz an das Ende der Funktion stellen. Das liegt daran, dass er nicht nur einen Wert an das Hauptprogramm übermittelt, sondern auch die Funktion beendet. Nach dem Ausdruck `return` folgt der Wert, den wir zurückgeben möchten:

```
1 #!/bin/bash
2
3 addieren () {
4     return 'expr $1 + $2'
5 }
```

Im Hauptprogramm müssen wir dann zunächst die entsprechende Funktion aufrufen. Der Wert, den wir über das `return`-Statement zurückgeben, wird daraufhin in einer Variable gespeichert, auf die wir mit dem Ausdruck `$?` zugreifen können. In dieser Variable wird stets der Rückgabewert des letzten Befehls gespeichert, den wir ausgeführt haben. Das hat zur Folge, dass sich deren Wert ständig ändert. Um den Rückgabewert dauerhaft verfügbar zu machen, müssen wir ihn daher in einer separaten Variable abspeichern:

```
1 ergebnis=$?
```

Wenn wir dieses Programm ausführen, kommt es genau zur gleichen Darstellung wie in Abbildung 8.27. Allerdings speichern wir dabei den Wert im Hauptprogramm, sodass er auch für weitere Aufgaben zur Verfügung steht. Der komplette Programmcode hierfür sieht wie folgt aus:

```
1 #!/bin/bash
2
3 addieren () {
4     return 'expr $1 + $2'
5 }
6
```


8 Shell-Programme erstellen

```
7
8
9  addieren 7 9
10 ergebnis=$?
11 echo "Ergebnis: $ergebnis"
```

Bei der Verwendung des `return`-Statements tritt jedoch ein Problem auf: Hierbei können wir nur ganze Zahlen zurückgeben. Wenn wir andere Datentypen übermitteln wollen, ist dies nicht möglich.

Um dennoch im Hauptprogramm auf den Wert in einer Funktion zuzugreifen, ist es möglich, sogenannte globale Variablen zu verwenden. Das bedeutet, dass wir zunächst im Hauptprogramm eine Variable erstellen. Auf diese können wir dann auch in der Funktion zugreifen, um sie zu verändern. Diese Veränderung wirkt sich auch auf das Hauptprogramm aus. Grundsätzlich ist die Verwendung globaler Variablen jedoch nicht zu empfehlen. Diese führen bei längeren Programmen mit vielen Funktionen häufig zu unerwünschten Effekten.

Eine weitere Möglichkeit besteht darin, die Ausgabe einer Funktion in einer Variable abzuspeichern. Das ist möglich, indem wir den Aufruf der Funktion in eine Klammer setzen und das Dollarzeichen davorstellen:

```
1 ergebnis=$(addieren 7 9)
```

Das bewirkt, dass die Ausgabebefehle innerhalb der Funktion nicht direkt ausgeführt werden. Stattdessen speichert das Programm sie in der Variable `ergebnis` ab. Das macht es möglich, den gewünschten Wert innerhalb der Funktion per `echo`-Befehl auszugeben, um ihn an das Hauptprogramm zu übermitteln. Dabei müssen wir lediglich beachten, dass hierbei die Ausgaben aller `echo`-Befehle innerhalb der Funktion an die Variable übergeben werden. Direkte Ausgaben innerhalb der Funktion sind daher nicht mehr möglich. Wenn wir diese Vorgehensweise für unser Programm verwenden, sieht der erforderliche Code dafür wie folgt aus:

```
1 #!/bin/bash
2
3 addieren () {
```

```
4     echo 'expr $1 + $2'
5 }
6
7 ergebnis=$(addieren 7 9)
8 echo "Ergebnis: $ergebnis"
```

Auch hierbei ändert sich die Darstellung des Programms nicht. Allerdings wäre es mit dieser Vorgehensweise auch möglich, Zeichenketten an das Hauptprogramm zurückzugeben.

8.9 Übungsaufgaben

8

1. Stellen Sie dem Anwender eine Rechenaufgabe. Überprüfen Sie, ob der eingegebene Wert richtig ist und geben Sie in diesem Fall eine passende Nachricht aus. Falls nicht, teilen Sie dem Anwender mit, ob der eingegebene Wert zu groß oder zu klein war.
2. Erstellen Sie eine Schleife, die es dem Anwender erlaubt, Namen einzugeben. Speichern Sie diese in einem Array ab. Führen Sie die Schleife so lange fort, bis der Anwender den Buchstaben x eingibt. Geben Sie danach das komplette Array aus.
3. Erstellen Sie ein Programm mit einer Funktion, die drei Übergabewerte erhält. Die ersten beiden sollen zwei Werte für eine Berechnung enthalten. Der dritte Wert gibt vor, welche Rechenoperation die Funktion ausführen soll. Bei der Eingabe des Werts 1 soll sie die Zahlen addieren, beim Wert 2 subtrahieren, beim Wert 3 multiplizieren und beim Wert 4 dividieren. Rufen Sie die Funktion dann im Hauptprogramm mit beliebigen Werten auf.

Lösungen:

1.

```
1  #!/bin/bash
2
3  echo "Berechnen Sie das Ergebnis aus 4 + 9"
4  read ergebnis
5
6  if [ $ergebnis -eq 13 ]
7  then
8      echo "Das Ergebnis ist richtig!"
9  elif [ $ergebnis -gt 13 ]
10 then
11     echo "Der eingegebene Wert ist zu groß!"
12 else
13     echo "Der eingegebene Wert ist zu klein!"
14 fi
```

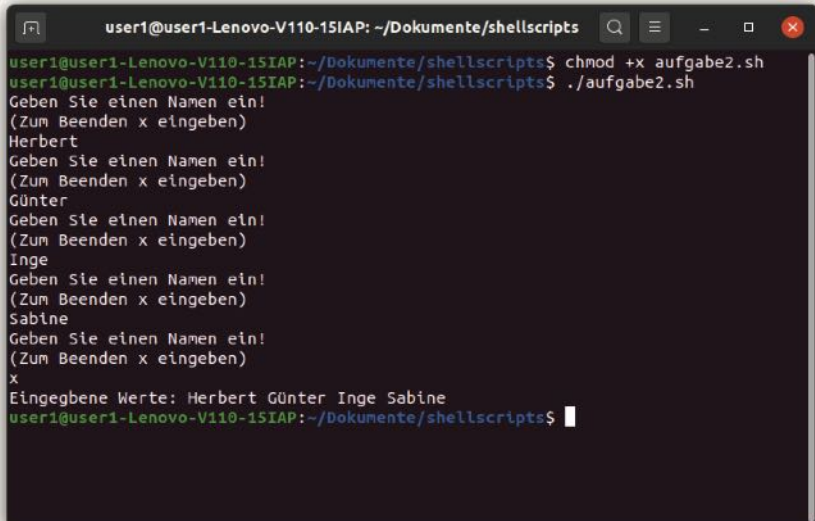
```
user1@user1-Lenovo-V110-15IAP: ~/Dokumente/shellscripts
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$ chmod +x aufgabe1.sh
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$ ./aufgabe1.sh
Berechnen Sie das Ergebnis aus 4 + 9
13
Das Ergebnis ist richtig!
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$ ./aufgabe1.sh
Berechnen Sie das Ergebnis aus 4 + 9
11
Der eingegebene Wert ist zu klein!
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$ ./aufgabe1.sh
Berechnen Sie das Ergebnis aus 4 + 9
18
Der eingegebene Wert ist zu groß!
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$
```

Abb. 8.28 Die Lösung zu Aufgabe 1

2.

```
1  #!/bin/bash
2
3  i=0
4  eingabe="y"
5  until [ $eingabe = "x" ]
6  do
7      echo "Geben Sie einen Namen ein!"
8      echo "(Zum Beenden x eingeben)"
9      read eingabe
10     if [ $eingabe = "x" ]
11     then
12         break
13     fi
14     namen[$i]=$eingabe
15     i='expr $i + 1'
16 done
17 echo "Eingegebene Werte: ${namen[*]}"
```

8

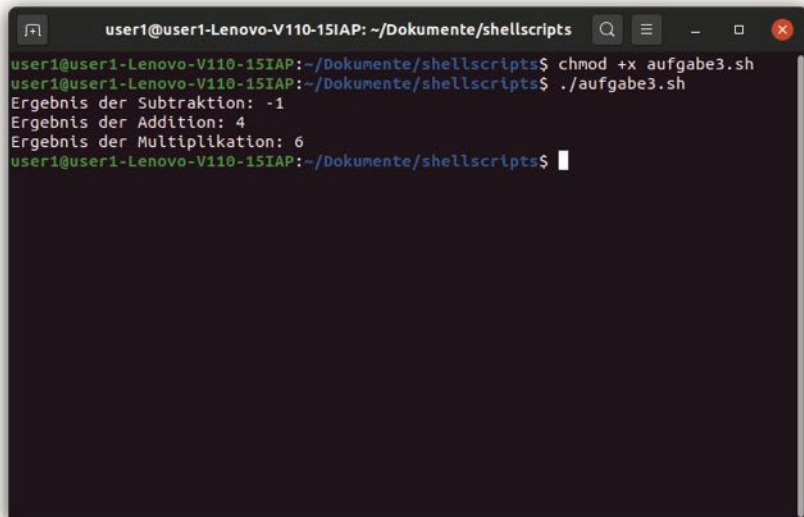


```
user1@user1-Lenovo-V110-15IAP: ~/Dokumente/shellscripts
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$ chmod +x aufgabe2.sh
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$ ./aufgabe2.sh
Geben Sie einen Namen ein!
(Zum Beenden x eingeben)
Herbert
Geben Sie einen Namen ein!
(Zum Beenden x eingeben)
Günter
Geben Sie einen Namen ein!
(Zum Beenden x eingeben)
Inge
Geben Sie einen Namen ein!
(Zum Beenden x eingeben)
Sabine
Geben Sie einen Namen ein!
(Zum Beenden x eingeben)
x
Eingegebene Werte: Herbert Günter Inge Sabine
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$
```

Abb. 8.29 Die Lösung zu Aufgabe 2

3.

```
1  #!/bin/bash
2
3  berechnen () {
4      case $3 in
5          1) echo "Ergebnis der Addition: 'expr $1 + $2'"
6              ;;
7          2) echo "Ergebnis der Subtraktion: 'expr $1 - $2'"
8              ;;
9          3) echo "Ergebnis der Multiplikation: 'expr $1 \* $2'"
10             ;;
11          4) echo "Ergebnis der Division: 'expr $1 / $2'"
12             ;;
13          *) echo "Keine gültige Auswahl!"
14             ;;
15      esac
16  }
17
18
19  berechnen 3 4 2
20  berechnen 2 2 1
21  berechnen 2 3 3
```



```
user1@user1-Lenovo-V110-15IAP: ~/Dokumente/shellscripts
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$ chmod +x aufgabe3.sh
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$ ./aufgabe3.sh
Ergebnis der Subtraktion: -1
Ergebnis der Addition: 4
Ergebnis der Multiplikation: 6
user1@user1-Lenovo-V110-15IAP:~/Dokumente/shellscripts$
```

Abb. 8.30 Die Lösung zu Aufgabe 3

Kapitel 9

Automatischer Aufruf von Programmen mit cron und at

Es gibt zahlreiche Aufgaben, die erhebliche Ressourcen benötigen. Wenn Sie diese während der gewöhnlichen Betriebszeiten ausführen, kann das den Computer über einen längeren Zeitraum hinweg blockieren. Um dieses Problem zu lösen, kann es sinnvoll sein, diese Aufgaben nachts zu erledigen. Auf diese Weise wird Ihre Arbeit an dem Gerät nicht beeinträchtigt. Allerdings haben die meisten Anwender wohl nur wenig Lust, mitten in der Nacht aufzustehen und die entsprechenden Aufgaben auszuführen. Aus diesem Grund kann es sinnvoll sein, hierbei eine automatische Durchführung vorzugeben.

Ein klassisches Beispiel hierfür ist das Erstellen eines Backups. Dieses ist für eine hohe Datensicherheit unverzichtbar – insbesondere wenn es sich um Computer handelt, die in einem Unternehmen zum Einsatz kommen. Wenn Sie das Backup während der gewöhnlichen Arbeitszeiten durchführen, kann dieses die Arbeit der Kollegen stark beeinträchtigen. Deshalb ist es sinnvoll, diese Aufgabe während der Nacht auszuführen. Damit die IT-Administratoren nicht nachts vor Ort sein müssen, ist es empfehlenswert, das Backup jede Nacht automatisch zu starten. Wenn Sie Linux verwenden, ist das hierfür erforderliche Werkzeug bereits integriert. Hierfür können Sie das Programm cron verwenden. Dieses erlaubt es, bestimmte Aufgaben regelmäßig auszuführen – je nach Anforderungen stündlich, täglich, wöchentlich, monatlich oder jährlich.

Darüber hinaus gibt es Aufgaben, bei denen nur eine einmalige Ausführung erforderlich ist. Diese muss jedoch zu einem ganz bestimmten Zeitpunkt erfolgen. Auch hierfür bietet Linux ein passendes Werkzeug an: at. In diesem Kapitel lernen Sie, wie Sie Ihre Aufgaben mit cron und at planen.

9.1 Cron und crontab: Was versteckt sich hinter diesen Begriffen?

Wenn Sie Ihre Aufgaben mit cron planen möchten, kommen Sie mit zwei verschiedenen Begriffen in Berührung: mit cron und crontab. Das sorgt bei vielen Anwendern für etwas Verwirrung. Manche Nutzer sind der Meinung, dass es sich hierbei um synonyme Ausdrücke oder um zwei alternative Techniken handelt. Das trifft jedoch nicht zu. Bei cron und crontab handelt es sich um zwei Elemente, die für die Programmierung der Aufgaben erforderlich sind. Um Verwechslungen zu vermeiden, ist es wichtig, zunächst genau zu klären, was sich dahinter verbirgt.

9

Bei cron handelt es sich um einen sogenannten Daemon. Das ist ein Programm, das beim Starten des Systems automatisch ausgeführt wird. Es bleibt dabei jedoch im Hintergrund, sodass die meisten Anwender es überhaupt nicht bemerken. Dabei bleibt es ständig aktiv und kann auf diese Weise fortwährend bestimmte Aufgaben erledigen.

Die Aufgabe des Daemons cron besteht darin, jede Minute zu überprüfen, ob eine geplante Aufgabe durchzuführen ist. Um dies herauszufinden, überprüft er die Einträge in der Datei crontab – als Abkürzung für cron table. Darin sind alle anstehenden Aufgaben vermerkt. Diese werden als cron jobs bezeichnet.

Bei crontab handelt es sich im Prinzip um eine einfache Textdatei. Diese muss jedoch nach einem ganz bestimmten Muster aufgebaut sein. Wenn der Daemon cron bei der Kontrolle dieser Datei feststellt, dass für den entsprechenden Zeitpunkt eine Aufgabe ansteht, führt er diese aus.

Das zeigt, dass cron das Programm ist, das für die Ausführung der Aufgaben zum vorgegebenen Zeitpunkt zuständig ist. Bei der Datei crontab handelt es sich hingegen um eine Liste, in der die entsprechenden Aufgaben vermerkt sind. Wenn Sie Aufgaben regelmäßig erledigen möchten, müssen Sie hierfür einen Eintrag in der crontab-Datei erstellen. Am Daemon cron müssen Sie hingegen keine Anpassungen vornehmen. Dieser läuft vollkommen selbstständig im Hintergrund ab.

9.2 Aufgaben mit cron regelmäßig ausführen

Bevor wir mit der Verwendung von cron beginnen, müssen wir uns überlegen, welche Aufgabe wir dabei ausführen möchten. Mit unseren bisherigen Kenntnissen wäre es beispielsweise möglich, eine Rechenaufgabe zu erledigen oder eine Schleife zu erzeugen. Da das jedoch keine besonders sinnvollen Aufgaben für eine regelmäßige Ausführung sind, ist es besser, die Befehle für die Steuerung des Systems zu verwenden, von denen wir in Kapitel 7 bereits einige vorgestellt haben. Beispielsweise wäre es auf diese Weise möglich, eine externe Festplatte an den Computer anzuschließen und jede Nacht automatisch einige Dateien zu kopieren und sie auf dieser Festplatte abzuspeichern. Auf diese Weise programmieren Sie Ihr eigenes automatisches Backup.

Zu diesem Zweck benötigen wir einen Befehl, den wir bislang nur am Rande erwähnt haben, der jedoch für diese Aufgabe sehr hilfreich ist: `cp`. Dieser dient in seiner Grundform dazu, Dateien zu kopieren. Wenn Sie ihn mit der Flag `-r` versehen, können Sie damit jedoch auch einen kompletten Ordner mit allen darin vorhandenen Dateien und Unterverzeichnissen kopieren. Das erscheint für unsere Aufgabe optimal. An diesen Befehl müssen Sie dann zunächst das Ausgangsverzeichnis anhängen, das Sie kopieren möchten. Danach folgt das Zielverzeichnis.

Für unser Beispiel erstellen wir täglich eine Sicherungskopie unserer Shell Scripts, die sich im Ordner Dokumente im Unterverzeichnis `shellscripts` befinden. Das führt zu folgendem Pfadnamen: `/home/user1/Dokumente/shellscript`. Wenn Sie hierbei ein anderes Ordnungssystem verwendet haben, können Sie jedoch auch jedes beliebige andere Verzeichnis auswählen. Als Ziel erstellen wir ein neues Verzeichnis mit der Bezeichnung `kopie`. Dieses soll in das Stammverzeichnis der externen Festplatte eingefügt werden. In unserem Beispiel wäre dies `/media/user1`. Der entsprechende `cp`-Befehl sieht daher wie folgt aus:

```
1 cp -r /home/user1/Dokumente/shellscripts /media/user1/kopie
```


In diesem Fall ist es sinnvoll, mit absoluten Pfadnamen zu arbeiten. Auf diese Weise spielt es später keine Rolle, in welchem Verzeichnis sich die entsprechende Datei befindet, die diesen Befehl ausführt.

Mit diesem Befehl können wir zwar eine manuelle Kopie des Ordners erstellen. Das Ziel besteht jedoch darin, eine regelmäßige Sicherungskopie durchzuführen. Dazu ist es zunächst sinnvoll, den angegebenen Befehl in eine separate Shell-Script-Datei zu schreiben. Zwar wäre es in diesem Fall auch möglich, das Kommando direkt in die crontab-Datei einzufügen. Wenn wir jedoch mehrere Befehle gemeinsam ausführen wollen, ist dies problematisch. Daher ist es üblich, ein Shell Script mit allen Befehlen zu erstellen, die automatisch ausgeführt werden sollen. Mit cron können wir diese Datei dann ausführen und damit das Verzeichnis kopieren. Für unser Beispiel enthält das Programm jedoch nur einen einzigen Befehl und ist daher sehr einfach aufgebaut:

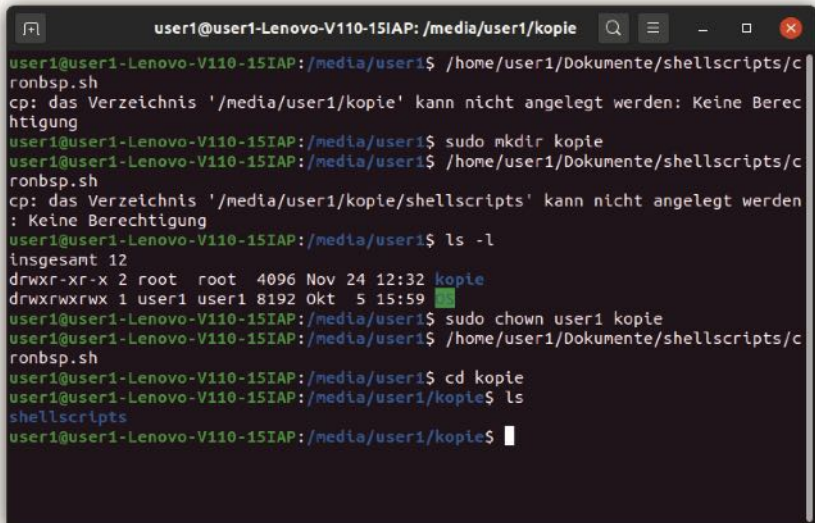
```
1 #!/bin/bash
2
3 cp -r /home/user1/Dokumente/shellscripts /media/user1/kopie
```

Dieses Script können Sie nun einmal in der Shell ausführen. Dabei kommt es jedoch zunächst zu einer Fehlermeldung. Das liegt daran, dass die externe Festplatte dem root-Anwender unterstellt ist. Daher haben wir hier keine Berechtigung. Deshalb ist es sinnvoll, hier den Ordner kopie zunächst manuell mit dem `mkdir`-Befehl zu erstellen. Da hier nur der `root`-User über Schreibrechte verfügt, müssen wir den Begriff `sudo` voranstellen.

Wenn wir das Script nun erneut ausführen, kommt es jedoch immer noch zu einer Fehlermeldung. Das liegt daran, dass auch der neue Ordner dem `root`-Anwender unterstellt ist. Deshalb ist es im nächsten Schritt erforderlich, die Rechte entsprechend anzupassen. Hierfür können wir entweder den Besitzer ändern oder allen Anwendern die Schreibrechte erteilen. Da die erste Alternative zu einer höheren Sicherheit beiträgt, setzen wir diese um: `sudo chown user1 kopie`.

Abbildung 9.1 zeigt, dass es nach diesen Vorbereitungsmaßnahmen nun möglich ist, den entsprechenden Befehl auszuführen. Darüber hi-

naus ist hier zu sehen, dass nun im Ordner kopie ein Unterverzeichnis mit der Bezeichnung shellscripts angelegt wurde. Das entspricht der Bezeichnung unseres Ausgangsverzeichnisses. Darin sind nun auch alle Dateien des ursprünglichen Ordners enthalten.



```

user1@user1-Lenovo-V110-15IAP: /media/user1/kopie
user1@user1-Lenovo-V110-15IAP:/media/user1$ /home/user1/Dokumente/shellscripts/c
ronbsp.sh
cp: das Verzeichnis '/media/user1/kopie' kann nicht angelegt werden: Keine Berec
htigung
user1@user1-Lenovo-V110-15IAP:/media/user1$ sudo mkdir kopie
user1@user1-Lenovo-V110-15IAP:/media/user1$ /home/user1/Dokumente/shellscripts/c
ronbsp.sh
cp: das Verzeichnis '/media/user1/kopie/shellscripts' kann nicht angelegt werden
: Keine Berechtigung
user1@user1-Lenovo-V110-15IAP:/media/user1$ ls -l
insgesamt 12
drwxr-xr-x 2 root root 4096 Nov 24 12:32 kopie
drwxrwxrwx 1 user1 user1 8192 Okt  5 15:59 
user1@user1-Lenovo-V110-15IAP:/media/user1$ sudo chown user1 kopie
user1@user1-Lenovo-V110-15IAP:/media/user1$ /home/user1/Dokumente/shellscripts/c
ronbsp.sh
user1@user1-Lenovo-V110-15IAP:/media/user1$ cd kopie
user1@user1-Lenovo-V110-15IAP:/media/user1/kopie$ ls
shellscripts
user1@user1-Lenovo-V110-15IAP:/media/user1/kopie$

```

Abb. 9.1 Die Anpassung der Rechte für die Kopie des Ordners auf eine externe Festplatte

Unser Script lässt sich nach diesen Vorbereitungsmaßnahmen nun ausführen und erstellt die gewünschte Sicherungskopie. Nun müssen wir es noch in die crontab-Datei eintragen, um für eine regelmäßige Ausführung zu sorgen.

Um diese Datei zu erreichen, ist es notwendig, den Befehl `crontab -e` einzugeben. Bei der ersten Ausführung können Sie dabei einen passenden Editor auswählen. In unserem Beispiel verwenden wir den Nano-Editor. Daraufhin erscheint der Inhalt der Textdatei. Diese enthält bislang lediglich Kommentare. Diese sind mit dem Raute-Zeichen (#) kenntlich gemacht. Wenn dieses Zeichen erscheint, bedeutet dies, dass es sich bei den restlichen Zeichen dieser Zeile um einen Kommentar

handelt, der keine Auswirkungen auf die Ausführung hat. Diese Einträge werden daher bei der Ausführung ignoriert und dienen lediglich dazu, den Code verständlicher zu gestalten. In diesem Beispiel geben uns die Kommentare einen Hinweis darauf, wie wir die Aktionen programmieren können.

Wenn wir hier mit der Pfeiltaste bis ans untere Ende der Datei steuern, erkennen wir die Inhalte, die in Abbildung 9.2 zu sehen sind. Hierbei ist insbesondere die letzte Zeile interessant:

```
1 # m h dom mon dow  command
```

Diese zeigt uns das Schema, das für die Programmierung der Aufgaben zum Einsatz kommt. Der Wert `m` steht dabei für die Minuten, der Wert `h` für Stunden, der Wert `dom` für Tag des Monats, `mon` für Monat und `dow` für Wochentag. Danach steht das Kommando, das ausgeführt werden soll.

The screenshot shows a terminal window titled 'user1@user1-Lenovo-V110-15/AP: ~/Dokumente'. The editor is GNU nano 4.8, editing the file '/tmp/crontab.Xtvtcr/crontab'. The content of the file is as follows:

```
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
# m h dom mon dow  command
```

The bottom of the terminal shows the nano editor's command palette with options like 'Hilfe', 'Speichern', 'Wo ist', 'Ausschneiden', 'Ausrichten', 'Textmarke', 'Beenden', 'Datei öffnen', 'Ersetzen', 'Text einfügen', 'Rechtsch.', and 'Zu Zeile gehen'.

Abb. 9.2 Der Inhalt der crontab-Datei

Bei den Zeitangaben können wir jeweils beliebige Werte eingeben – vorausgesetzt, dass diese zum entsprechenden Schema passen. Bei den Minutenangaben sind beispielsweise Werte zwischen 0 und 59 möglich, bei den Stundenangaben Werte zwischen 0 und 23 und bei den Tagen zwischen 1 und 31. Bei den Wochentagen ist es wichtig, darauf zu achten, dass der Sonntag den Wert 0 erhält. Die übrigen Tage erhalten dann ihrer Reihenfolge entsprechend die Werte zwischen 1 und 6.

Von großer Bedeutung ist hierbei auch das Sternsymbol. Dieses sagt aus, dass die Aktion unabhängig von der entsprechenden Zeitangabe ausgeführt werden soll. Wenn wir beispielsweise an der dritten Stelle – also an den Tagen des Monats – ein Sternsymbol einfügen, bedeutet das, dass die Aktion an jedem Tag des Monats ausgeführt werden soll. Fügen wir es hingegen bei den Stunden ein, wird die Aktion jede Stunde ausgeführt.

Es ist beispielsweise möglich, eine Aktion stündlich durchzuführen. In diesem Fall müssen wir nur eine Angabe für die Minuten eingeben. Wenn wir hierbei beispielsweise den Wert 5 verwenden, bedeutet das, dass die Aktion immer fünf Minuten nach der vollen Stunde ausgeführt wird. Danach folgt fünf Mal das Sternsymbol – jeweils durch ein Leerzeichen getrennt. Das erste davon sagt aus, dass diese Aktion jede Stunde ausgeführt werden soll. Das zweite Sternsymbol macht diese Vorgabe für jeden Tag des Monats, das dritte für jeden Monat des Jahres und das vierte für jeden Wochentag. Zusammengefasst wird diese Aktion daher einmal pro Stunde ausgeführt – unabhängig von Datum und Wochentag.

Eine weitere Möglichkeit besteht darin, eine Aktion für ein ganz bestimmtes Datum auszuführen. Wenn Sie beispielsweise den Befehl jedes Jahr am 13. November um 7:34 Uhr ausführen wollen, kommt dafür der folgende Eintrag zum Einsatz:

```
1 34 7 13 11 *
```

Wenn Sie hingegen das Script jeden Montag um 6 Uhr morgens ausführen lassen möchten, können Sie das folgende Schema verwenden:

```
1 0 6 * * 1
```

Die Möglichkeiten in diesem Bereich sind ausgesprochen vielfältig. Neben den Zahlenangaben sind beispielsweise auch spezielle Schlüsselbegriffe möglich, die ein Schema vorgeben. Diese können an dieser Stelle nicht alle aufgeführt werden. Wenn Sie die weiteren Möglichkeiten kennenlernen möchten, finden Sie diese jedoch beispielsweise in der Ubuntu-Hilfe:

<https://help.ubuntu.com/community/CronHowto>

9

Nachdem dieses Schema verstanden wurde, widmen wir uns wieder unserer eigentlichen Aufgabe: der Durchführung unseres Backups. Dieses wollen wir täglich um 22:30 Uhr durchführen. Als Kommando müssen wir das Script eingeben, das wir zuvor erstellt haben. Das führt zu folgendem Eintrag:

```
1 30 22 * * * /home/user1/Dokumente/shellscripts/cronbsp.sh
```

Diesen Befehl müssen wir nun unten in die Datei eintragen. Es ist wichtig, darauf zu achten, dass in dieser Zeile kein Rautezeichen steht, da der Befehl sonst als Kommentar gilt und ignoriert wird. Nun ist es nur noch notwendig, die Änderungen in der Datei abzuspeichern und den Editor zu schließen. Ab jetzt erstellt der Computer jeden Abend ein automatisches Backup Ihrer Daten – vorausgesetzt, dass er eingeschaltet und die entsprechende externe Festplatte daran angeschlossen ist.

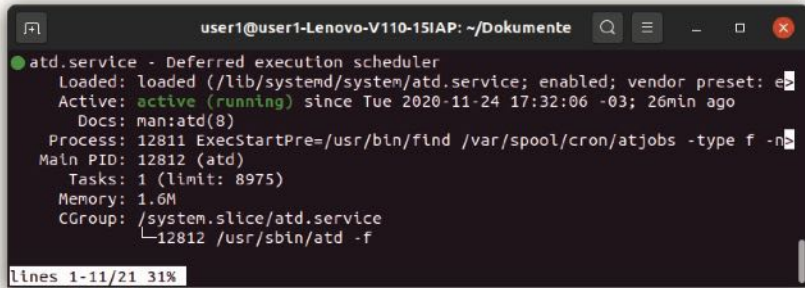
9.3 Das Programm at als Alternative für eine einmalige Ausführung

Die Verwendung von cron bietet sich für regelmäßige Aufgaben an. Es kommt aber auch vor, dass es notwendig ist, eine Aufgabe nur ein einziges Mal zu einem ganz bestimmten Zeitpunkt auszuführen. Hierfür stellt das Programm at das geeignete Mittel dar.

Dieses müssen wir zunächst installieren. Dazu verwenden wir den folgenden Befehl: `sudo apt install at`. Vor der Verwendung dieses Programms ist es sinnvoll, zu überprüfen, ob das System aktiviert ist.

9 Automatischer Aufruf von Programmen mit cron und at

Dazu verwenden wir den Befehl `systemctl status atd`. Das Ergebnis sollte etwa so aussehen, wie in Abbildung 9.3 zu sehen.



```
user1@user1-Lenovo-V110-15IAP: ~/Dokumente
● atd.service - Deferred execution scheduler
   Loaded: loaded (/lib/systemd/system/atd.service; enabled; vendor preset: en
   Active: active (running) since Tue 2020-11-24 17:32:06 -03; 26min ago
     Docs: man:atd(8)
   Process: 12811 ExecStartPre=/usr/bin/find /var/spool/cron/atjobs -type f -n
 Main PID: 12812 (atd)
    Tasks: 1 (limit: 8975)
   Memory: 1.6M
    CGroup: /system.slice/atd.service
            └─12812 /usr/sbin/atd -f

lines 1-11/21 31%
```

Abb. 9.3 Die Status-Abfrage bei aktiviertem at-Daemon

Dabei ist insbesondere der grün dargestellte Begriff „active“ von Bedeutung. Sollte dieser nicht erscheinen, müssen Sie das Programm mit dem folgenden Befehl aktivieren:

```
1 systemctl start atd
```

Für längerfristige Projekte kann es außerdem sinnvoll sein, dafür zu sorgen, dass der Daemon automatisch beim Booten gestartet wird. So steht er auch nach einem Neustart wieder automatisch zur Verfügung. Dazu können Sie den folgenden Befehl verwenden:

```
1 systemctl enable atd
```

Nun können Sie eine neue Aufgabe festlegen. Dazu müssen Sie den Begriff `at` und anschließend den gewünschten Zeitpunkt eingeben. Hierbei stehen verschiedene Formate zur Auswahl. Sie können beispielsweise nur eine Uhrzeit eingeben. Dabei werden die Stunden und die Minuten durch einen Doppelpunkt voneinander getrennt dargestellt. In diesem Fall wird der Befehl ausgeführt, wenn diese Uhrzeit das nächste Mal erreicht ist. Optional ist es danach auch möglich, ein konkretes Datum anzugeben. In diesem Fall wird der Befehl selbstverständlich genau zum angegebenen Zeitpunkt ausgeführt. Dabei können Sie das in Deutschland übliche Datumsformat verwenden, bei dem zunächst

der Tag, dann der Monat und anschließend das Jahr steht und die einzelnen Bereiche durch einen Punkt voneinander getrennt werden. Für eine Aufgabe, die am 24.11.2020 um 18:15 Uhr ausgeführt werden soll, können Sie daher den folgenden Befehl verwenden:

at 18:15 24.11.2020

Hinweis: Hierbei können Sie noch viele weitere Datumsformate verwenden. Wenn Sie einen Überblick über die Möglichkeiten erhalten möchten, können Sie beispielsweise die folgende Seite aufrufen: <http://manpages.ubuntu.com/manpages/trusty/man1/at.1posix.html>

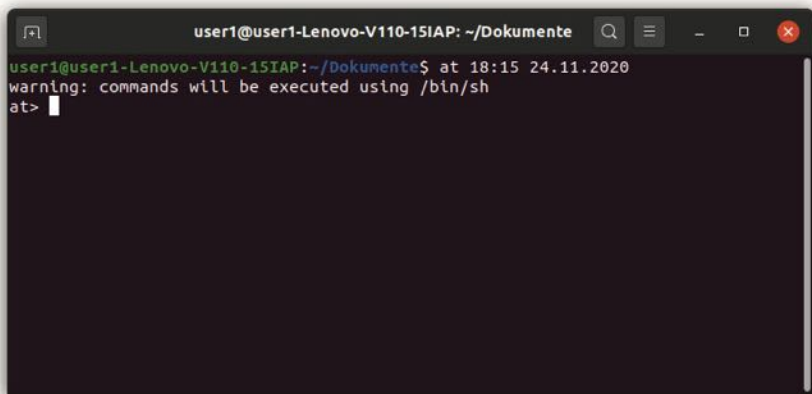


Abb. 9.4 Die Eingabeoberfläche für die Befehle

Nun können Sie einen beliebigen Befehl eingeben, den unser Programm ausführen soll. Beispielsweise können wir wieder unser Script verwenden, das wir in Kapitel 9.3 genutzt haben. Auf diese Weise erstellen wir zum angegebenen Zeitpunkt ein zusätzliches Backup. Doch können Sie auch jeden beliebigen anderen Befehl verwenden. Wenn Sie die Eingabe beendet haben, müssen Sie Strg + D drücken.

9.4 Übungsaufgaben

1. Erstellen Sie einen Eintrag für die `crontab`-Datei, der dafür sorgt, dass am ersten jeden Monats um 5:00 Uhr morgens das Programm `/home/user1/Dokumente/neuerMonat.sh` ausgeführt wird.
2. Geben Sie zunächst einen Befehl an, der es ermöglicht, eine bestimmte Aufgabe am 14. Mai 2024 um 19:23 Uhr zu erledigen. Kopieren Sie zu diesem Zeitpunkt den Inhalt der Datei `/home/user1/Dokumente/bsp1.txt` mit dem `cat`-Befehl in die Datei `/home/user1/Dokumente/bsp2.txt`.

Lösungen:

1.

`0 5 1 * * /home/user1/Dokumente/neuerMonat.sh`

2.

`at 19:23 14.05.2024``cat /home/user1/Dokumente/bsp1.txt > /home/user1/Dokumente/
bsp2.txt`A terminal window titled 'user1@user1-Lenovo-V110-15IAP: ~/Dokumente' shows the execution of an 'at' job. The user enters 'at 19:23 14.05.2024', followed by 'warning: commands will be executed using /bin/sh', then 'at> cat /home/user1/Dokumente/bsp1.txt > /home/user1/Dokumente/bsp2.txt', and finally 'at> <EOT>'. The terminal then displays 'job 8 at Tue May 14 19:23:00 2024' and returns to the prompt 'user1@user1-Lenovo-V110-15IAP:~/Dokumente\$'.**Abb. 9.5** Die Programmierung der Ausführung mit at

Kapitel 10

Netzwerke in Linux erstellen

In vielen Unternehmen kommen zahlreiche Computer zum Einsatz. Auch in Privathaushalten ist dies mittlerweile üblich. In diesen Fällen ist es sinnvoll, die einzelnen Geräte zu einem Netzwerk zusammenzufassen. Das bietet zahlreiche Vorteile. Einer der entscheidenden Aspekte besteht darin, dass es auf diese Weise ganz einfach möglich ist, Daten auszutauschen. Hierfür ist es beispielsweise empfehlenswert, einen speziellen Server oder einen Netzwerkspeicher einzurichten. Darauf können dann alle übrigen Geräte zugreifen. Doch ist es auch möglich, den Datenaustausch ohne eine solche spezielle Speicherinstanz zu ermöglichen. Das Netzwerk lässt sich auch so konfigurieren, dass der direkte Austausch von Daten möglich ist.

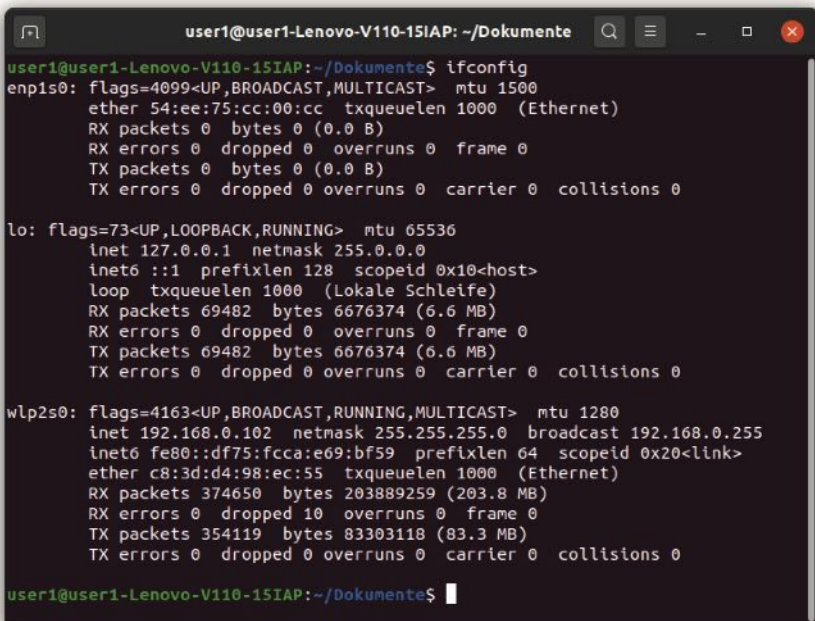
An dieser Stelle wäre es möglich einzuwenden, dass es auch über das Internet möglich ist, die entsprechenden Inhalte auszutauschen. Das trifft zwar zu. Dennoch bietet das Netzwerk zahlreiche Vorteile. Beispielsweise ist die Übertragung hierbei schneller und es wird keine Bandbreite des Internetanschlusses benötigt. Darüber hinaus ist es möglich, weitere Geräte in das Netzwerk zu integrieren und allen Anwendern die Nutzung zu erlauben. Insbesondere bei Druckern stellt dies eine gängige Praxis dar.

Eine wichtige Aufgabe für Linux-Fachkräfte besteht darin, Netzwerke einzurichten und zu warten. Daher wollen wir auch dieses Thema hier kurz ansprechen. Allerdings ist dies ein sehr umfangreiches Gebiet, über das man sicherlich eigene Bücher schreiben könnte. Daher ist es in diesem Kapitel lediglich möglich, die wesentlichen Grundlagen darzustellen. Damit können Sie ein kleines Heimnetzwerk einrichten. Wenn Sie jedoch tatsächlich einmal ein Firmennetzwerk verwalten möchten, sind dafür wesentlich umfangreichere Kenntnisse erforderlich.

10.1 Die Vorbereitungsmaßnahmen für die Einrichtung des Netzwerks

Wenn Sie ein Netzwerk einrichten, ist es zunächst wichtig, die IP-Adressen der einzelnen Geräte festzulegen. Im ersten Schritt bringen wir jedoch erst einmal einige Informationen über unsere Netzwerkanbindung in Erfahrung. Hierfür installieren wir zunächst ein neues Programm: `sudo apt install net-tools`

Nachdem wir dieses Programm installiert haben, können wir den Befehl `ifconfig` ausführen. Daraufhin erscheint eine Darstellung, die ähnlich wie in Abbildung 10.1 aussehen könnte. Die genauen Inhalte können dabei jedoch einige Unterschiede aufweisen.



```

user1@user1-Lenovo-V110-15IAP: ~/Dokumente$ ifconfig
enp1s0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether 54:ee:75:cc:00:cc txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Lokale Schleife)
    RX packets 69482 bytes 6676374 (6.6 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 69482 bytes 6676374 (6.6 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1280
    inet 192.168.0.102 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 fe80::df75:fcca:e09:bf59 prefixlen 64 scopeid 0x20<link>
    ether c8:3d:d4:98:ec:55 txqueuelen 1000 (Ethernet)
    RX packets 374650 bytes 203889259 (203.8 MB)
    RX errors 0 dropped 10 overruns 0 frame 0
    TX packets 354119 bytes 83303118 (83.3 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

user1@user1-Lenovo-V110-15IAP: ~/Dokumente$

```

Abb. 10.1 Die Darstellung unserer Netzwerkverbindungen

In der Regel sind an dieser Stelle zwei oder drei verschiedene Blöcke aufgeführt. Wenn Sie eine verkabelte Verbindung verwenden, sind zwei Blöcke angegeben. Ist wie in diesem Beispiel der WLAN-Empfänger aktiv, wird dieser ebenfalls angezeigt. An erster Stelle steht hierbei stets die verkabelte Ethernetverbindung. In der Abbildung trägt diese den Namen `enp1s0`. Es sind dabei aber auch andere Bezeichnungen möglich. Danach folgt der Begriff `lo`. Hierbei handelt es sich um eine virtuelle Verbindung, die für uns jedoch nicht weiter von Interesse ist. Falls Sie weitere virtuelle Verbindungen installiert haben, ist es auch möglich, dass hier noch weitere Einträge aufgeführt sind. Zum Schluss folgt dann der WLAN-Anschluss. Dieser trägt in der Abbildung den Namen `wlp2s0`. Da das Beispielgerät über WLAN mit dem Router verbunden ist, handelt es sich hierbei um die aktive Verbindung. Das ist unter anderem daran zu sehen, dass hierbei eine IP-Adresse angezeigt wird – in diesem Beispiel `192.168.0.102`.

Für die Einrichtung des Netzwerks ist es insbesondere von Bedeutung, sich die Bezeichner für die einzelnen Anschlüsse zu merken. Wenn Sie das Gerät per Ethernet in das Netzwerk einbinden möchten, müssen Sie sich den Namen merken, der beim ersten Block angegeben ist. Wollen Sie hingegen WLAN verwenden, ist der Bezeichner für den letzten Bereich von Bedeutung. Da diese Begriffe nicht einheitlich sind und sich von Gerät zu Gerät unterscheiden können, ist es wichtig, sich diese zu notieren.

Die IP-Adresse eines Geräts innerhalb eines Netzwerks spielt eine sehr wichtige Rolle. Über diese Zahlenkombination können wir es erreichen. Normalerweise wird sie dynamisch erstellt. Das bedeutet, dass sie sich bei jedem Neustart des Geräts verändern kann. Das ist jedoch für private Netzwerke nicht sehr praktisch. Denn auf diese Weise müssten wir immer erst die aktuelle IP-Adresse in Erfahrung bringen, um Konfigurationen vornehmen zu können. Aus diesem Grund ist es sinnvoll, mit statischen IP-Adressen zu arbeiten. Daher besteht der erste Schritt bei der Einrichtung eines Netzwerks darin, allen Geräten eine statische IP-Adresse zu geben.

Die IP-Adressen haben eigentlich eine internationale Gültigkeit und ermöglichen es, Geräte in aller Welt zu identifizieren. Diese IP-Adressen müssen einmalig sein. Allerdings kommen auch in privaten Netzwerken IP-Adressen zum Einsatz. Diese sind nur innerhalb dieses Netzwerks gültig. Das bedeutet, dass es unzählige Netzwerke geben kann, die die gleichen Adressen verwenden. Damit es dabei zu keinen Überschneidungen mit den international gültigen Adressen kommt, gibt es bestimmte Bereiche, die für lokale Netzwerke freigelassen wurden. Diese können wir für diesen Einsatzzweck nutzen.

In unserem Beispiel arbeiten wir mit den etwas älteren – aber einfacheren – IPv4-Adressen. Zwar kommen auf internationaler Ebene mittlerweile häufig IPv6-Adressen zum Einsatz, da die verfügbaren Adressen im IPv4-Bereich nicht mehr ausreichen. In unserem privaten Netzwerk besteht dieses Problem jedoch nicht, sodass wir hierbei problemlos die einfacheren IPv4-Adressen verwenden können. Diese bestehen aus vier Zahlen zwischen 0 und 255.

Das bedeutet, dass es eine enorme Anzahl an möglichen IP-Adressen gibt. Dabei sind vier Bereiche für lokale Netzwerke vorbehalten:

- ▶ 10.0.0.0 bis 10.255.255.255
- ▶ 172.16.0.0 bis 172.31.255.255
- ▶ 192.168.0.0 bis 192.168.255.255
- ▶ 100.64.0.0 bis 100.127.255.255

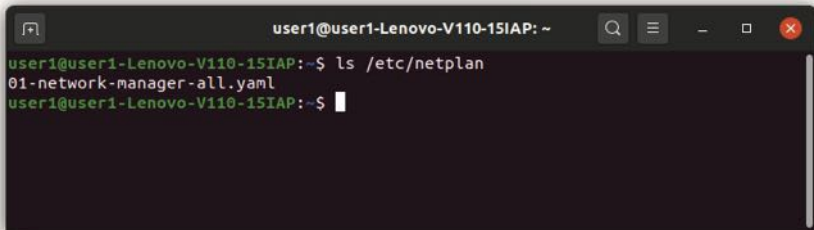
Für private Netzwerke kommt in der Regel der dritte dieser Bereiche zum Einsatz. Wenn wir nochmals Abbildung 10.1 betrachten, wird deutlich, dass auch die dynamisch vergebene IP-Adresse in diesem Bereich liegt. Wenn Sie dies selbst nochmals ausprobieren, kann dabei zwar eine unterschiedliche Adresse erscheinen. Diese sollte aber dennoch im angegebenen Bereich liegen.

Bevor wir Adressen für unsere Geräte auswählen können, müssen wir noch beachten, dass im Netzwerk bereits ein Gerät vorgegeben ist, das über eine feste IP-Adresse verfügt: der Router. Es ist wichtig, dessen Adresse herauszufinden. Dazu können Sie die Betriebsanleitung des Geräts lesen. Wenn Sie diese nicht zur Hand haben, können Sie auch im Internet danach suchen. Die meisten Hersteller verwenden für all ihre Geräte die gleiche Adresse. Daher sollte es kein Problem darstellen, den benötigten Wert herauszufinden. Wenn neben dem Router noch weitere Geräte mit fester IP-Adresse in das Netzwerk integriert werden sollen, ist es notwendig, deren Adresse ebenfalls herauszufinden.

Nun können Sie für die Computer, die Sie in das Netzwerk einbinden möchten, eine beliebige Adresse innerhalb des angegebenen Bereichs auswählen. Dabei müssen Sie zunächst darauf achten, dass diese nicht mit der Adresse eines anderen Geräts im Netzwerk übereinstimmt. Ein weiterer wichtiger Aspekt besteht darin, dass sich der Router und die einzelnen Geräte im gleichen Subnetzwerk befinden sollten. Daher müssen wir darauf achten, dass die vorletzte Zahl dabei identisch ist. Da in unserem Beispiel der Router die Adresse 192.168.0.1 aufweist, wählen wir für den ersten Computer die Adresse 192.168.0.10 aus. Alle weiteren Geräte können wir dann mit fortlaufenden Werten einbinden. So entsteht ein übersichtliches System.

Um die IP-Adressen vorzugeben, verwendet Ubuntu das Werkzeug Netplan – allerdings erst seit der Version 18.04. Wenn Sie eine andere Linux-Distribution oder eine ältere Ubuntu-Version verwenden, ist es möglich, dass dabei ein anderes Werkzeug zum Einsatz kommt. In diesen Fällen ist es nicht möglich, die folgenden Schritte durchzuführen. Doch sollte es über eine kurze Internetrecherche dennoch möglich sein herauszufinden, wie sich die IP-Adressen in den entsprechenden Systemen anpassen lassen.

Netplan verwendet eine Konfigurationsdatei, die sich im Ordner `/etc/netplan` befindet. Da deren Name nicht immer genau gleich ist, lassen wir uns den Inhalt des Ordners zunächst mit dem `ls`-Befehl anzeigen. Das ist in Abbildung 10.2 zu sehen.



```

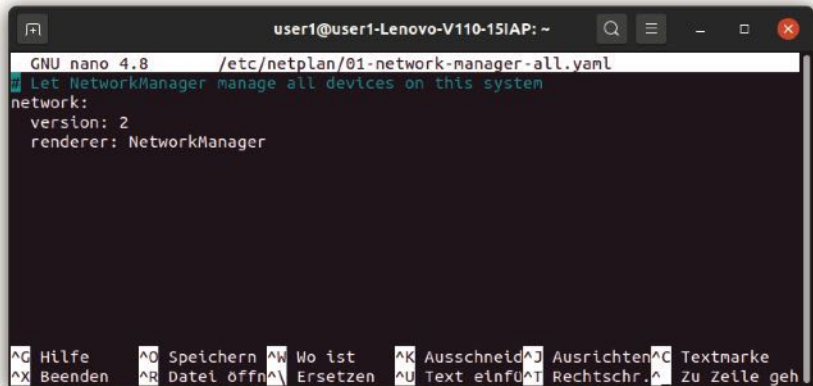
user1@user1-Lenovo-V110-15IAP: ~
user1@user1-Lenovo-V110-15IAP:~$ ls /etc/netplan
01-network-manager-all.yaml
user1@user1-Lenovo-V110-15IAP:~$

```

Abb. 10.2 Der Inhalt des Ordners /etc/netplan

Hier erkennen wir, dass sich in diesem Ordner nur eine Datei befindet – in unserem Beispiel mit der Bezeichnung 01-network-manager-all.yaml. Um die IP-Adressen vorzugeben, müssen wir diese bearbeiten. Zuvor ist es jedoch ratsam, eine Sicherungskopie anzulegen. Sollte die Konfiguration nicht erfolgreich sein, ist es auf diese Weise problemlos möglich, den ursprünglichen Zustand wiederherzustellen. Um mit der Bearbeitung zu beginnen, öffnen wir die Datei mit dem nano-Editor:

```
1 sudo nano /etc/netplan/01-network-manager-all.yaml.
```



```

GNU nano 4.8 /etc/netplan/01-network-manager-all.yaml
# Let NetworkManager manage all devices on this system
network:
  version: 2
  renderer: NetworkManager

```

[^]G Hilfe [^]O Speichern [^]W Wo ist [^]K Ausschneiden [^]J Ausrichten [^]C Textmarke
[^]X Beenden [^]R Datei öffnen [^]E Ersetzen [^]U Text einfügen [^]T Rechtsch. [^]Zu Zelle gehen

Abb. 10.3 Die Darstellung der Datei im nano-Editor

In Abbildung 10.3 ist zu sehen, dass diese Datei bislang nur wenige Inhalte aufweist. Das liegt daran, dass es eigentlich nicht erforderlich ist, hier eigene Vorgaben zu machen. Fehlen die Angaben, kommen für die Verbindung automatisch die Standard-Einstellungen zum Einsatz.

Um statische IP-Adressen zu vergeben, müssen wir den Inhalt dieser Datei jedoch anpassen. Wenn Sie eine verkabelte Ethernet-Verbindung verwenden, sollte der Eintrag wie folgt aussehen:

```
1 network:
2   version: 2
3   renderer: NetworkManager
4   ethernets:
5     enp1s0:
6       dhcp4: no
7       addresses: [192.168.0.10/24]
8       gateway4: 192.168.0.1
9       nameservers:
10        search: [local]
11        addresses: [4.2.2.2, 8.8.8.8]
```

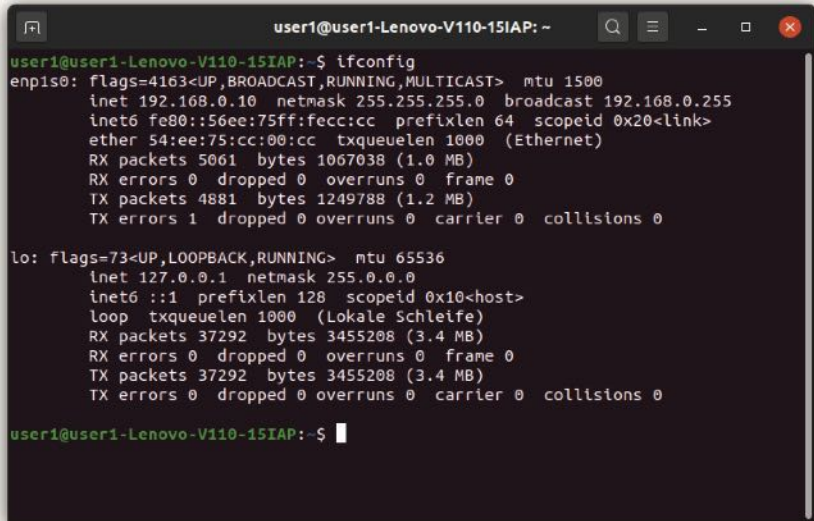
10

Dabei müssen Sie lediglich darauf achten, den Ausdruck `enp1s0` durch den Namen Ihrer Ethernet-Verbindung zu ersetzen. Außerdem ist es notwendig, bei `gateway4` die Adresse Ihres Routers anzugeben. Diese haben wir ja bereits zuvor ermittelt. Die übrigen Werte können Sie unverändert übernehmen. Allerdings sollten Sie dabei die Einrückungen berücksichtigen. Diese sind für die Funktionsweise von Netplan von großer Bedeutung.

Im nächsten Schritt müssen Sie die Datei speichern und den Nano-Editor schließen. Daraufhin ist es notwendig, `sudo netplan apply` auszuführen. Dabei kommt es zu einer kurzen Wartezeit. Dieser Befehl setzt nicht nur die neuen Vorgaben um. Darüber hinaus überprüft er die Datei auf ihre Korrektheit. Wenn sie hierbei einen formellen Fehler gemacht haben, erhalten Sie deshalb einen entsprechenden Hinweis und können diesen korrigieren. Solange bleiben die bisherigen Einstellungen erhalten.

Nun können Sie nochmals den Befehl `ifconfig` ausführen. Vorausgesetzt, dass Sie eine aktive, verkabelte Verbindung verwenden und das

Gerät nicht über das WLAN verbunden ist, sollte nun an der entsprechenden Stelle die von Ihnen vorgegebene IP-Adresse erscheinen. Dies ist in Abbildung 10.4 zu sehen.



```

user1@user1-Lenovo-V110-15IAP: ~$ ifconfig
enp1s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.10 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 fe80::56ee:75ff:fecc:cc prefixlen 64 scopeid 0x20<link>
    ether 54:ee:75:cc:00:cc txqueuelen 1000 (Ethernet)
    RX packets 5061 bytes 1067038 (1.0 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 4881 bytes 1249788 (1.2 MB)
    TX errors 1 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Lokale Schleife)
    RX packets 37292 bytes 3455208 (3.4 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 37292 bytes 3455208 (3.4 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

user1@user1-Lenovo-V110-15IAP:~$

```

Abb. 10.4 Die Verbindung über den neu konfigurierten Ethernet-Anschluss

Wenn Sie eine drahtlose Verbindung verwenden, müssen Sie ebenfalls die Netplan-Datei bearbeiten. Allerdings sind hierbei etwas umfangreichere Einträge notwendig, da Sie auch den Namen des drahtlosen Netzwerks sowie das Passwort eingeben müssen. Davon abgesehen sind die Einträge jedoch recht ähnlich. Eine drahtlose Netzwerkanbindung können Sie beispielsweise wie folgt konfigurieren:

```

1 network:
2     version: 2
3     renderer: NetworkManager
4     wifis:
5         wlp2s0:
6             access-points:
7                 "mein_netz":
8             password: "00000000"
9             dhcp4: no

```

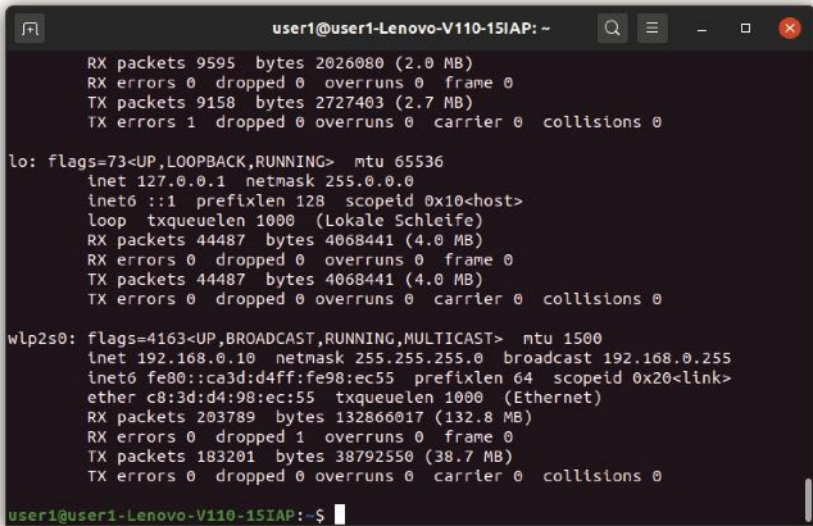
```
10 addresses: [192.168.0.10/24]
11 gateway4: 192.168.0.1
12 nameservers:
13 addresses: [8.8.8.8,4.2.2.2]
```

Dabei müssen Sie zunächst den Namen des WLAN-Empfängers so anpassen, dass dieser mit den Bezeichnungen auf Ihrem Gerät übereinstimmt. Anschließend ist es notwendig, unter dem Eintrag `access-points` den Eintrag „mein_netz“ durch den Namen Ihres Funk-Netzwerks zu ersetzen und anschließend – statt des Werts `oooooooo` – Ihr Passwort einzugeben.

10

Wenn Sie bereits zuvor einen Eintrag für einen verkabelten Netzwerkanschluss eingegeben haben, ist es übrigens nicht notwendig, diesen wieder zu löschen. Die können den Eintrag für das WLAN einfach danach einfügen – allerdings ohne die Einträge für das Netzwerk, die Version und den Renderer zu wiederholen.

Anschließend müssen Sie wieder den Befehl `sudo netplan apply` ausführen. Wenn Sie danach per `ifconfig` die neuen Verbindungen anzeigen lassen, erscheint hierbei jedoch in der Regel noch nicht die vorgegebene IP-Adresse. Das liegt daran, dass der Computer auch weiterhin den bisherigen Anschluss verwendet. Um dies zu ändern, müssen Sie auf das WLAN-Symbol rechts oben auf dem Bildschirm klicken und daraufhin „Einstellungen“ auswählen. Anschließend erscheinen alle sichtbaren Netzwerke. Hier sollte nun auch ein Eintrag auftauchen, der mit `netplan` beginnt und anschließend den Namen des WLAN-Empfängers und auch Ihr Netzwerk enthält. Wählen Sie diesen aus und aktivieren Sie ihn. Daraufhin nutzen Sie die neu erstellte Verbindung. Nun sollte bei Eingabe des `ifconfig`-Befehls die vorgegebene IP-Adresse erscheinen – so wie dies in Abbildung 10.5 der Fall ist.



```

user1@user1-Lenovo-V110-15IAP: ~
RX packets 9595 bytes 2026080 (2.0 MB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 9158 bytes 2727403 (2.7 MB)
TX errors 1 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1000 (Lokale Schleife)
RX packets 44487 bytes 4068441 (4.0 MB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 44487 bytes 4068441 (4.0 MB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.0.10 netmask 255.255.255.0 broadcast 192.168.0.255
inet6 fe80::ca3d:d4ff:fe98:ec55 prefixlen 64 scopeid 0x20<link>
ether c8:3d:d4:98:ec:55 txqueuelen 1000 (Ethernet)
RX packets 203789 bytes 132866017 (132.8 MB)
RX errors 0 dropped 1 overruns 0 frame 0
TX packets 183201 bytes 38792550 (38.7 MB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

user1@user1-Lenovo-V110-15IAP:~$

```

Abb. 10.5 Die neuen Konfigurationen für das WLAN

10.2 Dateien mit anderen Computern teilen

Nach dieser Vorarbeit können wir damit beginnen, das Netzwerk zu nutzen, um Dateien zu teilen. Zu diesem Zweck arbeiten wir nicht mehr mit der Shell. Das liegt daran, dass wir für die Einrichtung des Netzwerks ein Programm benötigen, das auch über eine grafische Benutzeroberfläche verfügt. Diese erleichtert die Anwendung deutlich und stellt daher eine sinnvolle Alternative zur Verwendung der Shell dar. Deshalb verwenden wir für die folgenden Aufgaben die grafische Oberfläche.

Im ersten Schritt erstellen wir einen Ordner, den wir mit den übrigen Anwendern teilen möchten. Wir erstellen ihn innerhalb des Verzeichnisses `Dokumente` und nennen ihn `GemeinsamerOrdner`. Sie können für den Speicherort und die Bezeichnung aber auch beliebige andere Vorgaben machen. Wenn Sie die Funktionsweise später ausprobieren möchten, können Sie darin eine beliebige Datei abspeichern.

Nun klicken wir den Ordner mit der rechten Maustaste an. Im Kontextmenü, das sich daraufhin öffnet, wählen wir „Freigabe im lokalen Netzwerk“ aus.

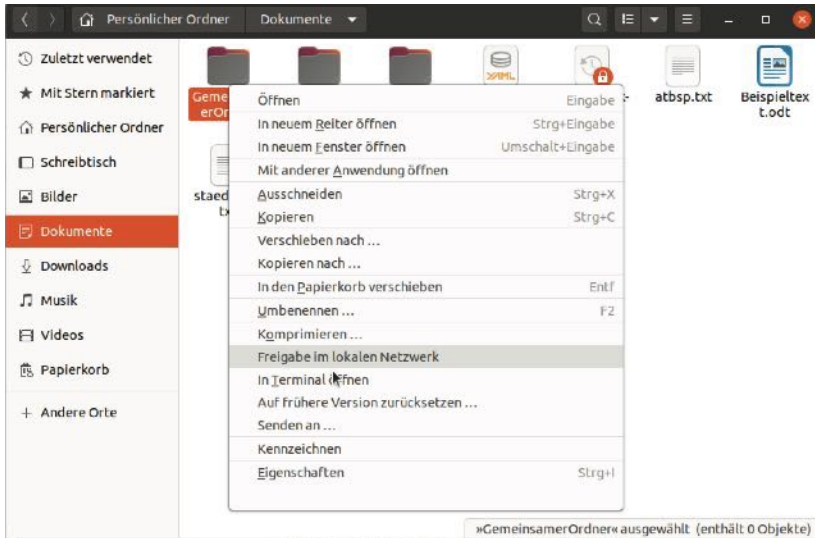


Abb. 10.6 Einen Ordner für das Netzwerk freigeben

Nun öffnet sich ein neues Fenster. Dieses ist in Abbildung 10.7 zu sehen. Hier müssen wir nun die Checkbox beim Eintrag „Diesen Ordner freigeben“ anklicken.

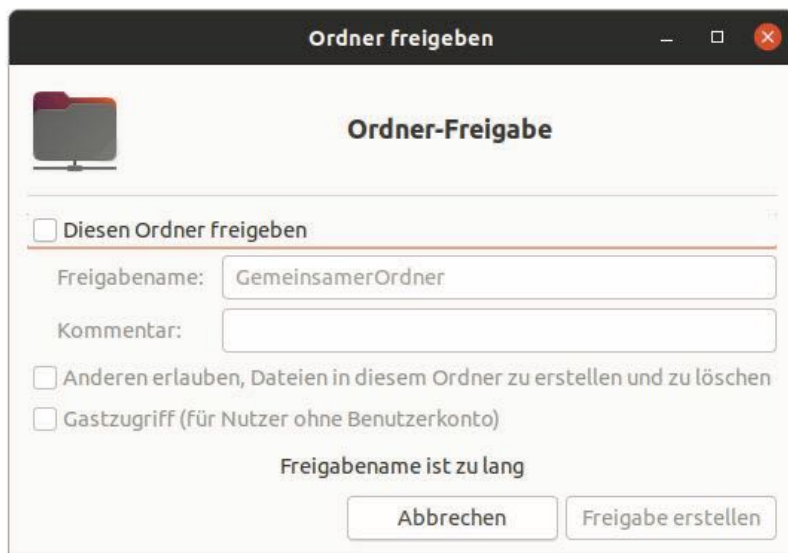


Abb. 10.7 Den Ordner freigeben

Wenn Sie diese Funktion zum ersten Mal aufrufen, erscheint daraufhin in der Regel das Fenster, das in Abbildung 10.8 zu sehen ist. Das liegt daran, dass wir für diese Aufgabe ein neues Programm benötigen, das nicht standardmäßig installiert ist. Dieses trägt den Namen Samba und dient der Organisation unseres Netzwerks. Ubuntu übernimmt die Installation in diesem Fall jedoch automatisch, sodass wir diesen Vorgang lediglich bestätigen müssen.



Abb. 10.8 Den Freigabedienst einrichten

Nachdem der Dienst installiert ist, können wir die entsprechende Checkbox auswählen. Danach müssen wir einen passenden Freigabennamen auswählen, über den der Ordner später im Netzwerk zur Verfügung steht. Hier wird der eigentliche Ordnername vorgeschlagen. Dieser überschreitet jedoch die von Samba vorgegebene Begrenzung der Zeichenzahl, sodass wir ihn etwas kürzen müssen.

Darüber hinaus ist es notwendig, sich Gedanken über die Freigaberechte zu machen.

10

Zum einen müssen Sie sich überlegen, ob es beim Zugriff über das Netzwerk lediglich erlaubt sein soll, die Daten zu lesen oder ob die Anwender sie auch verändern und löschen sowie neue Dateien erstellen dürfen. Wenn Sie dies wünschen, müssen Sie „Anderen erlauben, Dateien in diesem Ordner zu erstellen und zu löschen“ auswählen.

Zum anderen müssen Sie sich entscheiden, ob der Zugriff ohne Authentifizierung erlaubt sein soll oder ob Sie die Nutzung des Ordners nur bei der Eingabe eines Passworts ermöglichen möchten. Wünschen Sie einen freien Zugriff, müssen Sie die Checkbox „Gastzugriff (für Nutzer ohne Benutzerkonto)“ auswählen. Tun Sie dies nicht, ist stets die Eingabe eines Passworts erforderlich. Nach diesen Einstellungen können Sie die Freigabe erstellen.

Wenn Sie einen Gastzugriff erlaubt haben, können Sie sofort zum nächsten Schritt übergehen. Wenn Sie jedoch eine Authentifizierung wünschen, müssen Sie zunächst einen Account anlegen.

Als Nutzer kommen alle Anwender infrage, die Sie bereits unter Linux registriert haben – so wie dies in Kapitel 7.4 vorgestellt wurde. Wenn Sie einem anderen Anwender den Zugriff erlauben möchten, müssen Sie hierfür zunächst einen Account über das System erstellen. Hierbei ist es jedoch sehr wichtig, darauf zu achten, dass nur der Benutzername mit dem im System erstellten Account übereinstimmt. Das Passwort des Systems ist nicht für den Zugriff auf das Netzwerk gültig. Daher müssen Sie ein eigenes Passwort für Samba erstellen.

Dazu müssen wir doch noch einmal das Terminal öffnen. Hier geben wir dann den folgenden Befehl ein: `sudo smbpasswd -a`. Danach folgt der Benutzername des Anwenders, dem wir den Zugriff über das Netzwerk erlauben möchten. Wenn wir beispielsweise `user1` den Zugriff erlauben möchten, sieht der Befehl wie folgt aus: `sudo smbpasswd -a user1`. Danach müssen wir zunächst das `sudo`-Passwort eingeben. Anschließend können wir das neue Samba-Passwort vorgeben. Abbildung 10.9 zeigt diesen Vorgang.



```
user1@user1-Lenovo-V110-15IAP: ~  
user1@user1-Lenovo-V110-15IAP:~$ sudo smbpasswd -a user1  
[sudo] Passwort für user1:  
New SMB password:  
Retype new SMB password:  
user1@user1-Lenovo-V110-15IAP:~$
```

Abb. 10.9 Das Anlegen eines neuen Nutzers für Samba

Nun können wir mit einem anderen Gerät auf den entsprechenden Ordner zugreifen. Es ist aber auch möglich, die Funktionsweise auf dem gleichen Rechner auszuprobieren, wenn Sie nicht über mehrere Computer verfügen.

Zu diesem Zweck ist es notwendig, das Dateiverzeichnis zu öffnen. Klicken Sie danach in der linken Spalte auf „Andere Orte“. Daraufhin erscheint am unteren Bildrand ein Feld mit der Aufschrift „Mit Server verbinden“. In dieses Feld geben Sie nun den Begriff `smb://` ein. Anschließend fügen Sie die IP-Adresse des Geräts ein, auf dem der Ordner abgespeichert ist. Daran müssen Sie dann noch einen Schrägstrich anhängen: `smb://192.168.0.10/`.

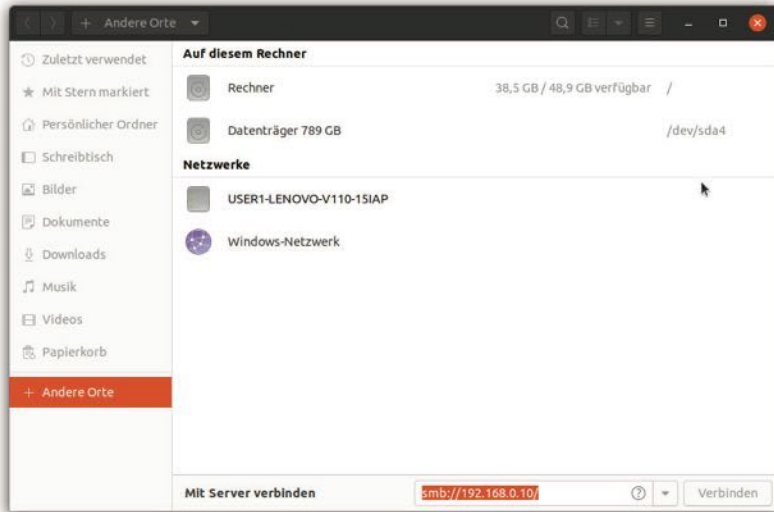


Abb. 10.10 Die Verbindung zum Server

Daran zeigt sich auch, dass es vorteilhaft war, feste IP-Adressen für das Netzwerk zu vergeben. Auf diese Weise können Sie stets mit der gleichen Adresse auf die Ordner auf anderen Geräten zugreifen.

Der Computer, auf dem Sie den Ordner ursprünglich erstellt haben, dient nun als Server. Auf diese Weise können Sie auf die Inhalte zugreifen. Wenn Sie nun auf „Verbinden“ klicken, sehen Sie den Ordner, den wir hierfür angelegt haben. Um diesen zu öffnen, kann es je nach Einstellungen notwendig sein, das soeben erstellte Passwort einzugeben. Daraufhin können Sie auf alle Dateien zugreifen, die sich im Ordner befinden.

Der Zugriff über das Netzwerk ist im Übrigen nicht nur mit Linux-Geräten möglich. Sie können auch problemlos mit einem Windows- oder macOS-Rechner die entsprechenden Dateien nutzen. Bei der genauen Vorgehensweise kann es jedoch zu einigen Unterschieden kommen.

10.3 Einen Print-Server zum Drucken einrichten

Einer der großen Vorteile der Einrichtung eines Netzwerks besteht darin, dass es auf diese Weise möglich ist, einen Drucker auch für andere Geräte freizugeben. Mittlerweile gibt es zwar Netzwerkdrucker, die sich eigenständig in das Netzwerk integrieren lassen. Die Regel besteht aber nach wie vor darin, dass der Drucker an ein einzelnes Gerät angeschlossen ist. Außerdem ist in den meisten Haushalten – und auch in vielen Büros – nur ein einziger Drucker vorhanden. Wenn Sie einen Druckauftrag von einem Gerät aus vergeben möchten, das nicht an einen Drucker angeschlossen ist, würde das ohne das Netzwerk einen erheblichen Aufwand mit sich bringen. Deshalb ist es empfehlenswert, einen sogenannten *Print-Server* einzurichten. Dabei handelt es sich um einen ganz gewöhnlichen Computer, der an einen Drucker angeschlossen ist. Für die Funktion als Print-Server ist es darüber hinaus notwendig, dass er Druckaufträge über das Netzwerk entgegennimmt. Hierfür müssen Sie lediglich einige Anpassungen vornehmen.

Um den Print-Server einzurichten, kommen mehrere Techniken infrage. Besonders häufig kommen hierfür CUPS und Samba zum Einsatz. Die Abkürzung CUPS steht für Common UNIX Printing System. Daraus geht bereits hervor, dass sich dieses für Geräte eignet, die UNIX oder ein UNIX-ähnliches Betriebssystem verwenden. Wenn Ihr Netzwerk ausschließlich aus Linux-Rechnern (oder auch UNIX-Rechnern) besteht, stellt diese Technik eine schnelle und einfache Lösung dar. Wenn Sie jedoch einen Print-Server einrichten möchten, auf den Sie auch mit einem Windows-Rechner zugreifen können, eignet sich diese Technik hingegen nicht. Hierfür ist es besser, das Programm Samba zu verwenden, das wir bereits im letzten Abschnitt kennengelernt haben. In diesem Kapitel stellen wir Ihnen beide Alternativen vor.

Print-Server mit CUPS einrichten

Um einen Print-Server mit CUPS einzurichten, müssen sie zunächst über das Start-Menü die Einstellungen aufrufen. Wählen Sie dann in der Liste am linken Rand den Begriff Drucker aus – so wie dies in Abbildung 10.11 zu sehen ist.

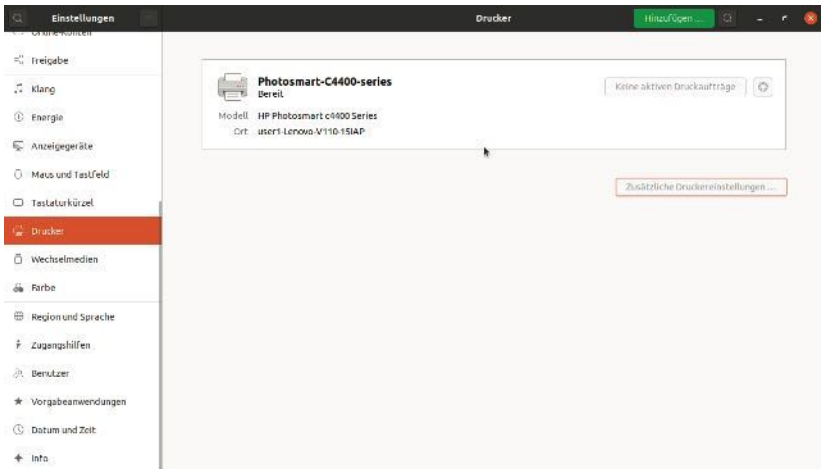


Abb. 10.11 Die Verwaltung der Drucker

Klicken Sie nun auf die Schaltfläche „Zusätzliche Druckereinstellungen“. Daraufhin öffnet sich das Fenster, das Sie in Abbildung 10.12 sehen. Wählen Sie nun in der Menüleiste „Server“ und anschließend auf „Einstellungen“ aus.

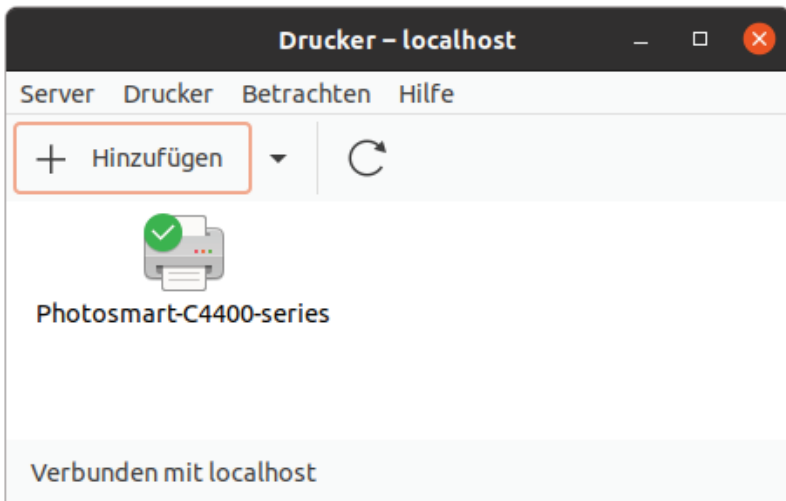


Abb. 10.12 Drucker verwalten

Daraufhin öffnet sich ein neues Fenster. In Abbildung 10.13 erkennen Sie, dass dieses die Option „Alle gemeinsamen Drucker freigeben, die mit diesem System verbunden sind“ enthält. Klicken Sie diese an und bestätigen Sie die Auswahl durch einen Klick auf die Schaltfläche „OK“.

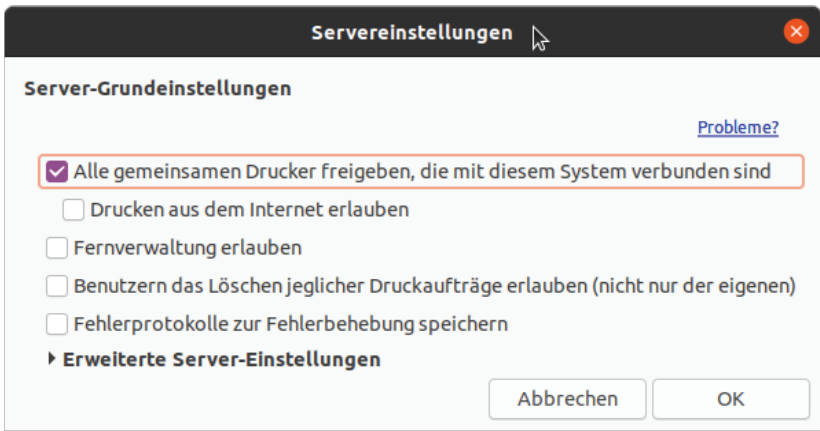


Abb. 10.13 Den Drucker freigeben

Damit können bereits andere Anwender auf den Drucker zugreifen. Darüber hinaus besteht jedoch die Möglichkeit festzulegen, welche Benutzer den Drucker verwenden dürfen und welche nicht. Dazu müssen Sie den installierten Drucker mit der rechten Maustaste anklicken und im Kontextmenü dann „Eigenschaften“ auswählen. Daraufhin öffnet sich das Fenster, das in Abbildung 10.14 zu sehen ist. Hier müssen Sie in der Liste am linken Rand den Begriff „Zugriffskontrolle“ auswählen. Daraufhin haben Sie zwei Optionen. Sie können das Drucken zum einen grundsätzlich allen Netzwerkteilnehmern erlauben. Dabei ist es jedoch auch möglich, Ausnahmen vorzugeben. Zum anderen besteht die Option, grundsätzlich alle Aufträge abzulehnen. Um dennoch das Drucken über das Netzwerk zu erlauben, müssen Sie in diesem Fall genau die Anwender vorgeben, die eine Berechtigung erhalten sollen.

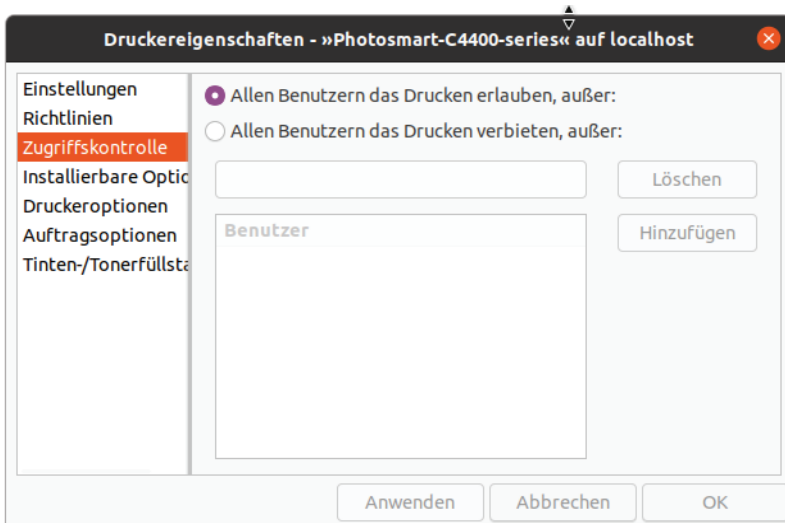


Abb. 10.14 Die Zugriffsrechte vorgeben

Print-Server mit Samba einrichten

Wenn Sie den Zugriff auf den Print-Server auch Geräten erlauben möchten, die Windows oder ein anderes Betriebssystem nutzen, funktioniert die oben beschriebene Methode nicht. In diesem Fall ist es sinnvoll, Samba zu verwenden. Wir gehen davon aus, dass dieses Programm entsprechend der Anleitung in Kapitel 10.2 installiert wurde und daher bereits zur Verfügung steht. In diesem Fall sollte eine Konfigurationsdatei unter der Adresse `/etc/samba/smb.conf` vorhanden sein. Diese müssen Sie nun mit einem Texteditor öffnen – entweder über die Dateiverwaltung oder mit dem Nano-Editor über das Terminal.

Hierbei handelt es sich um eine etwas längere Textdatei. Hier müssen Sie nun weit nach unten scrollen, bis Sie kurz vor dem Ende die Überschrift `[printers]` entdecken. Darin sind mehrere Einträge vorhanden. Zwei von diesen müssen Sie abändern: `browseable` und `guest ok`. Bei beiden ist bislang der Eintrag `no` vorhanden. Ändern Sie diesen zu `yes`:

```
1 browseable = yes
2 guest ok = yes
```



Abb. 10.15 Die Änderungen an der Samba-Konfigurationsdatei

Nachdem Sie die Änderungen so wie in Abbildung 10.15 zu sehen vorgenommen haben, müssen Sie die Datei speichern. Damit die neue Konfiguration umgesetzt wird, müssen Sie Samba neu starten. Das geschieht automatisch beim nächsten Neustart des Systems. Wenn Sie darauf nicht warten möchten, können Sie hierfür die Befehle `sudo service smbd restart` und anschließend `sudo service nmbd restart` über das Terminal ausführen.

Auf einen Print-Server zugreifen

Nachdem Sie die entsprechenden Einstellungen vorgenommen haben, können Sie mit einem anderen Gerät auf den Print-Server zugreifen. Wenn dieses ebenfalls das Betriebssystem Linux verwendet, müssen Sie hierfür nochmals die Einstellungen aufrufen und anschließend „Drucker“ auswählen. Es ist möglich, dass hierbei der neue Print-Server

bereits automatisch angezeigt wird. Ist dies nicht der Fall, müssen Sie rechts oben auf „Hinzufügen“ klicken. Daraufhin öffnet sich ein neues Fenster. Auch hierbei läuft zunächst eine automatische Suche ab, bei der es wahrscheinlich ist, dass der Print-Server angezeigt wird. Sollte er nicht erscheinen, können Sie unten im Suchfeld auch die IP-Adresse des Geräts eingeben. Daraufhin sollte der Drucker schließlich erscheinen. Indem Sie auf „Hinzufügen“ klicken, machen Sie ihn für das Gerät verfügbar. Daraufhin können Sie ihn für alle Druckaufträge verwenden. Abbildung 10.16 zeigt dieses Fenster.

10



Abb. 10.16 Einen Netzwerkdrucker hinzufügen

Wenn Sie für die Freigabe des Druckers Samba verwendet haben, können Sie auch mit Geräten mit anderen Betriebssystemen auf den Drucker zugreifen. Unter Windows müssen Sie beispielsweise die Druckereinstellungen aufrufen und daraufhin einen Netzwerkdrucker hinzufügen. In diesem Fall müssen Sie die IP-Adresse des Geräts und nach einem Schrägstrich den Namen des Druckers eingeben, um Ihren Print-Server zu finden.

10.4 Übungsaufgaben

Am Ende von Kapitel 10.2 wurde gesagt, dass es auch möglich ist, mit Windows-Geräten auf die freigegebenen Dateien zuzugreifen. Dabei wurde die Vorgehensweise jedoch nicht dargestellt. Recherchieren Sie im Internet, wie Sie dabei vorgehen müssen und stellen Sie dies stichwortartig dar.

Lösung:

- ▶ Öffnen Sie den Windows-Explorer.
- ▶ Klicken Sie mit der rechten Maustaste auf „Dieser PC“.
- ▶ Wählen Sie im Kontextmenü „Netzwerkadresse hinzufügen“ aus.
- ▶ Bestätigen Sie das nächste Fenster und klicken Sie dann auf „Eine benutzerdefinierte Netzwerkadresse auswählen“.
- ▶ Geben Sie die IP-Adresse des Geräts, auf dem sich der geteilte Ordner befindet und nach einem Schrägstrich den Freigabenamen ein.
- ▶ Wählen Sie einen Namen für den Server aus und bestätigen Sie die Eingaben.

Kapitel 11

Ubuntu aktualisieren

Die Linux-Distribution Ubuntu zeichnet sich durch regelmäßige Aktualisierungen aus. In der Regel erscheinen zweimal pro Jahr neue Versionen. Diese tragen als Versionsnummer zum einen die Jahreszahl und zum anderen die Nummer des Monats, in dem sie veröffentlicht wurden. Da das Ubuntu-Entwicklerteam bereits seit mehr als einem Jahrzehnt die neuen Versionen im April und im Oktober veröffentlicht, haben sich die Versionsnummern 04 und 10 eingebürgert.

In diesem Buch haben wir bisher mit der Version 20.04 gearbeitet. Während der Arbeiten an den Texten ist jedoch bereits die Version 20.10 erschienen. Bis das Werk im Buchhandel erhältlich ist, wird wahrscheinlich bereits die nächste Version verfügbar sein. Da die Unterschiede dabei in der Regel nicht allzu groß sind, sollten die dargestellten Techniken aber auch problemlos auf die neueren Versionen anwendbar sein.

Durch den halbjährlichen Rhythmus haben Sie als Anwender jedoch immer wieder die Möglichkeit, Ihre Linux-Distribution zu aktualisieren. Im letzten Kapitel stellen wir Ihnen vor, wie Sie dabei vorgehen können. Das gibt Ihnen die Möglichkeit, Ihr System stets auf dem aktuellen Stand zu halten.

Allerdings ist es vor diesem Schritt sinnvoll, sich genau zu überlegen, ob die Aktualisierung für Sie sinnvoll ist. Insbesondere wenn es sich wie beim hier installierten Beispiel um eine Version mit *Long-term Support* handelt. Diese sind durch den Zusatz LTS ausgezeichnet. Dabei beträgt der Support fünf Jahre. Bei gewöhnlichen Versionen hingegen erhalten Sie nur über neun Monate hinweg Sicherheitsupdates. Der Long-term Support kann einen erheblichen Vorteil darstellen, wenn Sie Wert auf ein stabiles System legen, das nur selten aktualisiert werden soll.

11.1 So nehmen Sie die Aktualisierung vor

Bei der Aktualisierung der Linux-Distribution handelt es sich um einen Vorgang, der viele Bereiche des Betriebssystems betrifft. Das kann manchmal zu Problemen führen, die einen Datenverlust nach sich ziehen. Deshalb ist es empfehlenswert, vor dem Update stets ein Backup Ihrer persönlichen Daten vorzunehmen.

Nachdem diese Aufgabe erledigt ist, können Sie mit der Aktualisierung beginnen. Dazu müssen Sie das Startmenü öffnen und dort das Programm mit der Bezeichnung „Anwendungen & Aktualisierungen“ auswählen. Im nächsten Schritt müssen Sie in der Menüleiste auf „Aktualisierungen“ klicken. Daraufhin erscheinen die Inhalte, die in Abbildung 11.1 zu sehen sind.



Abb. 11.1 Die Benachrichtigungen für neue Versionen festlegen

In der letzten Zeile entdecken Sie den Eintrag „Über neue Ubuntu-Versionen benachrichtigen“. Da es sich bei Ubuntu 20.04 um eine Version mit Long-term Support handelt, ist hierbei der Wert „Für Langzeitunterstützungsversionen“ ausgewählt. Das bedeutet, dass Sie nur dann eine Benachrichtigung erhalten, wenn eine neue Version mit LTS verfügbar ist. Wenn Sie nun jedoch auf Ubuntu 20.10 oder auf eine andere Version ohne LTS aktualisieren möchten, müssen Sie hier „Für jede neue Version“ auswählen. Diese Auswahl müssen Sie durch Ihr Passwort bestätigen. Wenn Sie ganz auf Aktualisierungen verzichten möchten, können Sie die Benachrichtigungen auch deaktivieren.

Nun können Sie dieses Programm schließen. Danach müssen Sie wieder über das Startmenü die Aktualisierungsverwaltung aufrufen. In der Regel sollten hier einige Updates erscheinen – so wie dies in Abbildung 11.2 zu sehen ist. Darunter befindet sich jedoch noch nicht das Update für Ubuntu. Es ist jedoch zunächst notwendig, alle übrigen Updates zu installieren. Daher klicken wir nun auf „Jetzt installieren“.



11

Abb. 11.2 Die verfügbaren Updates

Auch diese Aktion müssen Sie mit Ihrem Passwort bestätigen. Nach dem Installationsprozess ist ein Neustart des Rechners erforderlich. Anschließend müssen Sie die Aktualisierungsverwaltung erneut aufrufen. Nun kann es vorkommen, dass Sie die eben vorgenommen Änderungen nochmals bestätigen müssen. Dabei kann es wieder zu einer gewissen Wartezeit kommen. Danach sollte das Fenster erscheinen, das in Abbildung 11.3 zu sehen ist. Dabei ist die Nachricht zu erkennen, dass eine neue Ubuntu-Version verfügbar ist. Wenn Sie nun auf „Aktualisieren“ klicken, erscheint noch ein weiteres Fenster, das Sie ebenfalls bestätigen müssen. Danach wird das neue System installiert.



Abb. 11.3 Das System ist bereit für die Aktualisierung

Die Aktualisierung des Systems kann einige Zeit in Anspruch nehmen – in der Regel etwa eine Stunde. Daher ist es empfehlenswert, diese Aktion nur dann auszuführen, wenn Sie den Computer in der nächsten Zeit nicht benötigen. Wenn das Update abgeschlossen ist, erscheint ein neuer Startbildschirm. Daran lässt sich sofort erkennen, dass Sie nun eine neue Version nutzen. Unter Ubuntu 20.10 sieht die Oberfläche so aus, wie dies in Abbildung 11.4 zu sehen ist.



Abb. 11.4 Die Oberfläche von Ubuntu 20.10

11.2 Übungsaufgaben

Es wurde gesagt, dass es sich bei Ubuntu 20.04 um eine Version mit Long-term Support handelt. Recherchieren Sie, in welchem Abstand diese veröffentlicht werden und durch welche Versionsnummern sie sich auszeichnen.

Lösung:

Seit 2006 erscheinen Versionen mit Long-term Support in zweijährigem Abstand. Dabei handelt es sich stets um die April-Version. Die einzige Ausnahme hiervon ist die Version 6.06, die im Juni erschien. Von dieser Ausnahme abgesehen, zeichnen sich Versionen mit Long-term Support daher durch eine gerade Zahl bei den Jahreszahlen und durch den Wert 04 bei der Monatsangabe aus. Darüber hinaus sind diese mit dem Zusatz LTS gekennzeichnet: 8.04 LTS, 10.04 LTS, 12.04 LTS, 14.04 LTS, 16.04 LTS, 18.04 LTS und 20.04 LTS.

Glossar

Betriebssystem	Das Betriebssystem (Operating System – OS) ist die unterste Softwareschicht eines Computers. Es stellt das Bindeglied zwischen der Hardware des Geräts und den Anwenderprogrammen dar. Es organisiert die Prozesse, die auf dem Rechner ablaufen und stellt ihnen die erforderlichen Ressourcen zur Verfügung.
Hardware	Der Begriff Hardware steht für die physischen Komponenten eines Computersystems. Dabei kann es sich sowohl um mechanische als auch um elektronische Bauteile handeln. Beispiele hierfür sind die Festplatte, der Prozessor, der Arbeitsspeicher und der Monitor.
Anwenderprogramm	Ein Anwenderprogramm ist eine Software, die verschiedene Dienste für einen Anwender ausführt. Es weist keine systemtechnische Funktionalität auf und hat nur einen eingeschränkten Zugriff auf die Hardware.
Arbeitsspeicher	Der Arbeitsspeicher (auch Hauptspeicher oder Random Access Memory – RAM) ist ein elektronischer Speicher, der die Daten zu den Programmen aufnimmt, die der Computer im Moment ausführt. Der Arbeitsspeicher zeichnet sich durch geringe Zugriffszeiten aus. Allerdings sind die Inhalte flüchtig: Sobald das Gerät ausgeschaltet wird, gehen sie verloren.

Peripheriegeräte	Peripheriegeräte sind alle Komponenten eines Computers, die nicht direkt in das Gerät verbaut sind. Sie werden über Kabel oder über die Funktechnik mit dem Rechner verbunden. Beispiele hierfür sind der Monitor, die Maus, USB-Speicher, Lautsprecher und zahlreiche weitere Geräte.
Datei	Eine Datei ist eine Menge an fest abgespeicherten Daten, die in der Regel inhaltlich zusammengehören. Die Datei sorgt dabei nicht nur für eine Abgrenzung. Außerdem hat sie einen bestimmten Dateityp. Dieser gibt an, um welche Art von Informationen es sich handelt und ist für die Interpretation der binären Daten sehr wichtig.
Prozessor	Der Prozessor (auch: Central Processing Unit – CPU) ist eines der wesentlichen elektronischen Bauteile eines Computers. Er kann Befehle und Daten aufnehmen und damit Berechnungen durchführen.
Grafische Benutzeroberfläche	Eine grafische Benutzeroberfläche (Graphical User Interface – GUI) stellt die sichtbare Ausgabe vieler Programme dar. Häufig ist diese als Programmfenster gestaltet. Sie kann Texte, Grafiken und Animationen enthalten. Außerdem ermöglicht sie durch Eingabefelder, Buttons und ähnliche Schaltflächen die Steuerung des Programms. Sie steht im Gegensatz zu Konsolenprogrammen, die über einen Kommandozeileninterpreter ausgeführt werden.
GNU-Projekt	Das GNU-Projekt entstand 1984 mit dem Ziel, ein freies, UNIX-ähnliches Betriebssystem zu entwickeln. Dieses Ziel wurde jedoch bis heute nicht vollständig umgesetzt. Dennoch entstand auf diese Weise eine große Menge an freier Systemsoftware. Diese kommt bei den meisten Linux-Distributionen zum Einsatz und führt gemeinsam mit dem Linux-Kernel zu einem voll funktionsfähigen Betriebssystem.

Terminal	Ein Terminal ermöglicht die Steuerung eines Computers. Dieser Begriff kann sich zum einen auf physische Terminals beziehen. Das sind spezielle Ein- und Ausgabegeräte mit Bildschirm und Tastatur, die insbesondere bei Servern häufig zum Einsatz kommen. Zum anderen kommt der Ausdruck für Programme zum Einsatz, die durch die Eingabe von Befehlen die Steuerung des Betriebssystems erlauben.
UNIX	Das Betriebssystem UNIX entstand 1969 und kam in den darauffolgenden Jahrzehnten auf vielen Rechnern in Unternehmen und Universitäten zum Einsatz. Das Betriebssystem Linux ist ein unixoides oder UNIX-ähnliches Betriebssystem, da es viele Funktionen von UNIX übernommen hat.
Multi-Boot-System	Beim Einschalten eines Computers lädt das Gerät ein Betriebssystem, mit dem es möglich ist, auf alle Funktionen zuzugreifen. Multi-Boot-Systeme geben dem Anwender nach dem Anschalten des Rechners die Möglichkeit, aus mehreren verschiedenen Betriebssystemen auszuwählen.
Systemprogramme	Der Begriff Systemprogramm oder Systemsoftware bezieht sich auf alle Programme, die der Steuerung des Geräts dienen. Von besonderer Bedeutung ist hierbei das Betriebssystem. Aber auch Treiber und Compiler zählen in diesen Bereich.
Command Line Interface	Ein Command Line Interface (CLI) oder Kommandozeileninterpreter ist eine Software, die die Steuerung des Computers über schriftliche Befehle erlaubt. Es steht im Gegensatz zu grafischen Benutzeroberflächen, die mit Fenstern und Schaltflächen arbeiten.

Live-System	Ein Live-System ist ein Betriebssystem, das nicht fest auf dem Computer installiert ist. Die kompletten Informationen befinden sich auf einem Wechseldatenträger. Live-Systeme dienen insbesondere der Datenrettung. Außerdem können sie die Sicherheit auf Geräten erhöhen, die mehreren Anwendern zur Verfügung stehen.
Repository	Ein Repository ist ein Speicherplatz für verschiedene Programme, die der Anwender in der Regel über das Internet herunterladen kann. Jede Linux-Distribution verwendet ihr eigenes Repository. Welche Programme darin zur Verfügung stehen, stellt eines der wesentlichen Merkmale der entsprechenden Distribution dar.
Partition	Eine Partition stellt einen Block innerhalb eines Datenträgers dar. Sie erlaubt es, Inhalte zu trennen und trägt dadurch zu einer höheren Datensicherheit bei. Außerdem erlauben es Partitionen, mehrere Betriebssysteme auf dem gleichen Gerät zu installieren.
BIOS	Das BIOS (Basic Input/Output System) stellt die Firmware bei x86-Rechnern dar. Dabei handelt es sich um einen separaten nicht-flüchtigen Speicher, der sich auf der Hauptplatine befindet. Sein Inhalt wird direkt beim Starten automatisch in den Arbeitsspeicher geladen. Seine Aufgabe besteht darin, den Startvorgang zu initialisieren und das Betriebssystem zu laden.
UEFI	UEFI (Unified Extensible Firmware Interface) ist eine Schnittstelle, die dazu dient, beim Booten des Computers das Betriebssystem zu laden. UEFI stellt eine Alternative zu BIOS dar und ersetzt dieses immer häufiger.

Prozess	Als Prozess wird ein Computerprogramm während dessen Ausführung bezeichnet. Es handelt sich um eine Instanz des Programms. Ein Prozess benötigt verschiedene Ressourcen für die Ausführung – insbesondere Rechenkapazität auf dem Prozessor und Speicherplatz auf dem Arbeitsspeicher.
Programm	Ein Programm ist eine Folge von Anweisungen, die der Prozessor ausführen soll. Diese Anweisungen sind auf einem Datenspeicher gespeichert. Wenn es zur Ausführung kommt, entsteht eine Instanz des Programms. Diese wird als Prozess bezeichnet.
Paketmanager	Ein Paketmanager verwaltet die Programme auf einem Computer. Er übernimmt insbesondere die Installation, die Deinstallation sowie die Durchführung von Updates. Verschiedene Linux-Distributionen verwenden unterschiedliche Paketmanager. Daher zählt diese Software zu den wesentlichen Unterscheidungskriterien zwischen den einzelnen Distributionen.
Builtins	Builtins sind Befehle, deren Funktionsweise direkt in eine Shell integriert sind. Sie umfassen deren elementaren Funktionen. Die Mehrheit der Befehle ist über eigene Programme implementiert.
Routinen	Eine Routine ist ein kurzes Programm oder ein Programmteil, mit dessen Hilfe es möglich ist, eine bestimmte Teilaufgabe zu erledigen.
Variablen	Variablen dienen in der Informatik dazu, einen bestimmten Wert innerhalb eines Programms aufzunehmen. Sie können beispielsweise einzelne Zahlen, Buchstaben oder auch Wörter speichern.

Iteration	Eine Iteration beschreibt eine mehrfache Wiederholung einer Aufgabe, um ein bestimmtes Ziel zu erreichen. In der Informatik kommt dieser Begriff beim Umgang mit zusammengesetzten Datentypen zum Einsatz. Er bezieht sich dabei darauf, dass eine bestimmte Aktion mit jedem vorhandenen Element ausgeführt wird.
Parameter	Parameter (auch Übergabewerte) sind Werte, die ein Programm beim Aufruf einer Funktion an diese übermittelt. Innerhalb der Funktion stehen diese dann für Berechnungen zur Verfügung.
Funktionen	Eine Funktion ist ein Programmabschnitt, der eine bestimmte Aufgabe erledigt und separat formuliert wird. Innerhalb des Programms lässt sich diese Aufgabe dann durch einen einfachen Aufruf der Funktion erledigen.
Long-term Support	Ubuntu bietet einige Versionen mit Long-term Support (LTS) an. Diese werden über fünf Jahre hinweg mit Sicherheitsupdates versorgt. Für Systeme, bei denen eine hohe Stabilität wichtig ist und bei denen möglichst wenig Aktualisierungen durchgeführt werden sollen, bietet es sich an, eine Version mit LTS zu wählen.
Print-Server	Ein Print-Server ist ein Computer, der Druckaufträge über ein Netzwerk entgegennimmt und ausführt. Dazu ist es zum einen notwendig, dass er an einen Drucker angeschlossen ist. Zum anderen muss der Print-Server entsprechend konfiguriert sein, um diese Aufgabe zu erledigen.

Shell	Die Shell ist eine Software, die mit dem Betriebssystem interagiert. Sie stellt die Schnittstelle zwischen diesem und dem Anwender dar. Obwohl es auch grafische Benutzerschnittstellen gibt, kommt dieser Ausdruck insbesondere für Befehlszeilenschnittstellen zum Einsatz, die Befehle in Textform entgegennehmen.
Relativer Pfad	Ein relativer Pfad beschreibt den Speicherort einer Datei. Als Ausgangspunkt verwendet er stets das Verzeichnis, in dem sich der Anwender beziehungsweise das Programm, das die Datei aufruft, befindet.
Absoluter Pfad	Der absolute Pfad gibt genau wie der relative Pfad den Speicherort einer Datei an. Im Gegensatz zu diesem verwendet er jedoch nicht das aktuelle Verzeichnis als Ausgangspunkt, sondern das Stammverzeichnis. Daher spielt es hierbei keine Rolle, in welchem Verzeichnis sich das Programm befindet, das die entsprechende Datei aufruft.

Index

A

addgroup.....	180
apt install	193
apt remove	194
apt search	192
apt update	191
apt upgrade.....	192
Array	256

B

break.....	280
------------	-----

C

case	269, 270
cat	209
cd	169
chmod	184
chown.....	188
continue.....	282
cp	298
crontab -e.....	300

D

Dateisystem	119
dir	164

E

echo	239
elif	268
else	267
expr.....	241

F

for	276
-----------	-----

G

gedit.....	144
grep.....	206
groupdel.....	182

H

head	217
help	225

I

if	262
ifconfig	310
info	231

K

Kernel.....	33
-------------	----

L

LibreOffice	132
Linux-Distribution	43
Live-Systeme	20
ls.....	164

M

man.....	230
Mikrokernel.....	36
mkdir.....	170
monolithische Kernel.....	36

more 204

P

Partition 67

passwd 177

Pipes 202

ps 157

R

read 259

return 289

rm 171

rmdir 171

S

sh 159

Shebang 245

Shell 154

sort 205

su 177

sudo 174

Systemüberwachung 128

T

tail 217

Terminal 154

Texteditor 142

U

until 274

useradd 174

userdel 176

usermod 180

V

Vergleichsoperatoren 265

W

while 273

Wildcards 194