Kapitel 1
Was ist Linux?

Um die einleitende Frage zu beantworten, erkläre ich in diesem Kapitel zuerst einige wichtige Begriffe, die im gesamten Buch immer wieder verwendet werden: Betriebssystem, Unix, Distribution, Kernel etc. Ein knapper Überblick über die Merkmale von Linux und die verfügbaren Programme macht deutlich, wie weit die Anwendungsmöglichkeiten von Linux reichen.

Es folgt ein kurzer Ausflug in die Geschichte von Linux. Von zentraler Bedeutung ist dabei natürlich die *General Public License* (kurz GPL), die angibt, unter welchen Bedingungen Linux weitergegeben werden darf. Erst die GPL macht Linux zu einem freien System, wobei »frei« mehr heißt als einfach »kostenlos«.

1.1 Einführung

Linux ist ein Unix-ähnliches Betriebssystem. Der wichtigste Unterschied gegenüber historischen Unix-Systemen besteht darin, dass Linux zusammen mit dem vollständigen Quellcode frei kopiert werden darf.

Betriebssystem

Ein Betriebssystem ist ein Bündel von Programmen, mit denen die Grundfunktionen eines Rechners realisiert werden. Dazu zählen die Verwaltung von Tastatur, Bildschirm, Maus sowie der Systemressourcen (CPU-Zeit, Speicher, SSDs etc.). Sie benötigen ein Betriebssystem, damit Sie ein Anwendungsprogramm überhaupt starten und eigene Daten in einer Datei speichern können. Populäre Betriebssysteme sind Windows, Linux, macOS, Android und iOS.

Unix versus Linux

Schon lange vor Windows und Linux gab es Unix. Dieses Betriebssystem war technisch gesehen seiner Zeit voraus: echtes Multitasking, eine Trennung der Prozesse voneinander, Zugriffsrechte für Dateien etc. Allerdings bot Unix anfänglich nur eine spartanische Benutzeroberfläche und stellte hohe Hardware-Anforderungen.

Inzwischen hat Linux Unix verdrängt: Große Teile des Internets werden von Linux-Servern getragen. Linux läuft in Form von Android auf Smartphones, in Routern und NAS-Festplatten sowie in Supercomputern: Die 500 schnellsten Rechner der Welt laufen heute *alle* unter Linux (*https://top500.org/statistics/list*).

Kernel

Genau genommen bezeichnet der Begriff Linux nur den Kernel: Er ist der innerste Teil (Kern) eines Betriebssystems mit ganz elementaren Funktionen, wie Speicherverwaltung, Prozessverwaltung und Steuerung der Hardware. Die Informationen in diesem Buch beziehen sich auf den Kernel 5.*n*.

1.2 Hardware-Unterstützung

Linux unterstützt beinahe die gesamte gängige PC-Hardware und läuft darüber hinaus auch auf anderen Hardware-Plattformen, z. B. auf Smartphones oder Embedded Devices. Dennoch müssen Sie beim Kauf eines neuen Rechners aufpassen. Es gibt einige Hardware-Komponenten, die im Zusammenspiel mit Linux oft Probleme machen:

- **Grafikkarten:** Fast alle auf dem Markt vertretenen Grafikkarten bzw. in die CPU integrierten Grafik-Cores funktionieren unter Linux. Für viele Linux-Anwender ohne besondere Anforderungen an das Grafiksystem sind Intel-CPUs mit eingebautem Grafik-Core die optimale Lösung. Grafikkarten von NVIDIA erfordern hingegen oft Zusatztreiber, damit die Karte perfekt genutzt werden kann. Die Installation dieser Treiber kann Probleme bereiten.
- **WLAN- und Netzwerkadapter:** WLAN- und LAN-Controller machen selten Probleme. Nur ganz neue Modelle werden von Linux mitunter noch nicht unterstützt.
- **Energiesparfunktionen:** Gerade neue Notebooks haben unter Linux oft deutlich kürzere Akku-Laufzeiten als unter Windows. Dieses Ärgernis resultiert daraus, dass das Zusammenspiel diverser Energiesparfunktionen optimale Treiber voraussetzt, die für Linux oft gar nicht oder erst ein, zwei Jahre nach der Markteinführung verfügbar sind.

Stellen Sie also *vor* dem Kauf eines neuen Rechners bzw. einer Hardware-Erweiterung sicher, dass alle Komponenten von Linux unterstützt werden. Auch eine Internetsuche nach *linux <hardwarename>* kann nicht schaden. Lesenswert sind außerdem Testberichte der Zeitschrift c't: Deren Redakteure machen sich bei den meisten Geräten die Mühe, auch die Linux-Kompatibilität zu testen.

Checkliste für den idealen Linux-Rechner

Wenn ich mir einen neuen Rechner kaufe, achte ich auf die folgenden Punkte:

- **CPU und Grafik:** Soweit möglich, ziehe ich Rechner ohne eigenen Grafik-Chip vor. Die in vielen CPUs integrierten Grafikfunktionen sind zwar langsamer, aber für viele Anwendungen ausreichend. Ihr Vorteil ist die hohe Kompatibilität zu Linux.
- **Kein Windows:** Mit Geräten ohne vorinstalliertes Windows können Sie ein paar Euro sparen. Solche Geräte finden Sie vor allem unter Campus-Angeboten, die sich an Studenten und Lehrpersonal richten, sowie bei Firmen, die sich auf Linux-Hardware spezialisiert haben (in Deutschland z. B. Tuxedo oder Cirrus7).

- **Lieber etwas älter:** Um ganz neue Notebooks mache ich einen großen Bogen, auch wenn die Spezifikationen noch so verlockend sind. Sie verursachen oft Treiberprobleme.

1.3 Distributionen

Noch immer ist die einleitende Frage – Was ist Linux? – nicht ganz beantwortet. Viele Anwender interessiert der Kernel nämlich herzlich wenig, sofern er nur läuft und die vorhandene Hardware unterstützt. Für sie umfasst der Begriff Linux, wie er umgangssprachlich verwendet wird, neben dem Kernel auch das riesige Bündel von Programmen, das mit Linux mitgeliefert wird: Dazu zählen neben unzähligen Kommandos die Desktop-Systeme KDE und Gnome, das Office-Paket LibreOffice, der Webbrowser Firefox, das Zeichenprogramm GIMP sowie zahllose Programmiersprachen und Server-Programme (Webserver, Mail-Server, File-Server etc.).

Als Linux-Distribution wird die Einheit bezeichnet, die aus dem eigentlichen Betriebssystem (Kernel) und den vielen Zusatzprogrammen gebildet wird. Eine Distribution ermöglicht eine rasche und bequeme Installation von Linux. Die meisten Distributionen können kostenlos aus dem Internet heruntergeladen werden.

Distributionen unterscheiden sich vor allem durch folgende Punkte voneinander:

- **Umfang, Aktualität:** Die Anzahl, Auswahl und Aktualität der mitgelieferten Programme und Bibliotheken variiert stark. Manche Distributionen setzen bewusst auf etwas ältere, stabile Versionen – z. B. Debian.
- **Installations- und Konfigurationswerkzeuge:** Die mitgelieferten Programme zur Installation, Konfiguration und Wartung des Systems helfen dabei, die Konfigurationsdateien einzustellen. Das kann viel Zeit sparen.
- **Konfiguration des Desktops (KDE, Gnome):** Manche Distributionen lassen dem Anwender die Wahl zwischen KDE, Gnome und anderen Desktop-Systemen. Auch die Detailkonfiguration und optische Gestaltung variiert je nach Distribution.
- **Hardware-Unterstützung:** Linux kommt mit den meisten PC-Hardware-Komponenten zurecht. Dennoch gibt es im Detail Unterschiede zwischen den Distributionen, insbesondere wenn es darum geht, Nicht-Open-Source-Treiber (z. B. für NVIDIA-Grafikkarten) in das System zu integrieren.
- **Updates:** Sie können eine Linux-Distribution nur so lange sicher betreiben, wie Sie Updates bekommen. Danach ist aus Sicherheitsgründen ein Wechsel auf eine neue Version der Distribution erforderlich. Deswegen ist es bedeutsam, wie lange es für eine Distribution Updates gibt. Hier gilt meist die Grundregel: je teurer der Support, desto länger der Zeitraum. Einige Beispiele (Stand: Sommer 2019):

CentOS:	10 Jahre
Fedora:	13 Monate
openSUSE:	ca. 18 bis 24 Monate
Red Hat Enterprise Linux:	10 Jahre (mit Einschränkungen sogar 13 Jahre)
SUSE Enterprise Server:	10 Jahre (mit Einschränkungen sogar 13 Jahre)
Ubuntu LTS:	3 bis 5 Jahre
Ubuntu (sonstige Versionen):	9 Monate

- **Live-System:** Viele Distributionen ermöglichen den Linux-Betrieb direkt von einer DVD oder von einem USB-Stick. Das ermöglicht ein einfaches Ausprobieren. Außerdem bieten Live-Systeme eine gute Möglichkeit, um ein defektes Linux-System zu reparieren.
- **Zielplattform (CPU-Architektur):** Viele Distributionen sind nur für Intel- und AMD-kompatible Prozessoren erhältlich. Es gibt aber auch Distributionen für andere Prozessorplattformen.
- **Support:** Bei kommerziellen Distributionen bekommen Sie Hilfe bei der Installation und im Betrieb.
- **Lizenz:** Die meisten Distributionen sind kostenlos erhältlich. Bei einigen Distributionen gibt es hier aber Einschränkungen: Beispielsweise ist bei den Enterprise-Distributionen von Red Hat und SUSE ein Zugriff auf das Update-System nur für registrierte Kunden möglich. Sie zahlen hier nicht für die Software an sich, wohl aber für das Service-Angebot rund herum.

Linux Standard Base (LSB)

Das Linux-Standard-Base-Projekt (LSB) definiert Regeln, um einen gemeinsamen Nenner zwischen den Distributionen zu schaffen. Die meisten Distributionen sind LSB-konform:

https://wiki.linuxfoundation.org/lsb/start

Gängige Linux-Distributionen

Der folgende Überblick über die wichtigsten verfügbaren Distributionen soll Ihnen eine erste Orientierungshilfe geben. Die Liste ist alphabetisch geordnet und erhebt keinen Anspruch auf Vollständigkeit.

Android

Android ist eine von Google entwickelte Plattform für Mobilfunkgeräte und Tablets. Android hat damit Linux zu der Weltdominanz verholfen, über die Linux-Entwickler in der Vergangenheit gescherzt haben.

Arch Linux

Arch Linux ist eine für technische Anwender optimierte Linux-Distribution. Die manuell im Textmodus durchzuführende Installation stellt sicher, dass Einsteiger einen großen Bogen um Arch Linux machen. Dafür zählen *https://wiki.archlinux.org* und *https://wiki.archlinux.de* zu den besten Quellen für Linux-Konfigurationsdetails

im Netz. Arch-Linux-Derivate wie **Manjero** und **Antergos** mit grafischen Installations- und Konfigurationsprogrammen haben Arch-Linux zuletzt sogar in die Top-10-Liste von *distrowatch.com* gebracht.

CentOS

CentOS ist eine kostenlose Variante zu Red Hat Enterprise Linux (RHEL). CentOS ist binärkompatibel zu RHEL, es fehlen aber alle Red-Hat-Markenzeichen, -Logos etc. Die Distribution ist vor allem für Server-Betreiber interessant, die kompatibel zu RHEL sein möchten, sich die hohen RHEL-Kosten aber nicht leisten können.

Chrome OS

Das **Chrome OS** wird wie Android von Google entwickelt. Es ist für Notebooks optimiert und setzt zur Nutzung eine aktive Internetverbindung voraus. Die Benutzeroberfläche basiert auf dem Webbrowser Google Chrome. Chrome OS spielt aktuell in Europa keine große Rolle, wohl aber auf dem Bildungsmarkt in den USA: Dort werden billige Chrome-Books (also Notebooks mit Chrome OS) häufig in Schulen eingesetzt.

Clear Linux

Clear Linux ist eine von Intel zusammengestellte Linux-Distribution, die im Hinblick auf maximale Effizienz optimiert ist. Clear Linux gewinnt regelmäßig alle erdenklichen Benchmark-Tests auf der Seite *https://phoronix.com*. Die Distribution richtet sich an Linux-Profis.

Debian

Debian ist die älteste vollkommen freie Distribution. Sie wird von engagierten Linux-Entwicklern zusammengestellt, wobei die Einhaltung der Spielregeln »freier« Software eine hohe Priorität genießt. Die strikte Auslegung dieser Philosophie hat in der Vergangenheit mehrfach zu Verzögerungen geführt.

Debian richtet sich an fortgeschrittene Linux-Anwender und hat einen großen Marktanteil bei Server-Installationen. Im Vergleich zu anderen Distributionen ist Debian stark auf maximale Stabilität hin optimiert und enthält deswegen oft relativ alte Programmversionen. Dafür steht Debian für viele Hardware-Plattformen zur Verfügung. Viele Distributionen sind von Debian abgeleitet, z. B. Raspbian und Ubuntu.

Fedora

Fedora ist der kostenlose Entwicklungszweig von Red Hat Linux. Die Entwicklung wird von Red Hat unterstützt und gelenkt. Für Red Hat ist Fedora eine Art Spielwiese, auf der neue Funktionen ausprobiert werden können, ohne die Stabilität der Enterprise-Versionen zu gefährden. Programme, die sich unter Fedora bewähren, werden später in die Enterprise-Versionen integriert. Bei technisch interessierten Linux-Fans ist Fedora beliebt, weil diese Distribution oft eine Vorreiterrolle spielt: Neue Linux-Funktionen finden sich oft zuerst in Fedora und erst später in anderen Distributionen. Neue Fedora-Versionen erscheinen alle sechs Monate. Updates werden einen Monat nach dem Erscheinen der übernächsten Version eingestellt, d. h., die Lebensdauer ist mit 13 Monaten sehr kurz.

Kali Linux

Das auf Debian basierende **Kali Linux** enthält eine riesige Sammlung von Hacking- und Pen-Testing-Werkzeugen. Die Distribution gilt als *der* Werkzeugkasten für Hacker und Sicherheits-Experten.

openSUSE

openSUSE ist eine kostenlose Linux-Distribution, die auf den Enterprise-Versionen von SUSE basiert, sich aber speziell an Privatanwender und Entwickler wendet.

Oracle

Oracle bietet unter dem Namen **Oracle Linux** eine Variante zu Red Hat Enterprise Linux (RHEL) an. Das ist aufgrund der Open-Source-Lizenzen eine zulässige Vorgehensweise. Technisch gibt es nur wenige Unterschiede zu RHEL, die Oracle-Variante ist aber billiger und ohne Support sogar kostenlos verfügbar. Dennoch ist die Verbreitung von Oracles Linux-Variante verhältnismäßig gering.

Raspbian

Raspbian ist die Standarddistribution für den beliebten Minicomputer Raspberry Pi. Raspbian basiert auf Debian, wurde für den Raspberry Pi aber speziell adaptiert und erweitert.

Red Hat

Die 2018 von IBM übernommene Firma **Red Hat** ist das international bekannteste und erfolgreichste Linux-Unternehmen. Red-Hat-Distributionen dominieren insbesondere den amerikanischen Markt. Die Paketverwaltung auf der Basis des *Red Hat Package Formats* (RPM) wurde von vielen anderen Distributionen übernommen.

Red Hat ist überwiegend auf Unternehmenskunden ausgerichtet. Die Enterprise-Versionen (RHEL = **Red Hat Enterprise Linux**) sind vergleichsweise teuer. Sie zeichnen sich durch hohe Stabilität und einen zehnjährigen Update-Zeitraum aus. Für Linux-Enthusiasten und -Entwickler, die ein Red-Hat-ähnliches System zum Nulltarif suchen, bieten sich **CentOS** und **Fedora** an.

SUSE

SUSE ist nach diversen Übernahmen die wichtigste deutsche Linux-Firma. Ihr Hauptprodukt, **SUSE Enterprise**, ist vor allem im europäischen Markt verankert.

Ubuntu

Ubuntu ist die zurzeit populärste Distribution für Privatanwender. Ubuntu verwendet als Basis Debian, ist aber besser für Desktop-Anwender optimiert (Motto: *Linux for human beings*). Die kostenlose Distribution erscheint im Halbjahresrhythmus. Für gewöhnliche Versionen werden Updates über neun Monate zur Verfügung gestellt. Für die alle zwei Jahre erscheinenden LTS-Versionen gibt es sogar 3 bzw. 5 Jahre lang Updates (für Desktop- bzw. Server-Pakete). Für kommerzielle Kunden bietet die Firma Canonical diverse Support-Angebote. Canonical hat sich damit vor allem im Server- und Cloud-Sektor zu den weltweit wichtigsten Linux-Firmen entwickelt.

Ubuntu-Derivate

Zu Ubuntu gibt es eine Menge offizieller und inoffizieller Varianten. Etabliert und weit verbreitet sind **Ubuntu Server**, **Kubuntu**, **Xubuntu**, **Ubuntu MATE** und **Linux Mint**. Die amerikanische Firma System76 pflegt mit **Pop!_OS** eine Ubuntu-Variante, die speziell für Notebooks optimiert ist und NVIDIA-Grafikkarten besonders gut unterstützt. Interessant ist auch **KDE Neon**: Diese Distribution kombiniert Ubuntu LTS mit stets aktuellen KDE-Paketen und hat sich als *die* Distribution für KDE-Fans etabliert.

Andere Distributionen

Neben den oben aufgezählten »großen« Distributionen gibt es im Internet zahlreiche Zusammenstellungen von Miniatursystemen. Sie sind vor allem für Spezialaufgaben konzipiert, etwa für Wartungsarbeiten (Emergency-Systeme) oder um ein Linux-System ohne eigentliche Installation verwenden zu können (Live-Systeme). Populäre Vertreter dieser Linux-Gattung sind **Devil Linux**, **Parted Magic** und **TinyCore**.

Einen ziemlich guten Überblick über alle momentan verfügbaren Linux-Distributionen, egal ob kommerziellen oder anderen Ursprungs, finden Sie im Internet auf der folgenden Seite:

https://distrowatch.com

Die Qual der Wahl

Eine Empfehlung für eine bestimmte Distribution ist schwierig. Für Linux-Einsteiger ist es zumeist von Vorteil, sich vorerst für eine weitverbreitete Distribution wie Debian, Fedora, openSUSE oder Ubuntu zu entscheiden. Eine gute Wahl ist auch Linux Mint. Zu diesen Distributionen sind sowohl im Internet als auch im Buch- und Zeitschriftenhandel viele Informationen verfügbar. Bei Problemen ist es vergleichsweise leicht, Hilfe zu finden.

Kommerzielle Linux-Anwender bzw. Server-Administratoren müssen sich entscheiden, ob sie bereit sind, für professionellen Support Geld auszugeben. In diesem Fall spricht wenig gegen die Marktführer Red Hat, Canonical und SUSE. Andernfalls sind CentOS, Debian und Ubuntu attraktive kostenlose Alternativen.

1.4 Open-Source-Lizenzen (GPL & Co.)

Die Grundidee von »Open Source« besteht darin, dass der Quellcode von Programmen frei verfügbar ist und von jedem erweitert bzw. geändert werden darf. Allerdings ist damit auch eine Verpflichtung verbunden: Wer Open-Source-Code zur Entwicklung eigener Produkte verwendet, muss den gesamten Code ebenfalls wieder frei weitergeben.

Die Open-Source-Idee verbietet übrigens keinesfalls den Verkauf von Open-Source-Produkten. Auf den ersten Blick scheint das ein Widerspruch zu sein. Tatsächlich bezieht sich die Freiheit in »Open Source« mehr auf den Code als auf das fertige Produkt. Zudem regelt die freie Verfügbarkeit des Codes auch die Preisgestaltung von Open-Source-Produkten: Nur wer neben dem Kompilat eines Open-Source-Programms weitere Zusatzleistungen anbietet (Handbücher, Support etc.), wird überleben. Sobald der Preis in keinem vernünftigen Verhältnis zu den Leistungen steht, werden sich andere Firmen finden, die es günstiger machen.

General Public License (GPL)

Das Ziel der Open-Source-Entwickler ist es, Software zu schaffen, deren Quellen frei verfügbar sind und es auch bleiben. Um einen Missbrauch auszuschließen, sind viele

Open-Source-Programme durch die *GNU General Public License* (kurz GPL) geschützt. Hinter der GPL steht die *Free Software Foundation* (FSF). Diese Organisation wurde von Richard Stallman gegründet, um hochwertige Software frei verfügbar zu machen. Richard Stallman ist übrigens auch der Autor des Editors Emacs, der in Kapitel 15 beschrieben wird.

Die Kernaussage der GPL besteht darin, dass zwar jeder den Code verändern und sogar die resultierenden Programme verkaufen darf, dass aber gleichzeitig der Anwender/Käufer das Recht auf den vollständigen Code hat und diesen ebenfalls verändern und wieder kostenlos weitergeben darf. Jedes GNU-Programm muss zusammen mit dem vollständigen GPL-Text weitergegeben werden. Die GPL schließt damit aus, dass jemand ein GPL-Programm weiterentwickeln und verkaufen kann, *ohne* die Veränderungen öffentlich verfügbar zu machen. Jede Weiterentwicklung ist somit ein Gewinn für *alle* Anwender. Den vollständigen Text der GPL finden Sie hier:

https://gnu.org/licenses/gpl.html

Das Konzept der GPL ist recht einfach zu verstehen, im Detail treten aber immer wieder Fragen auf. Viele davon werden hier beantwortet:

https://gnu.org/licenses/gpl-faq.html

Wenn Sie glauben, dass Sie alles verstanden haben, sollten Sie das GPL-Quiz ausprobieren:

https://gnu.org/cgi-bin/license-quiz.cgi

Lesser General Public License (LGPL)

Neben der GPL existiert noch die Variante LGPL (Lesser GPL). Der wesentliche Unterschied zur GPL besteht darin, dass eine derart geschützte Bibliothek auch von kommerziellen Produkten genutzt werden darf, deren Code *nicht* frei verfügbar ist. Ohne die LGPL könnten GPL-Bibliotheken nur wieder für GPL-Programme genutzt werden, was in vielen Fällen eine unerwünschte Einschränkung für kommerzielle Programmierer wäre.

Andere Lizenzen

Durchaus nicht alle Teile einer Linux-Distribution unterliegen den gleichen Copyright-Bedingungen! Obwohl der Kernel und viele Tools der GPL unterliegen, gelten für manche Komponenten und Programme andere rechtliche Bedingungen:

- **MIT- und BSD-Lizenz:** Die MIT- und BSD-Lizenzen erlauben die kommerzielle Nutzung des Codes *ohne* die Verpflichtung, Änderungen öffentlich weiterzugeben. Die Lizenzen sind damit wesentlich liberaler als die GPL und eher mit der LGPL vergleichbar.
- **Doppellizenzen:** Für einige Programme gelten Doppellizenzen. Beispielsweise können Sie den Datenbank-Server MySQL für Open-Source-Projekte, auf einem eigenen Webserver bzw. für die innerbetriebliche Anwendung gemäß der GPL kostenlos einsetzen. Wenn Sie hingegen ein kommerzielles Produkt auf der Basis von MySQL entwickeln und samt MySQL verkaufen möchten, ohne Ihren Quellcode zur Verfügung zu stellen, dann kommt die kommerzielle Lizenz zum Einsatz. Die Weitergabe von MySQL wird in diesem Fall kostenpflichtig.
- **Kommerzielle Lizenzen:** Einige Programme unterstehen zwar einer kommerziellen Lizenz, dürfen aber dennoch kostenlos genutzt werden. Ein bekanntes Beispiel ist das Flash-Plugin von Adobe: Zwar ist das Programm unter Linux kostenlos erhältlich (und darf auch in Firmen kostenlos eingesetzt werden), aber der Quellcode zu diesem Programm ist nicht verfügbar.

Manche Distributionen kennzeichnen die Produkte, bei denen die Nutzung oder Weitergabe eventuell lizenzrechtliche Probleme verursachen könnte. Bei Debian befinden sich solche Programme in der Paketquelle *non-free*.

Das Dickicht der zahllosen, mehr oder weniger »freien« Lizenzen ist schwer zu durchschauen. Die Bandbreite zwischen der manchmal fundamentalistischen Auslegung von »frei« im Sinne der GPL und den verklausulierten Bestimmungen mancher Firmen, die ihr Software-Produkt zwar frei nennen möchten (weil dies gerade modern ist), in Wirklichkeit aber uneingeschränkte Kontrolle über den Code behalten möchten, ist groß. Eine gute Einführung in das Thema geben die beiden folgenden Websites:

https://opensource.org
https://heise.de/-221957

Lizenzkonflikte zwischen Open- und Closed-Source-Software

Open-Source-Lizenzen für Entwickler

Wenn Sie Programme entwickeln und diese zusammen mit Linux bzw. in Kombination mit Open-Source-Programmen oder -Bibliotheken verkaufen möchten, müssen Sie sich in die bisweilen verwirrende Problematik der unterschiedlichen Software-Lizenzen tiefer einarbeiten. Viele Open-Source-Lizenzen erlauben die Weitergabe nur, wenn auch Sie Ihren Quellcode im Rahmen einer Open-Source-Lizenz frei verfügbar machen. Auf je mehr Open-Source-Komponenten mit unterschiedlichen Lizenzen Ihr Programm basiert, desto komplizierter wird die Weitergabe.

Es gibt aber auch Ausnahmen, die die kommerzielle Nutzung von Open-Source-Komponenten erleichtern: Beispielsweise gilt für Apache und PHP sinngemäß, dass Sie diese Programme auch in Kombination mit einem Closed-Source-Programm frei weitergeben dürfen.

GPL-Probleme mit Hardware-Treibern

Manche proprietäre Treiber für Hardware-Komponenten (z. B. für NVIDIA-Grafikkarten) bestehen aus einem kleinen Kernelmodul (Open Source) und diversen externen Programmen oder Bibliotheken, deren Quellcode nicht verfügbar ist (Closed

Source). Das Kernelmodul hat nur den Zweck, eine Verbindung zwischen dem Kernel und dem Closed-Source-Treiber herzustellen.

Diese Treiber sind aus Sicht vieler Linux-Anwender eine gute Sache: Sie sind kostenlos verfügbar und ermöglichen es, diverse Hardware-Komponenten zu nutzen, zu denen es entweder gar keine oder zumindest keine vollständigen Open-Source-Treiber für Linux gibt. Die Frage ist aber, ob bzw. in welchem Ausmaß die Closed-Source-Treiber wegen der engen Verzahnung mit dem Kernel, der ja der GPL untersteht, diese Lizenz verletzen. Viele Open-Source-Entwickler dulden die Treiber nur widerwillig. Eine direkte Weitergabe mit GPL-Produkten ist nicht zulässig, weswegen der Benutzer die Treiber in der Regel selbst herunterladen und installieren muss.

1.5 Die Geschichte von Linux

1982: GNU

Da Linux ein Unix-ähnliches Betriebssystem ist, müsste ich an dieser Stelle eigentlich mit der Geschichte von Unix beginnen – aber dazu fehlt hier der Platz. Stattdessen beginnt diese Geschichtsstunde mit der Gründung des GNU-Projekts durch Richard Stallman. GNU steht für *GNU is not Unix*. In diesem Projekt wurden seit 1982 Open-Source-Werkzeuge entwickelt. Dazu zählen der GNU-C-Compiler, der Texteditor Emacs sowie diverse GNU-Utilities wie `find` und `grep` etc.

1989: GPL

Erst sieben Jahre nach dem Start des GNU-Projekts war die Zeit reif für die erste Version der *General Public License*. Diese Lizenz stellt sicher, dass freier Code frei bleibt.

1991: Linux-Kernel 0.01

Die allerersten Teile des Linux-Kernels (Version 0.01) entwickelte Linus Torvalds. Er gab seinen Code im September 1991 über das Internet frei. Schnell fanden sich weltweit Programmierer, die an der Idee Interesse hatten und Erweiterungen dazu programmierten. Als der Kernel von Linux die Ausführung des GNU-C-Compilers erlaubte, stand auch die gesamte Palette der GNU-Tools zur Verfügung. Weitere Komponenten waren das Dateisystem Minix, Netzwerk-Software von BSD-Unix, das X Window System des MIT und dessen Portierung XFree86 etc.

Linux ist also nicht nur Linus Torvalds zu verdanken. Hinter Linux stehen vielmehr eine Menge engagierter Menschen, die in ihrer Freizeit, im Rahmen ihres Studiums oder bezahlt von Firmen wie Google, IBM oder HP freie Software produzieren.

1994: Erste Distributionen

Informatik-Freaks an Universitäten konnten sich Linux und seine Komponenten selbst herunterladen, kompilieren und installieren. Eine breite Anwendung fand Linux aber erst mit Linux-Distributionen, die Linux und die darum entstandene Software auf Disketten bzw. CD-ROMs verpackten und mit einem Installationsprogramm versahen. Vier der zu dieser Zeit entstandenen Distributionen existieren heute noch: Debian, Red Hat, Slackware und SUSE.

1996: Pinguin

1996 wurde der Pinguin zum Linux-Logo.

1998: Microsoft nimmt Linux wahr

Mit dem rasanten Siegeszug des Internets stieg auch die Verbreitung von Linux, vor allem auf Servern. Gewissermaßen zum Ritterschlag für Linux wurde der legendäre Ausspruch von Steve Ballmer: *Microsoft is worried about free software* ... Ein Jahr später ging Red Hat spektakulär an die Börse.

2009: Android

Mit der Android-Plattform brachte Google Linux zuerst auf das Handy (2009), danach auch auf Tablets und in TV-Geräte.

2012: Raspberry Pi

2012 eroberte der Minicomputer Raspberry Pi die Herzen von Elektronikbastlern. Für nur rund 40 EUR können Sie mit dem Raspberry Pi selbst Hardware-Experimente durchführen, in die Welt der Heimautomation einsteigen, ein Medien-Center oder einen Home-Server betreiben. Der Raspberry Pi macht Embedded Linux zu einem Massenphänomen.

2018: IBM kauft Red Hat, Microsoft umarmt Open Source und Linux

2018 erwarb IBM die Fima Red Hat für beachtliche 34 Mrd. US-Dollar. Gleichzeitig wendet sich Microsoft unter der Führung von Satya Nadella immer stärker der Open-Source-Idee und Linux zu. Das *Windows Subsystem for Linux* erlaubt die Ausführung von Linux direkt in Windows (also ohne Virtualisierung). Mit dem Kauf von GitHub, der ebenfalls 2018 über die Bühne ging, beherrscht Microsoft nun auch das wichtigste Repository für Open-Source-Projekte.

1.6 Software-Patente und andere Ärgernisse

Patente schützen in den USA und anderen Ländern Software-Ideen, -Konzepte und Algorithmen. Alles Mögliche und Unmögliche ist patentiert, triviale Dinge wie die Darstellung eines Fortschrittsbalkens oder die 1-Click-Bestellung (Amazon). Der Missbrauch derartiger Trivialpatente und die für die schnelllebige Software-Branche sehr langen Laufzeiten von 20 Jahren tragen zum Widerwillen gegen Software-Patente bei.

Beispielsweise verzichteten viele Distributionen jahrelang aus Angst vor Klagen darauf, Bibliotheken zum Abspielen von MP3-Dateien mitzuliefern; die darin eingesetzten Algorithmen sind durch Patente geschützt. Die Anwender mussten die zum Abspielen von MP3-Dateien erforderlichen Bibliotheken selbst installieren. Glücklicherweise liefen die MP3-Patente 2017 aus, sodass zumindest dieses Problem aus der Welt geschafft ist.

Während Patente selten ein Risiko für einzelne Software-Entwickler sind, spielen sie im Kampf um Marktanteile eine immer größere Rolle, z. B. im heiß umkämpften Smartphone- und Tablet-Markt.

Patent-Pools der Open-Source-Gemeinde

Ganz aussichtslos ist die Lage zum Glück nicht. Das liegt vor allem daran, dass einige Linux nahestehende Firmen wie IBM selbst über riesige Patent-Pools verfügen. Diverse Linux-Firmen haben zudem begonnen, selbst Patente zu sammeln, die teilweise von anderen Firmen gleichsam für Open-Source-Zwecke »gespendet« wurden.

Das Absurde an der Situation besteht darin, dass ein verfehltes Patentrecht die Open-Source-Gemeinde dazu zwingt, selbst Patente einzusetzen, um sich gegen eventuelle Klagen zu schützen. Details über Patent-Pools der Open-Source-Gemeinde finden Sie hier:

https://openinventionnetwork.com

Multimedia

Auch abseits der MP3-Dateien ist der Multimedia-Markt ein Problemfeld. Beispielsweise können Sie unter Linux DVDs nicht ohne Weiteres abspielen. Diverse Gesetze verbieten in vielen Ländern sowohl die Weitergabe der erforderlichen Bibliotheken als auch die bloße Beschreibung, wie diese zu installieren sind – z. B. das Urheberrechtsgesetz in Deutschland.

Digital Rights Management

Nicht besser sieht es mit online erworbenen Daten (Videos, E-Books etc.) aus, die durch DRM geschützt sind. DRM steht für *Digital Rights Management* und bezeichnet diverse Verfahren, um die Nutzung der Daten so einzuschränken, dass sie nur auf einem ganz bestimmten Rechner möglich ist. Sozusagen nebenbei werden Sie dadurch auf eine bestimmte Hardware (z. B. iPod oder iPhone) bzw. auf ein bestimmtes Betriebssystem (z. B. Windows, macOS) beschränkt. DRM-Gegner bezeichnen das System nicht umsonst als *Digital Restriction Management*.

Kapitel 13
Netzwerk-Tools

Dieses Kapitel stellt Kommandos zur Benutzung, Steuerung und Analyse elementarer Netzwerkdienste vor. Sie lernen hier, wie Sie sich mit `ssh` auf einem anderen Rechner im Netzwerk einloggen und mit `wget` Dateien übertragen. Mit den Programmen Lynx und Mutt können Sie im Textmodus sogar Webseiten besuchen und Mails lesen und verfassen.

Weitere Kommandos zur Analyse des Netzwerkstatus sowie zur Suche nach offenen Ports auf fremden Rechnern, `netstat` und `nmap`, stelle ich Ihnen in Kapitel 33, »Firewalls«, vor.

13.1 Netzwerkstatus ermitteln

Dieser Abschnitt gibt einen Überblick über Kommandos zum Test der Grundfunktionen des Netzwerks. Weitere Informationen zu den hier vorgestellten Kommandos folgen in Abschnitt 18.3, »Manuelle LAN- und WLAN-Konfiguration«.

Netzwerkschnittstellen ermitteln

Das Kommando `ip addr` liefert eine Liste aller bekannten Netzwerkschnittstellen:

```
root#  ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN ...
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s31f6: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel ...
    link/ether 48:2a:e3:16:aa:24 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.82/24 brd 10.0.0.255 scope global dynamic noprefixroute enp0s31f6
       valid_lft 56324sec preferred_lft 56324sec
    inet6 fe80::b1bb:3bd5:8e0e:a6d9/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: wlp0s20f3: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN ...
    link/ether 34:e1:2d:e7:c1:c4 brd ff:ff:ff:ff:ff:ff
```

Typische Schnittstellen sind `eth<n>` oder `enp<n>s<m>` (Ethernet) sowie `wlan<n>` oder `wlp<xxx>` (WLAN). Beim Rechner, auf dem das obige Listing entstanden ist, gibt es einen WLAN-Adapter, es ist aber keine WLAN-Verbindung eingerichtet.

Bei den meisten gängigen Distributionen fließt in den Namen der Ethernet-Schnittstelle die interne Bus-Nummer ein – z. B. `enp0s1`. Auf PCs und Notebooks mit nur einer Ethernet-Schnittstelle wirkt der Name umständlich. Aber die busspezifische Nummerierung stellt sicher, dass sich auf Servern mit vielen Netzwerkschnittstellen die Nummerierung auch dann nicht ändert, wenn weitere Netzwerkadapter hinzugefügt werden.

Eine Sonderrolle nimmt die Schnittstelle `lo` ein: Sie ermöglicht es lokalen Programmen, über das Netzwerkprotokoll zu kommunizieren. Das funktioniert selbst dann, wenn ein Rechner nicht nach außen hin mit einem Netzwerk verbunden ist.

Wenn `ip addr` nur bei der Schnittstelle `lo` eine IP-Adresse angibt, wurde noch keine Netzwerkschnittstelle aktiviert. Abhilfe schafft das von Ihrer Distribution vorgesehene Werkzeug zur Netzwerkkonfiguration. Sie können die Netzwerkschnittstelle mit dem `ip`-Kommando auch manuell aktivieren. Details dazu sowie zur IPv6-Konfiguration finden Sie in Kapitel 18, »Netzwerkkonfiguration«.

Erreichbarkeit von localhost testen

`ping` sendet einmal pro Sekunde ein kleines Netzwerkpaket an die angegebene Adresse. Wenn sich dort ein Rechner befindet, sendet dieser eine Antwort, es sei denn, eine Firewall verhindert das. `ping` läuft so lange, bis es mit Strg+C beendet wird. `ping localhost` überprüft, ob das Loopback-Interface und damit die elementaren Netzwerkfunktionen des eigenen Rechners funktionieren:

```
user$  ping localhost
PING localhost (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=255 time=0.152 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=255 time=0.114 ms
...
```

Erreichbarkeit des lokalen Netzes testen

Indem Sie an `ping` statt `localhost` die IP-Nummer eines anderen Rechners im lokalen Netz übergeben, testen Sie, ob das lokale Netz funktioniert. `-c 2` bewirkt, dass `ping` nicht endlos läuft, sondern nach zwei Paketen endet:

```
user$  ping -c 2 192.168.0.99
PING 192.168.0.99 (192.168.0.99): 56 data bytes
64 bytes from 192.168.0.99: icmp_seq=0 ttl=255 time=0.274 ms
64 bytes from 192.168.0.99: icmp_seq=1 ttl=255 time=0.150 ms
...
```


Wenn es im lokalen Netz einen Nameserver gibt, der der IP-Nummer 192.168.0.99 einen Namen zuordnet, oder wenn die Datei `/etc/hosts` diese Aufgabe übernimmt, können Sie bei `ping` statt der IP-Nummer den Rechnernamen angeben:

```
user$  ping -c 2 mars
PING mars.sol (192.168.0.99) 56(84) bytes of data.
64 bytes from mars.sol (192.168.0.99): icmp_seq=1 ttl=64 time=0.281 ms
64 bytes from mars.sol (192.168.0.99): icmp_seq=2 ttl=64 time=0.287 ms

--- mars.sol ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.281/0.284/0.287/0.003 ms
```

Internetzugang testen

Als Nächstes können Sie testen, ob die Verbindung zum Internet gelingt. Das folgende Kommando testet gleichzeitig zwei Aspekte der Netzwerkkonfiguration: die Erreichbarkeit des Nameservers und die Funktion des Gateways.

```
user$  ping -c 2 google.com
PING google.com (172.217.22.78) 56(84) bytes of data.
64 bytes from fra15s17-in-f14.1e100.net (172.217.22.78): ... time=25.5 ms
64 bytes from fra15s17-in-f14.1e100.net (172.217.22.78): ... time=25.6 ms

--- google.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 3ms
rtt min/avg/max/mdev = 25.509/25.549/25.589/0.040 ms
```

Wenn das nicht funktioniert, sind mehrere Ursachen denkbar:

- Vielleicht ist der Server von Google gerade unerreichbar (das ist sehr unwahrscheinlich), oder der Server hat aus Sicherheitsgründen die Antwort auf `ping` deaktiviert. Probieren Sie eine andere bekannte Internetadresse aus.
- Für die Ermittlung der IP-Adresse zu `google.com` ist der Nameserver verantwortlich. Wenn Sie die Fehlermeldung *unknown host google.com* erhalten, gibt es Probleme mit dem Nameserver. Überprüfen Sie, ob `/etc/resolv.conf` die Adresse eines Nameservers enthält.
- Das Gateway ist dafür zuständig, IP-Pakete aus dem lokalen Netzwerk an das Internet weiterzuleiten. Wenn das nicht funktioniert, erhalten Sie die Fehlermeldung *connect: Network is unreachable*. Die Gateway-Konfiguration können Sie mit `ip route` überprüfen. Das Kommando liefert normalerweise mehrere Zeilen. Die Gateway-Adresse befindet sich in der dritten Spalte der Zeile, die mit `default` beginnt:

  ```
  user$  ip route
  default via 10.0.0.138 dev eth0
  10.0.0.0/24 dev eth0  proto kernel  scope link  src 10.0.0.42
  ```

- Falls Sie in einem lokalen Netz einen eigenen Rechner als Gateway eingerichtet haben, besteht die Möglichkeit, dass Sie die Masquerading-Funktion vergessen haben. In diesem Fall würde der Internetzugang für das gesamte lokale Netzwerk nicht funktionieren.

Den Weg von IP-Paketen verfolgen

Mit `traceroute` finden Sie heraus, welchen Weg ein Netzwerkpaket von Ihrem Rechner zu einem anderen Rechner nimmt und wie viele Millisekunden die Laufzeit bis zur jeweiligen Zwischenstation beträgt. Standardmäßig unternimmt das Kommando drei Versuche und liefert daher entsprechend drei Zeiten. Das Kommando funktioniert nicht, wenn sich auf einer der Zwischenstationen eine Firewall befindet, die den von `traceroute` genutzten UDP-Port 33434 blockiert. In diesem Fall liefert `traceroute` für diese und alle weiteren Stationen nur noch drei Sterne.

Folgende Zeilen zeigen den Weg von meinem Arbeitsrechner zu `google.com`. Zeile 1 gibt die IP-Adresse meines ADSL-Routers an, Zeile 2 das Gateway meines Internet-Providers:

```
user$  traceroute google.com
traceroute to google.com (216.58.206.14), 30 hops max, 60 byte packets
  1  _gateway (10.0.0.138)                                    1.178 ms ...
  2  91-114-247-254.adsl.highway.telekom.at (91.114.247.254)  14.512 ms ...
  3  195.3.74.81 (195.3.74.81)                                15.545 ms ...
  4  lg1-9071.as8447.a1.net (195.3.64.14)                     17.040 ms ...
...
 11  fra16s20-in-f14.1e100.net (216.58.206.14)                33.093 ms ...
```

Firewalls umgehen

Mitunter behindern Sicherheitseinstellungen und Firewalls die Arbeit von `traceroute`. Anstelle von IP-Adressen zeigt das Kommando dann nur `* * *` an. In solchen Fällen können Sie versuchen, mit den Optionen `-T` oder `-I` andere Verfahren zu verwenden, um den Weg von Paketen zu verfolgen. Beide Optionen erfordern `root`-Rechte.

mtr

Das Kommando `mtr` sendet regelmäßig Netzwerkpakete zum angegebenen Host und analysiert die Antworten. Die Ergebnisliste kombiniert Daten von `ping` und `traceroute`:

```
user$  mtr -c 10 -r google.com
Start: 2019-05-09T16:00:34+0200
HOST: p1                          Loss%   Snt   Last   Avg  Best  Wrst StDev
  1.|-- _gateway                   0.0%    10    1.5   1.2   0.9   1.5   0.2
  2.|-- 91-114-247-254.adsl.highw  0.0%    10    6.7   6.9   6.3   7.5   0.4
  3.|-- 195.3.74.81                0.0%    10    9.9   9.5   8.9  10.2   0.4
  4.|-- lg1-9071.as8447.a1.net     0.0%    10   10.3   9.6   8.8  10.3   0.4
 ...
 11.|-- fra16s18-in-f142.1e100.ne  0.0%    10   25.2  25.3  24.7  26.2   0.4
```


gnome-nettool

Unter Gnome können Sie einen Großteil der oben aufgezählten Informationen ganz komfortabel mit dem Programm `gnome-nettool` ermitteln (siehe Abbildung 13.1). Bei einigen Distributionen müssen Sie das gleichnamige Paket zuerst installieren.

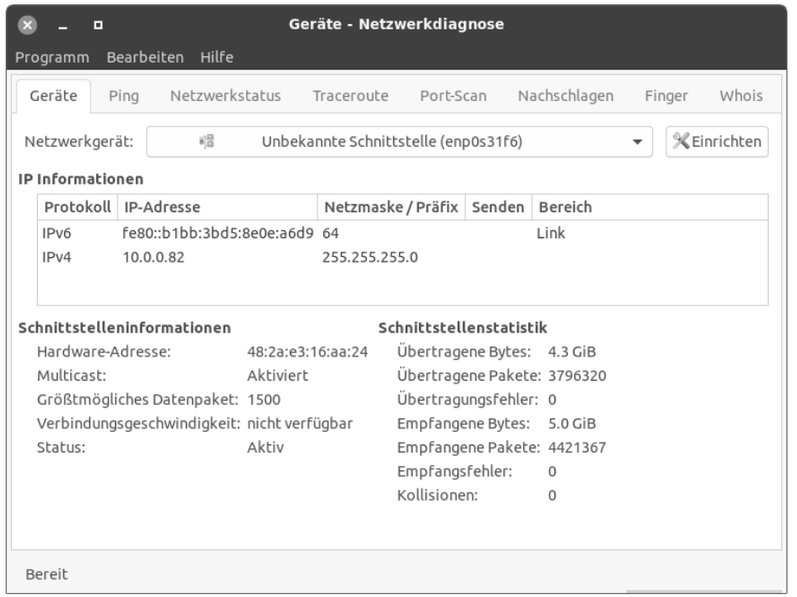

Abbildung 13.1 Netzwerkdiagnose unter Gnome

13.2 Auf anderen Rechnern arbeiten (SSH)

Mit dem Kommando `ssh` können Sie auf einem anderen Rechner im Textmodus arbeiten, als stünde er vor Ihnen. Dazu wären auch `telnet` und `rlogin` in der Lage – aber diese beiden Kommandos sollten aus Sicherheitsgründen nicht mehr eingesetzt werden. Sie übertragen die Login-Informationen inklusive des Passworts unverschlüsselt.

Die Grundvoraussetzung für die Anwendung von `ssh` besteht darin, dass auf dem zweiten Rechner ein SSH-Server läuft, also das Programm `sshd`. Bei manchen Linux-Distributionen ist dies standardmäßig der Fall, bei anderen muss das Programm (zumeist als Paket `openssh-server`) zuerst installiert und/oder aktiviert werden (`systemctl start sshd` und `systemctl enable sshd`). Wenn auf den Rechnern Firewalls laufen, dürfen diese den Port 22 nicht blockieren.

Einen eigenen SSH-Server einrichten

Informationen zur Installation, Konfiguration und Absicherung eines SSH-Servers folgen in Kapitel 26, »Secure Shell (SSH)«. Dort erfahren Sie auch, wie Sie den SSH-Server absichern.

Gewöhnliche Shell-Session

Wenn Sie auf dem Rechner `uranus` arbeiten und nun eine Shell-Session auf dem Rechner `mars` starten möchten, führen Sie zum Verbindungsaufbau das folgende Kommando aus:

```
user@uranus$  ssh mars
user@mars's password:  ********
```

Beim ersten Verbindungsaufbau zu einem neuen Rechner erscheint eine Warnung nach dem folgenden Muster:

```
The authenticity of host 'mars (192.168.0.10)' can't be established.
RSA1 key fingerprint is 1e:0e:15:ad:6f:64:88:60:ec:21:f1:4b:b7:68:f4:32.
Are you sure you want to continue connecting (yes/no)?  yes
Warning: Permanently added 'mars,192.168.0.10' (RSA1) to the list
  of known hosts.
```

Das bedeutet, dass `ssh` sich nicht sicher ist, ob es dem Rechner `mars` mit der IP-Adresse 192.168.0.10 vertrauen darf. Es könnte sein, dass ein fremder Rechner vortäuscht, `mars` zu sein. Wenn Sie die Rückfrage mit `YES` beantworten, speichert `ssh` den Namen, die Adresse und den RSA-Fingerprint (einen Code zur eindeutigen Identifizierung des Partnerrechners) in `~/.ssh/known_hosts`.

Falls Sie auf `mars` unter einem anderen Login-Namen als auf `uranus` arbeiten möchten (z. B. als `root`), geben Sie den Namen mit der Option `-l` an:

```
user@uranus$  ssh -l root mars
root@mars's password:  *********
```

SSH-Authentifizierung mit Schlüsseln

Wesentlich sicherer als ein Login mit Passwort ist die Authentifizierung durch einen Schlüssel. Die Vorgehensweise wird im Detail in Abschnitt 26.4 beschrieben. Die Verwendung von Schlüsseln ermöglicht es auch, auf SSH basierende Kommandos und Scripts automatisch per Script auszuführen.

Kommandos ausführen

Statt `ssh` interaktiv zu nutzen, können Sie auf dem entfernten Rechner auch einfach nur ein Kommando ausführen. Das Kommando und seine Parameter werden einfach als weitere Parameter an `ssh` übergeben. `ssh` endet nach diesem Kommando.

```
user@uranus$  ssh mars kommando optionen
user@mars's password:  *********
```


Aus dieser scheinbar trivialen Funktion ergeben sich weitreichende Möglichkeiten: Sie können nun beispielsweise auf dem entfernten Rechner `tar` starten, das damit erstellte Archiv an die Standardausgabe weiterleiten (geben Sie dazu einen Bindestrich - nach der Option `-f` ein, also `-f -`) und die Standardausgabe mit | als Eingabe für ein zweites `tar`-Kommando verwenden, das lokal läuft. Damit können Sie einen ganzen Verzeichnisbaum sicher via SSH kopieren.

Das folgende Kommando zeigt, wie ich den gesamten `/var/www`-Verzeichnisbaum meines Webservers `kofler.info` in das lokale Verzeichnis `~/bak` kopiere. Das Kommando setzt dabei voraus, dass alle Dateien in `/var/www` vom Benutzer `username` gelesen werden können.

```
user$  ssh -l username kofler.info tar -cf - /var/www | tar -xC ~/bak/ -f -
username@kofler.info's password:  ******
```

Mehrere Kommandos ausführen (Scripts)

Wenn Sie in einem Script mehrere Kommandos via SSH ausführen möchten, verwenden Sie am besten die Heredoc-Syntax (siehe auch Abschnitt 9.11, »Code-Strukturierung in bash-Scripts«):

```
#!/bin/bash
pw=strenggeheim
...
ssh -T root@host <<ENDSSH
echo root:$pw | chpasswd
rm -f /etc/file1
cp /root/file2 /userxy/file3
ENDSSH
...
```

Damit führt `ssh` alle Kommandos aus, bis im Script die mit `ENDSSH` markierte Zeile erreicht wird. Die Option `-T` verhindert dabei, dass SSH versucht, ein Pseudo-Terminal zu öffnen. Das ist hier unerwünscht, weil die Kommandoausführung nicht interaktiv erfolgen soll.

Rückfrage bei erstem Verbindungsaufbau verhindern

Beim ersten SSH-Verbindungsaufbau zu einem neuen Host fragt `ssh`, ob Sie dem Host vertrauen. Normalerweise ist diese Rückfrage sinnvoll. Wenn Sie aber mit `ssh` automatisiert auf mehreren Hosts (oft in virtuellen Maschinen) Arbeiten durchführen möchten, stört die Rückfrage. Abhilfe schafft in solchen Fällen die Option `-o StrictHostKeyChecking=no`.

Diese und andere Optionen können Sie auch global in `/etc/ssh/ssh_config` oder individuell für einen Benutzer in `.ssh/config` einstellen. Verwechseln Sie aber `/etc/ssh/ssh_config` nicht mit `/etc/ssh/sshd_config`! Die erste Datei enthält SSH-Client-Optionen, die zweite Datei Optionen für den SSH-Server.

SSH und X

Sofern als Grafiksystem X und nicht Wayland verwendet wird (sowohl auf dem Client als auch auf dem Server!), können Sie in einer SSH-Verbindung, die Sie mit `ssh -X` initiiert haben, auch Grafikprogramme ausführen. Die Option `-X` ist erforderlich, damit sich `ssh` um die korrekte Einstellung der `DISPLAY`-Variablen kümmert.

```
user@localhost$  ssh -X otheruser@otherhost
otheruser@otherhost$  firefox &   (Firefox läuft extern, wird aber lokal angezeigt)
```

Dateien sicher kopieren mit scp

Um eine Datei via SSH über das Netzwerk zu kopieren, gibt es das Kommando `scp`. Die Syntax sieht so aus:

```
user$  scp [[user1@]host1:]filename1 [[user2@]host2:][filename2]
user2@host2's password:  ******
```

Damit wird die Datei `filename1` vom Rechner `host1` zum Rechner `host2` übertragen und dort in der Datei `filename2` gespeichert. Einige Anmerkungen zu den vielen optionalen Bestandteilen der Kopieranweisung:

- `host1` und `host2` müssen nicht angegeben werden, wenn der lokale Rechner (also `localhost`) gemeint ist.
- `user1` muss nicht angegeben werden, wenn der aktive Benutzer gemeint ist.
- `user2` muss nicht angegeben werden, wenn auf dem Rechner `host2` der aktuelle Benutzername von `host1` bzw. `user1` verwendet werden soll.
- `filename1` darf auch ein Verzeichnis sein. Sie müssen dann die Option `-r` angeben, damit das gesamte Verzeichnis mit allen Unterverzeichnissen übertragen wird.
- `filename2` muss nicht angegeben werden, wenn der Dateiname unverändert bleiben soll. Die Datei wird dann in das Home-Verzeichnis von `user2` kopiert.

  Statt `filename2` kann auch das Zielverzeichnis angegeben werden, wobei wie üblich `~` für das Home-Verzeichnis von `user2` verwendet wird.

Nehmen Sie an, die Benutzerin `gabi` arbeitet auf dem Rechner `uranus`. Sie will die Datei `abc.txt` in das Verzeichnis `~/efg` auf dem Rechner `mars` übertragen. Das `scp`-Kommando sieht so aus:

```
gabi@uranus$  scp abc.txt mars:~/efg/
gabi@mars's password:  ******
```

Falls Sie beim `scp`-Kommando eine IPv6-Adresse angeben wollen, müssen Sie diese in eckige Klammern stellen. Andernfalls kommt `scp` bei den vielen Doppelpunkten durcheinander.

```
user$  scp kofler@[2001:1234:5678::1]:datei.txt .
```

`scp` kann auch rekursiv ganze Verzeichnisbäume übertragen:

```
user$  scp -rp srcdir user@desthost:/destdir
```


Sicherheitsprobleme in »scp«

Anfang 2019 wurden einige Sicherheitsprobleme beim `scp`-Kommando entdeckt. Diese sind mittlerweile zwar behoben, die verantwortlichen Entwickler weisen aber daraufhin, dass das `scp`-Protokoll veraltet und schlecht zu warten ist. Sie empfehlen den Einsatz des auf SSH basierenden Protokolls SFTP (siehe Abschnitt 13.3, »Dateien übertragen (FTP & Co.)«) oder von `rsync` (siehe Abschnitt 32.3, »Verzeichnisse synchronisieren (rsync)«).

SSH-Tunnel

Eine SSH-Anwendungsmöglichkeit für fortgeschrittene Linux-Anwender ist der Tunnelbau. Derartige Tunnel eignen sich zwar nicht als Transportmöglichkeit für Autos oder Züge, sie ermöglichen aber die Übertragung aller IP-Pakete, die an einen bestimmten Port gerichtet sind. SSH-Tunnel bieten damit einen sicheren Weg, um IP-Pakete zwischen zwei Rechnern zu übertragen – und das selbst dann, wenn sich zwischen den beiden Rechnern eine Firewall befindet, die den Port eigentlich blockiert. Eine Einführung in die Welt der IP-Pakete und eine Erklärung des Begriffs *Port* finden Sie in Kapitel 33, »Firewalls«.

Wenn der Tunnelbau vom Client-Rechner aus erfolgt, kommt die Option `-L localport:localhost:remoteport` zum Einsatz. Beispielsweise bewirkt das folgende Kommando, dass der Port 3306 des Rechners `mars` über den Port 3307 des lokalen Rechners zugänglich ist. Durch das Kommando wird gleichzeitig eine SSH-Session gestartet, was Sie durch `-N` aber verhindern können (wenn Sie nur den Tunnel, aber keine Shell benötigen). Falls der Login bei `mars` unter einem anderen Namen erfolgen soll, müssen Sie den Login-Namen wie üblich durch `-l name` oder durch `name@remotehost` angeben.

```
user@uranus$  ssh -L 3307:localhost:3306 username@mars
user@mars's password:  ******
```

Der Tunnel bleibt so lange offen, bis die SSH-Session mit Strg+D beendet wird. Falls Sie `ssh` mit der Option `-N` gestartet haben, muss das Programm mit Strg+C gestoppt werden.

3306 ist der übliche Port von MySQL. Sie können nun auf dem Rechner `uranus` über dessen Port 3307 auf den MySQL-Server zugreifen, der auf `mars` läuft. Beim `mysql`-Kommando müssen Sie den Port 3307 und den Hostname 127.0.0.1 angeben, damit der SSH-Tunnel tatsächlich benutzt wird. Standardmäßig stellt `mysql` lokale Verbindungen über eine Socket-Datei her.

```
user@uranus$  mysql -u mysqllogin -P 3307 -h 127.0.0.1 -p
Enter password:  *******
```

Damit der MySQL-Login funktioniert, müssen zwei Voraussetzungen erfüllt sein:

- Erstens muss der MySQL-Server auf dem Rechner `mars` grundsätzlich IP-Verbindungen akzeptieren. Der MySQL-Server kann aus Sicherheitsgründen auch so konfiguriert sein, dass Verbindungen nur über eine Socket-Datei möglich sind. Dann hilft ein Tunnel nicht weiter, weil ein Tunnel nur Ports verbinden kann.
- Zweitens muss der MySQL-Server die Kombination aus Login-Name und Hostname akzeptieren. Als Hostname wird der Name des Rechners verwendet, zu dem `ssh` den Tunnel errichtet hat – hier also `mars` bzw. `mars.sol`, wenn die Domain `sol` lautet.

SSH-Dateisystem

Mit dem Kommando `sshfs`, das sich bei vielen Distributionen im gleichnamigen Paket befindet, können Sie das Dateisystem eines externen Rechners in den lokalen Verzeichnisbaum integrieren. Das kann beispielsweise die Durchführung von Backups vereinfachen.

```
root#  mkdir /media/ext-host
root#  sshfs user@hostname /media/ext-host
root#  ...
root#  umount /media/ext-host
```

Beachten Sie aber, dass Sie im SSH-Dateisystem wegen der Verschlüsselung aller Daten zumeist einen geringeren Durchsatz als mit Samba oder NFS erzielen werden. Das SSH-Dateisystem ist deswegen für den Einsatz in lokalen Netzwerken nur bedingt geeignet. Ich habe zudem die Erfahrung gemacht, dass `sshfs` auf kurzzeitige Netzwerkausfälle allergisch reagiert und dann hängen bleibt. Ich bin deswegen vom Einsatz dieses an sich praktischen Dateisystems wieder abgekommen.

telnet

Ein Vorgänger von `ssh` war `telnet`. Da `telnet` keine Daten verschlüsselt, sollte das Kommando auf keinen Fall dazu verwendet werden, um auf externen Rechnern zu arbeiten. Aktuelle Linux-Distributionen lassen dies standardmäßig ohnedies nicht zu, aber man stößt immer wieder auf Router, ADSL-Modems etc., die diese Art der Kommunikation zulassen.

Der Grund, warum ich Ihnen hier `telnet` überhaupt präsentiere, ist ein anderer: `telnet` eignet sich gut dazu, um zu überprüfen, ob auf einem externen Rechner auf einem bestimmten Port ein Netzwerkdienst läuft und auf einen Verbindungsaufbau wartet. Beispielsweise können Sie mit `telnet` sicherstellen, dass der zuvor eingerichtete Mail-Server tatsächlich läuft. Dazu übergeben Sie an `telnet` den Namen oder die IP-Adresse des Servers sowie die Port-Nummer:

```
user$  telnet kofler.info 25
Trying 5.9.22.29...
Connected to kofler.info.
Escape character is '^]'.
220 kofler.info ESMTP Postfix (Ubuntu)
 helo kofler.info
250 kofler.info
^]      (Verbindung mit Strg + ] beenden)
```


13.3 Dateien übertragen (FTP & Co.)

Grundlagen

FTP steht für *File Transfer Protocol* und bezeichnet ein recht altes Verfahren zur Übertragung von Dateien über ein Netzwerk. Seine große Popularität verdankt FTP der Spielart Anonymous FTP: Viele große Internet-Server bieten allen Anwendern Zugang zu sogenannten FTP-Archiven. Dieser Zugang ist (im Gegensatz zum sonstigen FTP) nicht durch ein Passwort versperrt.

Ein großer Nachteil von FTP besteht darin, dass beim Login-Prozess der Benutzername und das Passwort unverschlüsselt übertragen werden. Eine sichere Alternative ist SFTP (Secure FTP) auf der Basis von SSH (siehe Kapitel 26, »Secure Shell (SSH)«). Auch HTTP, also das Protokoll zur Übertragung von Webseiten, wird oft als Alternative zu FTP eingesetzt.

In diesem Kapitel geht es nur um die Nutzung von FTP, also um die Client-Sichtweise. Damit FTP funktioniert, muss auf der Gegenstelle ein FTP-Server laufen. Dessen Konfiguration ist in Abschnitt 27.9, »FTP-Server (vsftpd)«, beschrieben.

FTP-Kommando

Der Urahn aller FTP-Clients ist das interaktive Textkommando `ftp`. Da es Dateien normalerweise aus dem aktuellen Verzeichnis bzw. in das aktuelle Verzeichnis überträgt, sollten Sie vor dem Start von `ftp` mit `cd` in das gewünschte Arbeitsverzeichnis wechseln. Die FTP-Sitzung wird dann mit dem Kommando `ftp user@ftpservername` oder einfach `ftp ftpservername` eingeleitet. Falls Sie Anonymous FTP nutzen möchten, geben Sie als Benutzernamen `anonymous` ein.

Nach dem Verbindungsaufbau und der Eingabe des Passworts kann es losgehen: Mit den Kommandos `cd`, `pwd` und `ls`, die dieselbe Bedeutung wie unter Linux haben, können Sie sich durch die Verzeichnisse des FTP-Archivs bewegen. Um eine Datei vom FTP-Archiv in das aktuelle Verzeichnis Ihres Rechners zu übertragen, führen Sie `get datei` aus. Der Dateiname bleibt dabei unverändert.

Umgekehrt können Sie mit `put` eine Datei aus Ihrem aktuellen Verzeichnis in ein Verzeichnis des FTP-Archivs übertragen. Das geht freilich nur dann, wenn Sie eine Schreiberlaubnis für das Verzeichnis haben. Bei Anonymous FTP ist das zumeist nur für ein Verzeichnis mit einem Namen wie `/pub/incoming` der Fall. Die FTP-Sitzung wird mit dem Kommando `quit` oder `bye` beendet. Eine Referenz der wichtigsten FTP-Kommandos finden Sie in Tabelle 13.1.

Kommando	Funktion
`?`	zeigt eine Liste aller FTP-Kommandos an.
`cd verz`	wechselt in das angegebene FTP-Verzeichnis.
`close`	beendet die Verbindung zum FTP-Server.
`get datei`	überträgt die Datei vom FTP-Archiv in das aktuelle Verzeichnis.
`lcd verz`	wechselt das aktuelle Verzeichnis auf dem lokalen Rechner.
`ls`	zeigt die Liste der Dateien auf dem FTP-Server an.
`lls`	zeigt die Liste der Dateien auf dem lokalen Rechner an.
`mget *.muster`	überträgt alle passenden Dateien vom FTP-Archiv in das aktuelle Verzeichnis.
`open`	stellt die Verbindung zum fremden Rechner her (wenn es beim ersten Versuch nicht geklappt hat).
`put datei`	überträgt die Datei in das FTP-Archiv (*upload*).
`quit`	beendet FTP.
`reget datei`	setzt die Übertragung einer bereits teilweise übertragenen Datei fort.

Tabelle 13.1 Wichtige ftp-Kommandos

Andere FTP-Programme

Das Kommando `ftp` ist nicht komfortabel zu bedienen. Zum Glück gibt es unzählige Alternativen. Die meisten unter Linux verfügbaren Webbrowser und Dateimanager können auch zum FTP-Download verwendet werden. Wenn Sie im Textmodus arbeiten möchten, bietet sich das interaktivie Kommando `ncftp` an. Die Kommandos `wget`, `curl` und `lftp` helfen bei der automatisierten Übertragung von Dateien bzw. ganzer Verzeichnisbäume via FTP.

SFTP (Secure FTP)

Das Kommando `sftp` ist Teil des `openssh`-Pakets. `sftp` verwendet intern ein ganz anderes Protokoll als `ftp` und kann wie `ssh` nur eingesetzt werden, wenn auf der Gegenstelle ein SSH-Server läuft. Anonymous FTP ist mit `sftp` nicht möglich. Davon abgesehen, erfolgt die Bedienung des Programms wie die von `ftp`. Mit `sftp -b batchdatei` können Sie SFTP-Downloads automatisieren.

Auch mit den Dateimanagern Dolphin (KDE) oder Nautilus (Gnome) können Sie eine SFTP-Verbindung initiieren, indem Sie die Adresse `sftp://user@servername` eingeben.

Nach der Passwortabfrage zeigen die Programme das FTP-Verzeichnis wie ein lokales Verzeichnis an. Beide Dateimanager unterstützen auch direkt das SSH-Protokoll, das selbst dann funktioniert, wenn `sftp` nicht zur Verfügung steht. Dazu geben Sie die Adresse in der Form `fish://user@servername` an.

wget

Der interaktive Ansatz des Kommandos `ftp` ist zur Automatisierung von Downloads – beispielsweise in einem Script – ungeeignet. Auch sonst ist `ftp` reichlich inflexibel. Beispielsweise ist es unmöglich, einen unterbrochenen Download selbstständig wieder aufzunehmen. Abhilfe schafft das Kommando `wget`, das speziell zur Durchführung großer Downloads bzw. zur Übertragung ganzer Verzeichnisse konzipiert ist. `wget` unterstützt gleichermaßen die Protokolle FTP, HTTP und HTTPS.

Beispiele

In der Grundform lädt `wget` die angegebene Datei einfach herunter:

```
user$ wget ftp://myftpserver.de/name.abc
```

Wenn der Download aus irgendeinem Grund unterbrochen wird, kann er mit `-c` ohne Umstände wieder aufgenommen werden:

```
user$ wget -c ftp://myftpserver.de/name.abc
```

Downloads von großen Dateien, beispielsweise von ISO-Images von Linux-Distributionen, dauern bei einem nicht so guten Internetzugang mehrere Stunden. Da bietet es sich an, den Download über Nacht durchzuführen. Das folgende Kommando stellt nahezu sicher, dass sich die Datei am nächsten Morgen tatsächlich auf dem Rechner befindet. Wegen `-t 20` wird der Download nach einem Verbindungsabbruch bis zu 20-mal neu aufgenommen. `--retry-connrefused` bewirkt, dass selbst nach dem Fehler *connection refused* ein neuer Versuch gestartet wird. Das ist dann zweckmäßig, wenn der Download-Server bekanntermaßen unzuverlässig ist und immer wieder für kurze Zeit unerreichbar ist.

```
user$ wget -t 20 --retry-connrefused http://mydownloadserver.de/name.iso
```

Das folgende Kommando lädt sämtliche Dateien herunter, die notwendig sind, um die angegebene Webseite später in unverändertem Zustand offline zu lesen. Kurz zur Bedeutung der Optionen: `-p` lädt auch CSS-Dateien und Bilder herunter. `-k` verändert in den heruntergeladenen Dateien die Links, sodass diese auf lokale Dateien verweisen. `-E` fügt heruntergeladenen Script-Dateien (ASP, PHP etc.) die Kennung `.html` hinzu. `-H` verfolgt auch Links auf externe Websites.

```
user$ wget -p -k -E -H http://mywebsite.de/seite.html
```

Wenn Sie eine ganze Website offline lesen möchten, hilft das folgende rekursive Download-Kommando (Option `-r`). Die Rekursionstiefe wird durch `-l 4` auf vier Ebenen limitiert.

```
user$  wget -r -l 4 -p -E -k http://mywebsite.de
```

curl

Das Kommando `curl` hilft dabei, Dateien von oder zu FTP-, HTTP- oder sonstigen Servern zu übertragen. Die `man`-Seite listet eine beeindruckende Palette von Protokollen auf, die `curl` beherrscht. In diesem Abschnitt beschränke ich mich allerdings auf FTP-Uploads. Für die Script-Programmierung besonders praktisch ist, dass `curl` auch Daten aus der Standardeingabe verarbeiten bzw. zur Standardausgabe schreiben kann. Sie müssen also nicht zuerst eine `*.tar.gz`-Datei erstellen und diese dann zum FTP-Server übertragen, sondern können beide Operationen mittels einer Pipe gleichzeitig ausführen.

Das folgende Kommando überträgt die angegebene Datei zum FTP-Server `backupserver` und speichert sie im Verzeichnis `verz`:

```
user$  curl -T datei -u username:password ftp://backupserver/verz
```

Um Daten aus dem Standardeingabekanal zu verarbeiten, geben Sie mit `-T` als Dateinamen einen Bindestrich an. Das folgende Kommando speichert das aus dem `tar`-Kommando resultierende Ergebnis direkt in der Datei `name.tgz` auf dem FTP-Server:

```
user$  tar czf - verz/ | curl -T - -u usern:pw ftp://bserver/name.tgz
```

lftp

`lftp` ist ein komfortabler interaktiver FTP-Client. Das Kommando eignet sich aber auch gut, um FTP-Uploads oder andere Kommandos in einem Script auszuführen. Dazu können Sie an `lftp` entweder mit `-c` mehrere durch Strichpunkte getrennte FTP-Kommandos übergeben oder mit `-f` eine Datei angeben, die diese Kommandos zeilenweise enthält. Das erste Kommando wird dabei immer `user benutzername,passwort servername` lauten, um die Verbindung zum FTP-Server herzustellen. Das folgende Kommando demonstriert einen Datei-Upload:

```
root#  lftp -c "open -u username,password backupserver; put www.tgz"
```

Wenn Sie der Datei auf dem FTP-Server einen anderen Namen geben möchten, geben Sie zusätzlich die Option `-o <neuerName>` an. `lftp` zeigt während des Uploads den aktuellen Fortschritt an.

Um statt einer Datei ein ganzes Verzeichnis zum Backup-Server zu übertragen, verwenden Sie das Kommando `mirror -R`. (`mirror` kopiert normalerweise Verzeichnisse

vom FTP-Server auf den lokalen Rechner. `-R` dreht die Übertragungsrichtung um.) Auch hierzu ein Beispiel:

```
root#  lftp -c "open -u usern,passw bserver; mirror -R verzeichnis"
```

Im Unterschied zu anderen FTP-Clients unterstützt `lftp` das Kommando `du`, mit dem Sie feststellen können, wie viel Speicherplatz Ihre Backup-Dateien bereits belegen. Das ist dann wichtig, wenn Ihr Speicherplatz auf dem Backup-Server streng limitiert ist. Das folgende Kommando zeigt, wie Sie ohne interaktiven Eingriff den bereits belegten Speicherplatz ermitteln. Die Option `-s` gibt an, dass Sie nur an der Endsumme interessiert sind. `-m` bewirkt, dass als Maßeinheit MiB verwendet wird.

```
user$  lftp -c "open -u username,password bserver; du -s -m"
2378   .
```

Wenn Sie das Ergebnis für eine Berechnung verwenden möchten, stört die zweite Spalte (also der Punkt, der angibt, dass sich der Zahlenwert auf das aktuelle Verzeichnis bezieht). Stellen Sie dem Kommando einfach `cut -f 1` hintan, um die erste Spalte zu extrahieren:

```
user$  lftp -c "open -u usern,passw bserver; du -s -m" | cut -f 1
2378
```

rsync, mirror, sitecopy

`rsync` hilft dabei, ganze Verzeichnisbäume zu kopieren bzw. zu synchronisieren. Eine ausführliche Beschreibung dieses Kommandos finden Sie in Abschnitt 32.7, »Verzeichnisse synchronisieren (rsync)«. Sofern auf dem Partnerrechner weder ein SSH- noch ein `rsync`-Server läuft, können Sie anstelle von `rsync` auf die Kommandos `mirror` oder `sitecopy` zurückgreifen. Das Perl-Script `mirror` aus dem gleichnamigen Paket kopiert ganze Verzeichnisbäume von einem FTP-Server auf den lokalen Rechner. Das Kommando `sitecopy` ist hingegen dahingehend optimiert, einen Verzeichnisbaum auf einen Webserver hochzuladen, wobei der Datentransfer wahlweise via FTP oder WebDAV erfolgt.

13.4 Lynx

Webbrowser im Textmodus

Webbrowser wie Firefox oder Chrome sind in einer Textkonsole oder in einem Terminalfenster unbrauchbar. Um dennoch auch im Textmodus rasch eine Webseite zu besuchen oder ein HTML-Dokument zu lesen, helfen Programme wie ELinks, Lynx oder w3m. Nebenbei können Sie mit diesen Programmen einfache HTML-Dokumente in reinen Text umwandeln. Alle drei Programme sind ähnlich zu bedienen. Zahlreiche Optionen sowie Tastenkürzel sind in den `man`-Seiten bzw. im integrierten Hilfesystem

dokumentiert. Aus Platzgründen stelle ich hier nur exemplarisch das bekannteste Programm Lynx näher vor.

Lynx

Die Bedienung von Lynx ist einfach: Sie starten das Programm im Regelfall dadurch, dass Sie eine WWW-Adresse oder den Namen einer HTML-Datei als Parameter angeben. Lynx lädt das Dokument und zeigt die erste Seite an, wobei Überschriften und Links durch unterschiedliche Farben gekennzeichnet sind. Wenn Sie Lynx mit der Option `-use_mouse` starten, können Sie das Programm auch per Maus bedienen: Mit der linken Taste folgen Sie einem Link, die mittlere Taste zeigt ein Kontextmenü an, und die rechte Taste führt zur vorherigen Seite zurück.

Lynx verwendet zur Ausgabe standardmäßig den Latin-1-Zeichensatz. Damit Sonderzeichen in Unicode-Konsolen richtig dargestellt werden, geben Sie die Option `-display_charset=utf-8` an. Das folgende Kommando zeigt, wie Sie Lynx als Konverter von HTML in reinen Text einsetzen:

```
user$  lynx -dump quelle.html > ziel.txt
```

13.5 Mutt

Zum Lesen lokaler E-Mails bietet sich das textbasierte E-Mail-Programm Mutt an (siehe Abbildung 13.2). Vor dem ersten Einsatz muss das zumeist gleichnamige Paket installiert werden. In einem Konsolenfenster führen Sie zuerst `su -l` aus, um sich als `root` anzumelden, und starten das Programm dann mit dem Kommando `mutt`.

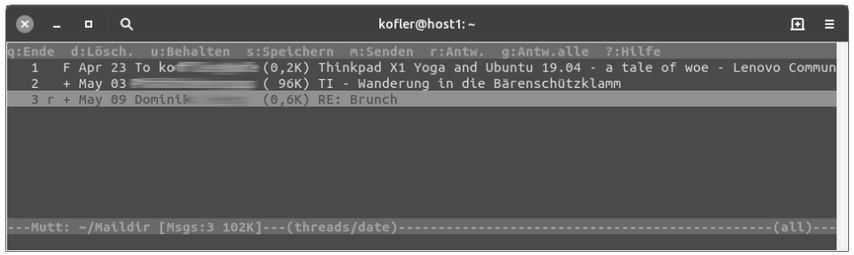

Abbildung 13.2 Lokale E-Mails mit Mutt lesen

Das Programm zeigt auf der Startseite die Titelzeilen aller E-Mails an. Wenn der aktive Benutzer noch keine einzige E-Mail empfangen hat, beklagt sich Mutt darüber, dass es die Datei `/var/mail/benutzer` noch nicht gibt. Diese Warnung können Sie ignorieren. Sie tritt nicht mehr auf, sobald die erste E-Mail eingetroffen ist.

Mit den Cursortasten bewegen Sie sich durch die Inbox. ↵ zeigt den Text der ausgewählten E-Mail an. Mit der Leertaste blättern Sie durch die Nachricht. J führt zur

nächsten Nachricht, I zurück in die Inbox. ? zeigt einen Hilfetext mit allen wichtigen Tastenkürzeln an.

Um eine neue E-Mail zu verfassen, drücken Sie M und geben den Empfänger und die Subject-Zeile an. Anschließend startet Mutt den Editor, den Sie mit der Umgebungsvariable `$EDITOR` oder mit dem Link `/etc/alternatives/editor` ausgewählt haben. Dort schreiben Sie den Nachrichtentext, speichern ihn und verlassen den Editor. Anschließend versenden Sie die E-Mail in Mutt durch Y.

Q beendet das Programm. Beim Verlassen stellt Mutt zwei Fragen: Sollen mit D als gelöscht markierte E-Mails endgültig gelöscht werden? Und sollen gelesene Nachrichten nach `/home/username/mbox` verschoben werden? Wenn Sie vorhaben, die E-Mails später noch mit einem anderen Programm zu bearbeiten, sollten Sie beide Fragen mit N beantworten. Besonders die zweite Frage ist kritisch: In der lokalen `mbox`-Datei findet nur noch Mutt die E-Mails, nicht aber ein externes Programm wie z. B. der POP-Server Dovecot.

Konfiguration

Mutt funktioniert auf Anhieb, wenn sich Ihre E-Mail in einer mbox-Datei im Verzeichnis `/var/mail/name` befindet. Wenn Ihre E-Mails hingegen im Maildir-Format im Verzeichnis `Maildir` gespeichert werden, müssen Sie die Konfigurationsdatei `.muttrc` mit dem folgenden Inhalt einrichten:

```
# Datei .muttrc
set mbox_type=Maildir
set folder="~/Maildir"
set mask="!^\\.[^.]"
set mbox="~/Maildir"
set record="+.Sent"
set postponed="+.Drafts"
set spoolfile="~/Maildir"
```

Weitere Maildir-Konfigurationstipps für diverse Spezialfälle finden Sie hier:

https://gitlab.com/muttmua/mutt/wikis/MuttFaq/Maildir
https://eising.wordpress.com/mutt-maildir-mini-howto

Kapitel 16
Atom und VSCode

Wenn Sie die beiden vorigen Kapitel gelesen haben, glauben Sie vielleicht, Linux wäre in der Steinzeit stehen geblieben. Aber keine Sorge, unter Linux laufen natürlich auch unzählige Editoren mit einer modernen Oberfläche. Zwei davon möchte ich Ihnen in diesem Kapitel näher vorstellen: Atom und Visual Studio Code, im Folgenden kurz VSCode.

Die beiden Editoren haben etliche Gemeinsamkeiten: Sie basieren auf Electron, einem Framework, das sich aus Teilen des Webbrowsers Chromium und der JavaScript-Bibliothek Node.js zusammensetzt. Beide Editoren laufen auf allen gängigen Plattformen, also unter Linux, Windows und macOS. Beide Editoren sind noch recht jung: Die ersten stabilen Versionen von Atom bzw. VSCode wurden im Februar 2016 bzw. im April 2016 veröffentlicht. Beide Editoren lassen sich unkompliziert durch Plugins erweitern. Und bei beiden Editoren steht der Quellcode unter einer Open-Source-Lizenz zur Verfügung.

Unterschiede gibt es in der Herkunft:

- Das Atom-Projekt wurde von der Firma GitHub initiiert – quasi als Git-kompatibler Editor für Entwickler. Atom kommt mit nahezu jeder Art von Code- und Textformaten zurecht, ganz egal, ob die Dateien unter der Versionskontrolle von Git stehen oder nicht.
- VSCode wurde hingegen maßgeblich von einem Team rund um den Microsoft-Mitarbeiter Erich Gamma (Co-Autor des Buchs *Entwurfsmuster*) entwickelt. Lassen Sie sich übrigens nicht vom Namen *Visual Studio Code* täuschen: Der Editor hat weder funktionell noch in seiner Codebasis etwas mit den Visual-Studio-Entwicklungsumgebungen von Microsoft zu tun. Und ja, obwohl VSCode ein Microsoft-Projekt ist, untersteht es einer Open-Source-Lizenz!

Welcher Editor ist nun besser, werden Sie vielleicht fragen. Eine klare Antwort muss ich Ihnen schuldig bleiben. Populärer ist gegenwärtig VSCode. Beide Editoren haben ihren Charme, mit beiden machen Sie nichts verkehrt. Persönlich zieht es mich eher zu Atom, was aber vielleicht nur daran liegt, dass ich dieses Programm zuerst kennengelernt habe. Wenn Sie eine Entscheidungshilfe brauchen, suchen Sie im Internet nach *atom vs vscode*! Sie werden unzählige Vergleichstests finden.

Natürlich habe ich den ersten Teil dieses Kapitels mit Atom, den zweiten dann mit VSCode verfasst. Dabei habe ich jeweils die LaTeX-Erweiterung für den jeweiligen Editor aktiviert. Grundsätzlich war das Schreiben in beiden Editoren problemlos. Aber ich kam nie in Versuchung, von meinem seit 25 Jahren geliebten Editor Emacs Abschied zu nehmen :-)

Alternativen

Das Editor-Angebot hat sich in den vergangenen Jahren erfreulich vergrößert. Ein interessanter Editor mit starkem Fokus auf Webentwickler ist *Adobe Brackets*.

Unter professionellen Entwicklern sehr beliebt ist das kommerzielle Programm *Sublime Text*. Im Gegensatz zu den anderen genannten Programmen basiert es nicht auf Web-Frameworks, sondern wurde in C++ entwickelt und hat damit einen spürbaren Geschwindigkeitsvorteil. Es mangelt nicht an cleveren Funktionen, für die Sie aber aktuell ca. 70 EUR investieren müssen. Immerhin erwerben Sie mit der Lizenz das Recht, den Editor auf allen Ihren Rechnern auszuführen, was dem Linux-Entwickleralltag mit vielen Parallel- und Neuinstallationen entgegenkommt. Sie können Sublime Text unbegrenzt kostenlos testen, aber das Programm wird Sie immer wieder daran erinnern, dass eine Lizenz fällig ist.

Vi und Emacs bleiben wichtig!

Weder die elegante Oberfläche noch die einfache Erweiterbarkeit von Atom, VSCode und Co. machen Vi und Emacs obsolet. Der entscheidende Vorteil dieser Unix-Urgesteine besteht darin, dass Vi und Emacs auch im Textmodus, in Minimalinstallationen und via SSH laufen. Außerdem gehen Vi und selbst Emacs vergleichsweise sparsam mit den Ressourcen um, was man von Atom & Co. leider wirklich nicht behaupten kann.

16.1 Atom

Snap- und Flatpak-Pakete

Unter Ubuntu und Fedora können Sie den Editor direkt als Snap-Paket bzw. als Flatpak-Paket installieren (siehe auch Abschnitt 19.10, »Flatpak und Snap«):

https://snapcraft.io/atom
https://flathub.org/apps/details/io.atom.Atom

Manuelle Installation

Wenn Sie mit anderen Distributionen arbeiten oder einen Widerwillen gegen den riesigen Overhead von Snap bzw. Flatpak haben, müssen Sie eine manuelle Installation durchführen. Auf der Website *https://atom.io* finden Sie RPM- und Debian-Pakete zum Download. Bei den meisten Distributionen wird direkt aus dem Webbrowser heraus ein Paketmanager zur Installation gestartet. Funktioniert das nicht, verwenden Sie `rpm -i paketname` bzw. `dpkg -i paketname`.

Die Installation ohne Paketquelle hat den offensichtlichen Nachteil, dass Sie sich selbst um Updates kümmern müssen – und die gibt es reichlich, typischerweise alle zwei bis drei Monate. Für ein Update beenden Sie Atom, laden das gerade aktuelle RPM- oder Debian-Paket von *https://atom.io* neuerlich herunter und wiederholen die Installation. Keine Angst, Ihre Konfiguration bleibt dabei erhalten. Einstellungen, Tastenkürzel etc. werden im Verzeichnis `.atom` gespeichert.

Atom verfügt über zwei eingebaute Update-Funktionen: Zum einen können Sie mit Edit • Preferences • Updates alle installierten Atom-Erweiterungspakete aktualisieren. Zum anderen gibt es in den Einstellungen im Dialogblatt Core die Option Automatically Update: Unter Windows und macOS sorgt diese Option dafür, dass Atom selbst automatisch aktualisiert wird. Unter Linux funktioniert diese Option aber nicht.

Erste Schritte

Das Atom-Fenster ist beim ersten Start in zwei Teile (*Panes*) geteilt: Links heißt Atom Sie im 21. Jahrhundert willkommen, rechts enthält der Welcome Guide einige Links zu wichtigen Funktionen (siehe Abbildung 16.1). Wenn Sie beide Texte schließen, bleibt das Dokument *untitled* übrig, in dem Sie Atom ausprobieren können. Den Welcome Guide können Sie bei Bedarf über das Help-Menü öffnen.

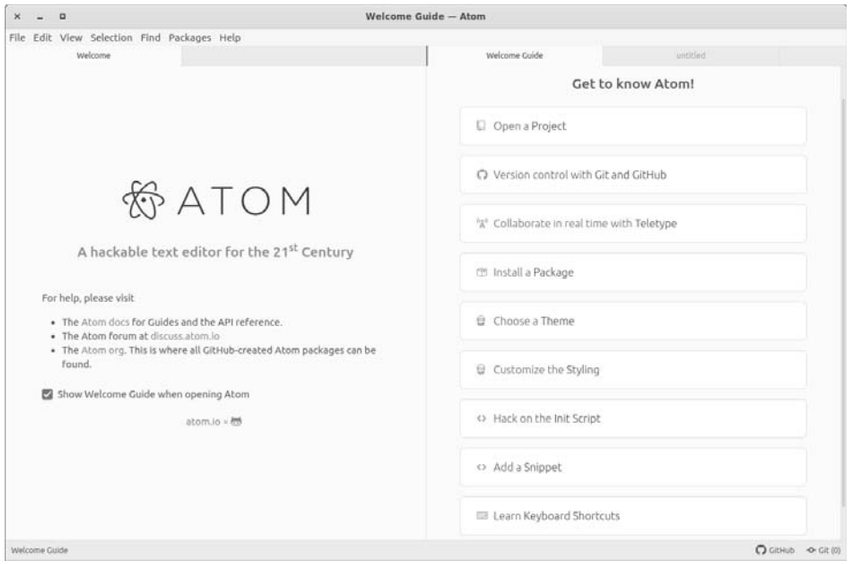

Abbildung 16.1 Die Benutzeroberfläche von Atom

Projekte

Wie in jedem anderen Editor können Sie im File-Menü einzelne Dateien zur Bearbeitung öffnen. Wenn Sie mehrere Dateien bearbeiten möchten, die sich in einem Verzeichnis befinden, ist File • Open Folder die bessere Wahl: Damit wird links der Verzeichnisbaum eingeblendet. Sie können dann per Mausklick zwischen den Dateien des Projekts wechseln. Wenn Ihnen der Verzeichnisbaum zu viel Platz wegnimmt, können Sie ihn mit View • Toggle Tree View ein- und ausblenden.

Aus Atom-Sicht gilt jedes einmal geöffnete Verzeichnis als Projekt. Mit File • Reopen Project wechseln Sie zwischen in der Vergangenheit bearbeiteten Projekten, wobei Atom für jedes Projekt ein neues Fenster öffnet.

Menü und Command View

Die wichtigsten Atom-Kommandos finden Sie wie üblich im Menü. Darüber hinaus gibt es aber unzählige weitere Kommandos, die aus Gründen der Übersichtlichkeit im Menü fehlen. Eine Referenz aller Kommandos gibt die Command View. Das ist ein Auswahldialog, den Sie mit Strg+⇧+P öffnen (siehe Abbildung 16.2). Durch die Eingabe einiger Zeichen können Sie die dort angezeigte Kommandoliste filtern. Mit *delete* als Filterbegriff finden Sie also diverse Kommandos, um Text zu löschen. Soweit die Kommandos mit Tastenkürzeln verbunden sind, werden diese in der Kommandoreferenz ebenfalls angezeigt.

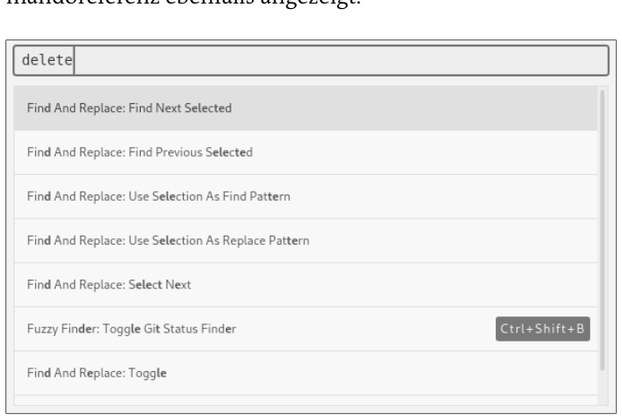

Abbildung 16.2 Auf der Suche nach dem richtigen Kommando

Tabs und Panes

Mehrere zugleich geöffnete Dateien werden als Tabs dargestellt, zwischen denen Sie mit Strg+↹ wechseln können. Atom gliedert das Fenster außerdem in »Panes«, also verschiebbare Fensterbereiche, die wiederum Tabs aufnehmen können. Die Administration der Panes erfolgt wahlweise über Menükommandos (View • Panes), über Tastenkürzel, die zumeist mit Strg+K beginnen, oder über weitere Kommandos, die Sie mit Strg+⇧+P und dem Suchbegriff *panes* in der Kommandoreferenz finden.

Grundeinstellungen

Light oder Dark Theme

Nach der Installation zeigt sich Atom von seiner dunklen Seite, d. h., Text wird in weißer/heller Schrift auf schwarzem Hintergrund angezeigt. Das mag modern sein, ist tagsüber in einem hellen Büro aber nicht ergonomisch. Abhilfe: Führen Sie Edit • Preferences aus, wechseln Sie in das Dialogblatt Themes, und stellen Sie dort eines der Light Themes für die Oberfläche und für das Syntax-Highlighting ein. Alle Bildschirmabbildungen in diesem Kapitel wurden der besseren Lesbarkeit wegen im Light Theme erstellt.

Line Wrapping

Die Line-Wrapping-Einstellungen steuern, wie Atom mit zu langen Zeilen umgeht. Standardmäßig werden diese am Fensterrand einfach abgeschnitten; Sie müssen also den Fensterinhalt nach links schieben, wenn Sie das Ende einer langen Zeile sehen möchten. Mit der Option Soft Wrap im Dialogblatt Edit • Preferences • Editor erreichen Sie, dass derartige Zeilen am Bildschirm (nicht aber in der resultierenden Datei) umbrochen werden. Die Einstellung Soft Wrap Hanging Indent gibt an, wie weit der umbrochene Text eingerückt werden soll (relativ zur Einrückung des Zeilenbeginns).

Wenn Sie mit Atom Fließtext und nicht Programmcode verfassen, dann ist es oft wünschenswert, dass Atom den Text wirklich über mehrere Zeilen verteilt (»Hard Wrap«). Das gelingt manuell mit Edit • Reflow Selection bzw. Strg+⇧+Q. Wenn Sie diese Funktion oft brauchen, sollten Sie ihr ein anderes Tastenkürzel zuweisen; andernfalls ist die Gefahr groß, dass Sie Atom versehentlich beenden.

Ich habe keine Möglichkeit gefunden, Atom so einzustellen, dass er lange Zeilen bereits bei der Eingabe umbricht (*Hard Wrap*).

Rechtschreibkontrolle

Standardmäßig kennzeichnet Atom falsch geschriebene Wörter. Je nach Text funktioniert das nicht immer optimal. Um die Rechtschreibkontrolle abzustellen, suchen Sie in Edit • Preferences • Packages nach dem Paket `spell-check` und deaktivieren es.

Autocomplete

Die Autocomplete-Funktion schlägt nach der Eingabe einiger Buchstaben passende Vervollständigungen vor, wobei Wörter aus allen geladenen Dateien berücksichtigt werden. Für diverse Programmiersprachen gibt es spezielle Autocomplete-Pakete, die diese Funktion weiter optimieren.

Tabulatoren

Im Dialogblatt Edit • Preferences • Editor können Sie im Listenfeld Tabtype einstellen, ob ↹ einen Tabulatorcode (hard) oder mehrere Leerzeichen (soft) in Ihre Datei einfügen soll. Die Defaulteinstellung lautet auto: Je nachdem, ob bei der ersten eingerückten Zeile ein Leerzeichen oder ein Tabulatorzeichen verwendet wird, passt sich Atom entsprechend für den Rest der Datei an.

Paketverwaltung

In Atom ist der Großteil aller Funktionen in Form von Paketen realisiert. Standardmäßig sind in Atom bereits rund 80 sogenannte Core-Pakete aktiv. Diese Pakete kümmern sich z. B. um das Syntaxhighlighting in diversen Sprachen, um die automatische Erstellung von Backups aller offenen Dateien oder um die vorhin beschriebene Autocomplete-Funktion.

Einen Überblick über alle installierten Pakete gibt Edit • Preferences • Packages. In diesem Dialog können Sie Pakete auch konfigurieren, vorübergehend deaktivieren oder ganz entfernen (siehe Abbildung 16.3). Zur Suche nach neuen Paketen sowie zu deren Installation wechseln Sie in das Dialogblatt Install. Dort haben Sie aktuell die Wahl zwischen mehr als 5000 Atom-Erweiterungen! Im Dialogblatt Updates können Sie die installierten Pakete auf den neuesten Stand bringen.

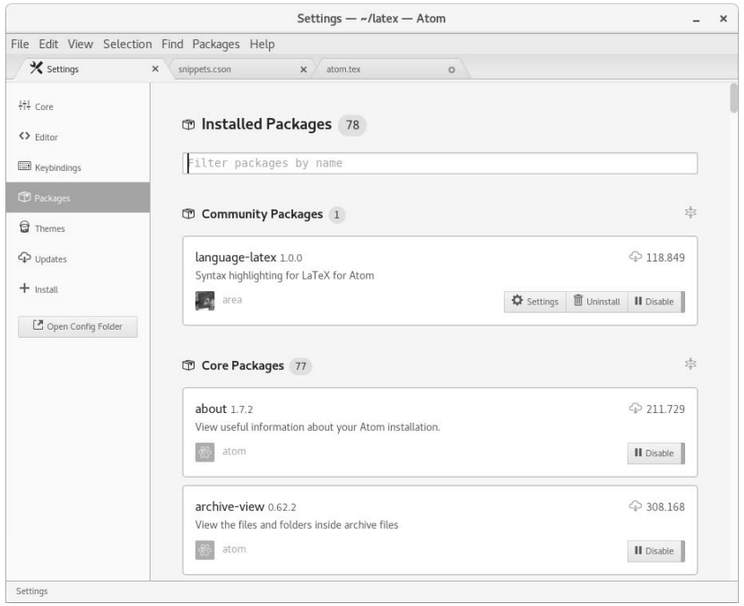

Abbildung 16.3 Atom verfügt über eine eigene Paketverwaltung.

Manche Pakete erweitern das Atom-Menü. Paketspezifische Kommandos finden Sie unter Packages • Paketname.

apm-Kommando

Interessanterweise können Sie die Paketverwaltung auch außerhalb von Atom mit dem Kommando `apm` durchführen. `apm list` zählt alle installierten Pakete auf, `apm install name` installiert das gewünschte Paket etc. Das funktioniert auch, wenn Atom aktuell nicht läuft. `atom help` gibt eine kurze Beschreibung des Kommandos, und `atom help subcmd` liefert ausführlichere Informationen zu einem Subkommando.

Tastatur

keymap.cson

Es ist in Atom relativ einfach, eigene Tastenkürzel zu definieren. Dazu öffnen Sie mit Edit • Keymap die Datei `.atom/keymap.cson` und fügen dort Anweisungen entsprechend dem folgenden Muster ein:

```
'atom-text-editor':
   'ctrl-a': 'editor:move-to-beginning-of-line'
   'ctrl-e': 'editor:move-to-end-of-screen-line'
```

Im Detail ist die Keymap-Syntax hier beschrieben:

https://flight-manual.atom.io/behind-atom/sections/keymaps-in-depth

Durchgeführte Änderungen werden sofort wirksam, sobald Sie die Datei speichern. (Wenn `keymap.cson` bisher nicht existierte, muss Atom einmalig neu gestartet werden, damit die Tastenkürzel berücksichtigt werden.) Die Einstellungen haben Vorrang gegenüber den Tastenkürzeln, die in Paketen definiert sind. Atom passt übrigens die Menüeinträge an die gerade geltenden Tastenkürzel an!

Bei der Suche nach den Namen der auszuführenden Atom-Kommandos hilft einerseits das Keybindings-Dialogblatt der Settings und andererseits die Kommandopalette, die Sie mit Strg+⇧+P öffnen.

Emacs-Tastenkürzel

Für Emacs-Fans gibt es gleich mehrere Pakete, die Emacs-Tastenkürzel unter Atom nachbilden. Am populärsten ist `atomic-emacs`, weniger umfassend (im positiven Sinne) ist `emacs-core-keys`.

snippets.cson

Mit Edit • Snippets gelangen Sie in die Datei `.atom/snippets.cson`. In dieser Datei können Sie Textbausteine definieren. Die folgenden Zeilen geben zwei Beispiele für derartige Bausteine:

```
'.text':
  'mfg':
    'prefix': 'mfg'
    'body': 'Mit freundlichen Grüßen,\n\n          Michael Kofler'

'.text.tex.latex':
  'latex vl':
    'prefix': 'vl'
    'body': '\\\\begin{verbatim}\n\n\\\\end{verbatim}'
```

Der erste Baustein gilt in allen Textdateien, aber nicht in Codedateien. Wenn Sie `mfg` und dann ↹ eingeben, macht Atom daraus die Floskel *Mit freundlichen Grüßen*. Der zweite Baustein gilt nur in LaTeX-Dateien und hilft bei der Eingabe eines Codeblocks. Die vierfachen Backslashes sind notwendig, damit ein Backslash korrekt in den Text eingefügt wird.

Markdown-Dateien bearbeiten

Das für mich seit etlichen Jahren wichtigste Textformat ist Markdown (siehe auch Abschnitt 12.5, »Markdown und Pandoc«). Atom kommt auf Anhieb mit `*.md`-Dateien zurecht. Die Tastenkombination Strg+⇧+M öffnet neben dem Text in einer neuen Pane eine Preview-Ansicht, die für die meisten Zwecke ausreicht (siehe Abbildung 16.4). Damit übertrifft Atom quasi aus dem Stand und ohne jede Konfigurationsarbeit den Funktionsumfang vieler Markdown-Editoren.

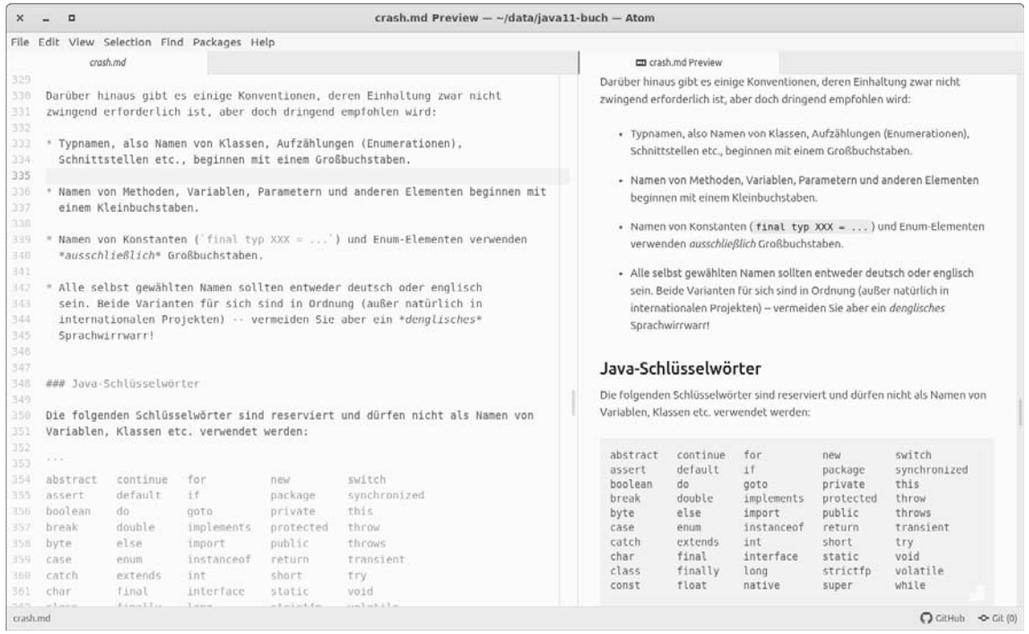

Abbildung 16.4 Links ein Markdown-Text, rechts die Vorschau

Pandoc-Integration

Standardmäßig geht die Markdown-Preview davon aus, dass Sie Markdown in der GitHub-Variante nutzen. Wenn Sie die erweiterten Formatierungsmöglichkeiten von Pandoc nutzen, müssen Sie das standardmäßig aktive Package `markdown-preview` deaktivieren (Button DISABLE) und stattdessen das Paket `markdown-preview-plus` installieren. Die Tastenkombination für die Vorschau bleibt mit Strg+⇧+M unverändert.

Der entscheidender Vorteil von `markdown-preview-plus` besteht darin, dass es anstatt der simplen GitHub-Markdown-Engine auch Pandoc aufrufen kann. Dazu müssen Sie einige Optionen des Packages konfigurieren:

- Als RENDERER BACKEND stellen Sie PANDOC ein.
- Am Beginn der Optionsgruppe PANDOC SETTINGS geben Sie den Pfad zum `pandoc`-Kommando ein, also das Ergebnis von `which pandoc`.
- Mit PANDOC SETTINGS • COMMANDLINE ARGUMENTS können Sie die Optionen angeben, die beim Aufruf an `pandoc` übergeben werden sollen. (Gescheitert bin ich allerdings beim Versuch, mit den Optionen `-s -c my.css` eine eigene CSS-Datei anzugeben.)
- Mit PANDOC SETTINGS • MARKDOWN FLAVOR geben Sie an, welche Markdown-Variante `pandoc` verwenden soll. Standardmäßig gilt `markdown-raw_tex+tex_math_single_backslash`. Oft reicht einfach `markdown`, also der Standard-Markdown-Dialekt von Pandoc.

16.2 VSCode

Installation und Updates

VSCode kann wie Atom unter Ubuntu als Snap-Paket installiert werden, unter Fedora als Flatpak-Paket:

https://snapcraft.io/vscode
https://flathub.org/apps/details/com.visualstudio.code

Für die manuelle Installation gibt es RPM- und Debian-Pakete auf der Projektwebsite zum Download:

https://code.visualstudio.com

Bei einer manuellen Installation müssen Sie sich um eventuell veröffentlichte Updates selbst kümmern und gegebenenfalls den Download und die Installation wiederholen.

Erste Schritte

Seitenleiste

In der Seitenleiste von VSCode können Sie über fünf Icons einen Verzeichnis-Browser, die Suchfunktion, ein Git-Menü, einen Debugger sowie die Extension-Verwaltung ein- und durch nochmaliges Anklicken wieder ausblenden. Für diese Aktionen gibt es jeweils auch Strg+⇧-Tastenkombinationen (siehe das VIEW-Menü).

Tabs und Gruppen

Mehrere offene Dateien werden in Form von Dialogblättern (*Tabs*) angezeigt (siehe Abbildung 16.5). Außerdem können Sie den Editor vertikal in mehrere Gruppen unterteilen, z. B., um mehrere Dateien (oder Ausschnitte derselben Datei) nebeneinander darzustellen. Mit Strg+↹ wechseln Sie zwischen den Tabs der aktiven Gruppen, mit Strg+1, Strg+2 etc. zwischen den Gruppen. Weitere Kommandos zum Wechseln zwischen Dateien und Gruppen finden Sie im Menü GO.

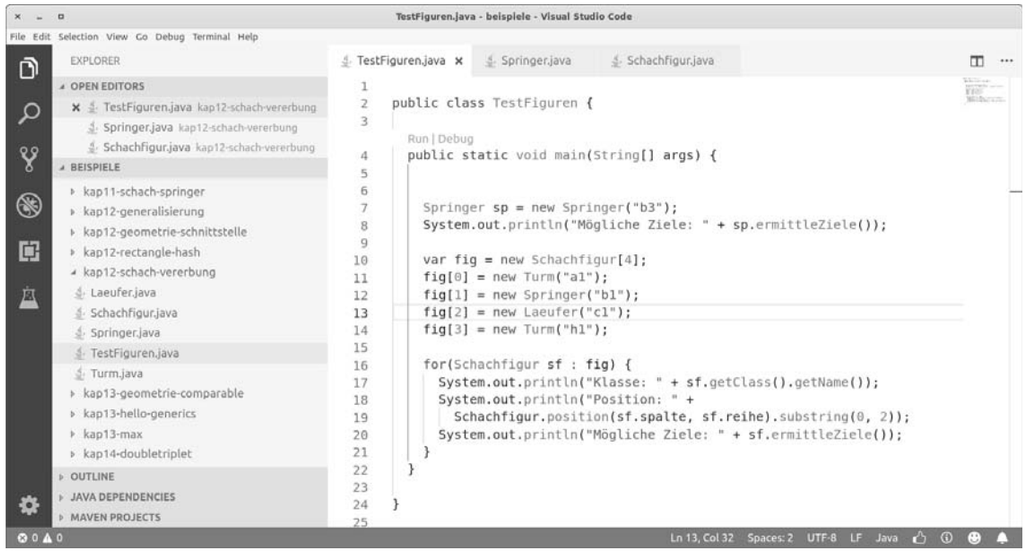

Abbildung 16.5 VSCode mit Seitenleiste und Tabs für mehrere geöffnete Java-Dateien

Menüs und Kommandos

VSCode verwendet wie Atom die Tastenkombination Strg+⇧+P zum Aufruf eines kleinen Dialogs, in dem Sie alle verfügbaren Kommandos suchen und ausführen können.

Ordner

Wenn Sie File • Open Folder ausführen, entspricht dies einem Projektwechsel. VSCode schließt die aktuell geöffneten Dateien und wechselt in das neue Verzeichnis. Wenn Sie parallel Dateien aus mehreren Verzeichnissen bearbeiten möchten, müssen Sie vorher mit Strg+⇧+N ein weiteres VSCode-Fenster öffnen.

Zen-Mode

Mit Strg+K,Z aktivieren Sie den Zen-Modus. In diesem Modus wechselt VSCode in den Vollbildmodus und zeigt nur den aktuellen Text an, um jede Ablenkung zu vermeiden. Zur Rückkehr in den Standardmodus drücken Sie zweimal Esc.

Terminal

VSCode enthält ein integriertes Terminal, das Sie mit View • Terminal öffnen. Standardmäßig ist im Terminal das Verzeichnis aktiv, in dem sich die aktuell bearbeitete Datei befindet.

Grundeinstellungen

Zur Veränderung der Grundeinstellungen führen Sie File • Preferences • Settings aus. VSCode differenziert dabei zwischen User Settings und Workspace Settings und zeigt die auf den ersten Blick identischen Einstellungen in zwei Dialogblättern an.

Die User Settings gelten generell für VSCode, während Workspace Settings nur für das aktuelle Projekt gelten (also für die Dateien des aktuellen Verzeichnisses).

Hinter den Kulissen werden die Einstellungen in `.config/Code/User/settings.json` (relativ zum Heimatverzeichnis) bzw. in `.vscode/settings.json` (relativ zum Projektverzeichnis) gespeichert.

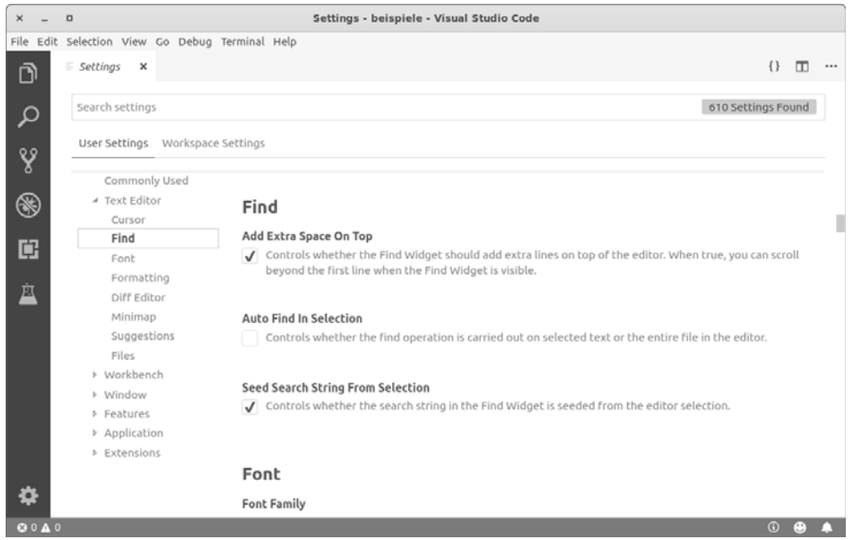

Abbildung 16.6 VSCode-Konfiguration

Extensions

VSCode ist durch sogenannte Extensions erweiterbar. View • Extensions zeigt eine Liste aller aktuell installierten Extensions an und bietet die Möglichkeit, nach weiteren Extensions zu suchen und diese zu installieren. In bester Microsoft-Manier erfordern viele Extensions nach der Installation einen Neustart des Editors, um wirksam zu werden. Nach geeigneten Extensions können Sie auch auf der folgenden Webseite suchen:

https://marketplace.visualstudio.com/VSCode

Line Wrapping

VSCode kann wie die meisten anderen Editoren zu lange Zeilen automatisch umbrechen (Option Editor • Word Wrap). Es gibt aber erstaunlicherweise keine Funktion für einen *Hard Wrap*, also für das Einfügen von echten Zeilenumbrüchen in mehrzeilige Texte oder Kommentare. Abhilfe schafft die Extension `rewrap`.

Light Theme

Wie Atom verwendet VSCode standardmäßig dunkle Farben. Wenn Sie ein helleres Erscheinungsbild vorziehen, wählen Sie mit File • Preferences • Color Theme ein anderes Farbschema aus.

Sprache

VSCode zeigt Menüs und Dialoge standardmäßig in englischer Sprache an. Um die deutsche Lokalisierung zu aktivieren, führen Sie View • Extensions aus und suchen nach dem *German Language Pack for VS Code*, installieren dieses und starten VSCode dann neu.

Persönlich habe ich Zweifel, ob lokalisierte Menüs bei einem Editor für Programmentwickler eine gute Idee sind. Ich bin deswegen für diesen Abschnitt bei den englischen Menüs geblieben.

Konfiguration der Tastatur und andere Eingabeerleichterungen

Tastenkürzel

Zur Veränderung von Tastenkürzeln führen Sie File • Preferences • Keyboard Shortcuts aus. Nun verwenden Sie die Suchfunktion, um den Namen der gewünschten Aktion zu ermitteln, und stellen dann das gewünschte Tastenkürzel ein (siehe Abbildung 16.7). Die Einstellungen werden in `.config/Code/User/keybindings.json` gespeichert.

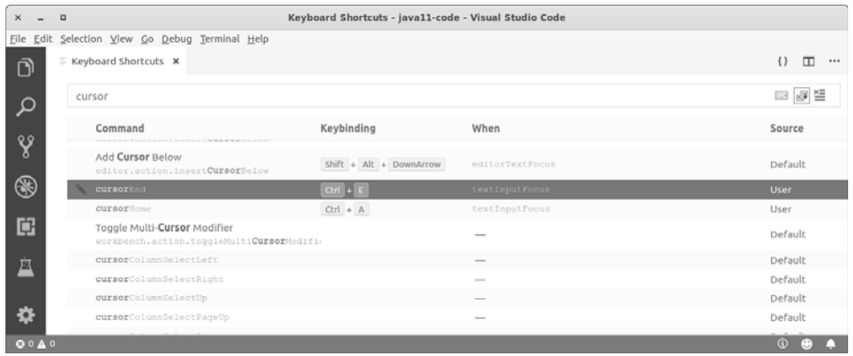

Abbildung 16.7 Eigene Tastenkürzel einrichten

Keymaps anderer Editoren

File • Preferences • Keymaps hilft bei der Installation von Extensions, die die Keymaps anderer Editoren nachbilden (Atom, Eclipse, Emacs, Sublime, Vim etc.).

IntelliSense

Die automatische Vervollständigung von (Schlüssel-)Wörtern heißt in Microsoft-Produkten schon seit Jahrzehnten »IntelliSense«. Besonders zielsicher sind die IntelliSense-Vorschläge, wenn Sie eine Extension mit spezifischen VSCode-Erweiterungen für die Programmiersprache Ihrer Wahl installieren.

Snippets

Sie können häufig benötigte Codestrukturen oder Texte mit einer Abkürzung versehen. Dazu führen Sie File • Preferences • User Snippets aus und wählen eine Sprache oder ein Textformat. Damit gelangen Sie in eine Snippet-Datei für die jeweilige Sprache. Alternativ können Sie auch eine globale Datei mit Abkürzungen einrichten.

Anschließend definieren Sie wie im folgenden Beispiel für die Programmiersprache Java Codeblöcke samt einer dazugehörigen Abkürzung (hier `syso`):

```
{
  "println": {
    "prefix": "syso",
    "body": [ "System.out.println();" ],
    "description": "Print to console"
  }
}
```

Wenn Sie in einer Java-Datei nun `syso` eingeben, schlägt VSCode als Vervollständigung den Textblock (`body`) vor.

Emmets

VSCode unterstützt auch Emmets. Das ist eine spezielle Syntax zur Eingabe von HTML- und CSS-Code. Wenn Sie in einer HTML-Datei beispielsweise `nav>ul>li` und dann ⇥ eingeben, macht VSCode daraus den folgenden Codeblock:

```
<nav>
  <ul>
    <li></li>
  </ul>
</nav>
```

Eine Menge Beispiele für die Emmet-Syntax finden Sie hier:

https://docs.emmet.io/cheat-sheet

Kapitel 35
VirtualBox und Vagrant

Virtualisierung macht es möglich, auf einem Rechner mehrere Betriebssysteme parallel auszuführen. Daraus ergeben sich unzählige Anwendungen: Sie können Linux unter Windows ausprobieren oder Windows unter Linux ausführen, eine neue Alpha-Version der Distribution xyz gefahrlos testen, ohne die vorhandene Linux-Installation zu gefährden, Server-Funktionen sicher voneinander trennen etc.

Das für die Plattformen Windows, Linux und macOS verfügbare Programm VirtualBox eignet sich am besten zur Desktop-Virtualisierung, also zur Ausführung von virtuellen Maschinen, die im Grafiksystem bedient werden sollen. Hinter VirtualBox stand ursprünglich die deutsche Firma InnoTek. 2008 übernahm Sun InnoTek, und 2010 kaufte Oracle Sun. Damit ist nun Oracle der Eigentümer von VirtualBox. Umfassende Dokumentation zu VirtualBox finden Sie unter:

https://www.virtualbox.org

Große Teile von VirtualBox bestehen aus Open-Source-Code. Die einzige Ausnahme sind einige Zusatzfunktionen, die extra installiert werden müssen. Ihre Nutzung ist für Privatanwender ebenfalls kostenlos, für kommerzielle Anwender hingegen kostenpflichtig.

Dieses Kapitel beschreibt, wie Sie VirtualBox unter Linux installieren und darin virtuelle Maschinen ausführen. Mit Einschränkungen ist VirtualBox auch zur Server-Virtualisierung geeignet. Für diesen Zweck ist das Programm KVM, das ich Ihnen in Kapitel 36 näher vorstelle, wesentlich besser geeignet.

Vagrant

Das Einrichten neuer virtueller Maschinen ist mit Arbeit verbunden. Mit Vagrant können Sie diesen Prozess automatisieren. Das ist vor allem dann praktisch, wenn Sie reproduzierbar Testumgebungen aufsetzen möchten, z. B. für eine bestimmte Server-Konfiguration. Vagrant kann derartige Aufgaben auch für andere Virtualisierungssysteme erledigen. VirtualBox eignet sich aber besonders gut, um Vagrant kennenzulernen.

35.1 VirtualBox installieren

Zur Installation von VirtualBox gibt es grundsätzlich zwei Möglichkeiten. Die bequeme Variante besteht darin, einfach die VirtualBox-Pakete zu verwenden, die sich in den Paketquellen Ihrer Distribution befinden. Sollten diese Pakete fehlen oder nicht ausreichend aktuell sein, können Sie VirtualBox selbst von der Website *https://www.virtualbox.org* herunterladen und manuell installieren. Das ist ein wenig umständlicher.

In diesem Abschnitt gehe ich auf beide Varianten ein. Losgelöst davon sind nach Abschluss der Installation noch einige Vorbereitungsarbeiten zu erledigen, die ich am Ende dieses Abschnitts erläutere.

Host und Gast

Bei der Beschreibung von Virtualisierungssystemen hat es sich eingebürgert, das Grundsystem als Wirt (*Host*) und die darauf laufenden virtuellen Maschinen als Gäste (*Guests*) zu bezeichnen. In diesem Kapitel gehe ich davon aus, dass der Host ein bereits funktionierendes Linux-System ist.

VirtualBox-Pakete Ihrer Distribution

VirtualBox unter Linux installieren

Die meisten Distributionen bieten fertige VirtualBox-Pakete an. Bei Fedora müssen Sie vorher die `rpmfusion`-Paketquelle aktivieren. Bei openSUSE befinden sich die Kernfunktionen und die Benutzeroberfläche in getrennten Paketen; dort müssen Sie auch das Paket `virtualbox-qt` installieren.

Nicht erforderlich sind hingegen die diversen `virtualbox-guest`-Pakete! Diese Pakete enthalten Treiber, die *in* virtuellen Maschinen auszuführen sind, also wenn eine Linux-Distribution selbst innerhalb von VirtualBox ausgeführt werden soll.

VirtualBox-Kernelmodule

VirtualBox greift auf dem Wirtssystem auf die vier Kernelmodule `vboxdrv`, `vboxpci`, `vboxnetadp` und `vboxnetflt` zurück. Manche Distributionen stellen diese Module in binärer Form durch ein eigenes Paket zur Verfügung, das bei jedem Kernel-Update aktualisiert wird. Bei openSUSE lautet der Paketname `virtualbox-host-kmp-default`.

DKMS

Bei anderen Distributionen wird der Quellcode der VirtualBox-Pakete installiert. Bei jedem Kernel-Update müssen die entsprechenden VirtualBox-Module neu kompiliert werden. Darum kümmert sich bei einigen Distributionen DKMS (*Dynamic Kernel Module Support*). Dies ist z. B. bei Ubuntu der Fall, wo Sie das Paket `virtualbox-dkms` installieren müssen.

akmods (Fedora)

Die RPMFusion-Paketquelle für Fedora sieht anstelle von DKMS das Kommando `akmods` vor. Falls Sie gerade ein Kernel-Update durchgeführt haben, starten Sie den Rechner zuerst neu. Anschließend aktivieren Sie die RPMFusion-Paketquellen und führen die folgenden Kommandos aus:

```
root#  dnf install VirtualBox akmod-VirtualBox
root#  akmods
root#  systemctl restart systemd-modules-load
```


Das erste Kommando installiert die erforderlichen Pakete, das zweite kompiliert die Kernelmodule, das dritte lädt sie. Von nun an soll sich `akmods` nach jedem Kernel-Update selbstständig um die Aktualisierung der VirtualBox-Treiber kümmern. Bei meinen Tests hat das häufig nicht funktioniert. Sie müssen dann die obigen drei Kommandos neuerlich ausführen.

Manuell kompilieren

Steht weder DKMS noch `akmods` zur Verfügung, können Sie die Module bei manchen Distributionen durch ein Script manuell kompilieren:

```
root#  /usr/lib/virtualbox/vboxdrv.sh setup
```

Zum Kompilieren sind aber auch der C-Compiler `gcc` sowie die Kernel-Header-Dateien erforderlich. Bei vielen Distributionen müssen Sie die entsprechenden Pakete vorher installieren (siehe Abschnitt 24.3, »Kernelmodule selbst kompilieren«).

Test

Ob das Kompilieren und Laden der VirtualBox-Kernelmodule funktioniert hat, prüfen Sie mit dem folgenden Kommando:

```
root#  lsmod | grep vbox
vboxpci                24576  0
vboxnetadp             28672  0
vboxnetflt             28672  0
vboxdrv               434176  3 vboxnetadp,vboxnetflt,vboxpci
```

VirtualBox-Pakete von Oracle

Statt der mit Ihrer Distribution mitgelieferten VirtualBox-Pakete können Sie auch die von Oracle zum Download angebotene Version installieren. Das ist vor allem dann zweckmäßig, wenn Oracle eine neuere VirtualBox-Version anbietet als Ihre Distribution.

https://www.virtualbox.org/wiki/Linux_Downloads

Auf der obigen Website finden Sie VirtualBox in verschiedenen Formaten: als RPM- und Debian-Paket für diverse Distributionen sowie als Universal-Installer, den Sie wie folgt starten:

```
root#  chmod u+x VirtualBox_nnn.run install
root#  ./VirtualBox_nnn.run install
```

Kernelmodule

Nach Möglichkeit sollten Sie vor VirtualBox das `dkms`-Paket Ihrer Distribution installieren. In diesem Fall verwaltet DKMS die VirtualBox-Kernelmodule und kümmert sich bei Kernel-Updates automatisch um eine Neukompilierung. Bei meinen VirtualBox-Installationen hat das allerdings nicht immer zuverlässig funktioniert.

Wenn DKMS nicht zur Verfügung steht bzw. versagt, kompilieren Sie die Kernelmodule selbst:

```
root#  /usr/lib/virtualbox/vboxdrv.sh setup
```

Wie vorhin schon erwähnt, müssen Sie gegebenenfalls vorher den C-Compiler und die Kernel-Header-Dateien oder den Kernel-Quellcode installieren:

APT-Paketquelle

Für Debian- und Ubuntu-Anwender gibt es eine eigene APT-Paketquelle. Gegenüber der manuellen Installation eines einzelnen Pakets hat die Paketquelle den Vorteil, dass Sie innerhalb der gewählten Major-Version automatisch Updates erhalten. Dazu fügen Sie zu `/etc/apt/sources.list` eine der folgenden Zeilen hinzu:

```
deb https://download.virtualbox.org/virtualbox/debian stretch contrib
deb https://download.virtualbox.org/virtualbox/debian disco   contrib
```

Anstelle von `stretch` bzw. `disco` müssen Sie den Codenamen der von Ihnen eingesetzten Debian- bzw. Ubuntu-Distribution verwenden. Werfen Sie gegebenenfalls einen Blick in die Datei `/etc/os-release`.

Außerdem führen Sie diese beiden Kommandos aus, um den Schlüssel der Paketquelle zu installieren:

```
root#  wget -q https://www.virtualbox.org/download/oracle_vbox_2016.asc
root#  apt-key add oracle_vbox_2016.asc
```

Anschließend installieren Sie VirtualBox mit `apt` oder `apt-get`, wobei Sie die gerade aktuelle VirtualBox-Versionsnummer angeben:

```
root#  apt update
root#  apt install virtualbox-6.0
```

Yum-Paketquelle

Für Anwender von Yum-kompatiblen Distributionen (CentOS, Fedora, openSUSE, Red Hat etc.) gibt es analog eine Yum-Paketquelle. Auch in diesem Fall müssen Sie zuerst den Schlüssel importieren:

```
root#  wget -q https://www.virtualbox.org/download/oracle_vbox.asc
root#  rpm --import oracle_vbox.asc
```

Danach laden Sie die für Ihre Distribution passende `*.repo`-Datei von der VirtualBox-Download-Seite herunter und kopieren sie in das Verzeichnis `/etc/yum.repos.d`. Die folgenden Zeilen zeigen die Fedora-Variante der `*.repo`-Datei:


```
# Datei /etc/yum.repos.d/virtualbox.repo
[virtualbox]
name=Fedora $releasever - $basearch - VirtualBox
baseurl=http://download.virtualbox.org/virtualbox/rpm/fedora/$releasever/$basearch
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://www.virtualbox.org/download/oracle_vbox.asc
```

Die VirtualBox-Installation führen Sie nun mit `dnf install` oder `yum install` oder `zypper install` durch.

Vorbereitungsarbeiten

Speicherort für virtuelle Maschinen

VirtualBox richtet für jede virtuelle Maschine ein Unterverzeichnis innerhalb von `VirtualBox VMs` ein. In mehreren Dateien werden dort die Einstellungen der virtuellen Maschine sowie die virtuelle Festplatte gespeichert. Mit Datei • Globale Einstellungen können Sie gegebenenfalls einen anderen Speicherort einstellen.

Extension Pack

Oracle bietet auf seiner Website ein sogenanntes Extension Pack zum Download an. Beim Download des Extension Packs schlägt der Webbrowser vor, die Datei direkt mit VirtualBox zu öffnen. Diesem Vorschlag folgen Sie einfach.

Das Extension Pack ergänzt VirtualBox um einige Zusatzfunktionen: Unter anderem können Sie dann in den virtuellen Maschinen auf USB-Geräte (USB 2 und USB 3), PCI-Karten und Webcams zugreifen und die virtuellen Maschinen via RDP (Remote Display Protocol) auf einem anderen Rechner im Netzwerk steuern. Diese Erweiterungen werden nur in Binärform vertrieben, es handelt sich also nicht um Open-Source-Code. Die kommerzielle Nutzung dieser Erweiterungen erfordert eine Lizenz von Oracle!

vboxusers-Gruppe

Unabhängig davon, aus welcher Quelle Ihre VirtualBox-Installation stammt, wurde die Gruppe `vboxusers` eingerichtet. Nur Benutzer, die dieser Gruppe angehören, können in virtuellen Maschinen auf USB-Geräte zugreifen. Deswegen müssen Sie vor dem ersten Start von VirtualBox Ihren Account der Gruppe `vboxusers` hinzufügen. Ersetzen Sie beim folgenden Kommando `kofler` durch Ihren Login-Namen:

```
root#  usermod -a -G vboxusers kofler
```

Damit die geänderte Gruppenzuordnung wirksam wird, müssen Sie sich aus- und neu einloggen. Anschließend starten Sie die Benutzeroberfläche von VirtualBox über das KDE- oder Gnome-Menü bzw. mit dem Kommando `VirtualBox`.

VirtualBox unter Windows oder macOS installieren

Die Installation von VirtualBox unter Windows oder macOS ist grundsätzlich ein Kinderspiel: Sie laden das passende Setup-Programm von der VirtualBox-Seite herunter und führen es aus. Das Extension Pack muss auch in diesem Fall extra heruntergeladen und eingerichtet werden.

Unter Windows kann es allerdings passieren, dass VirtualBox nicht richtig funktioniert: Virtuelle Maschinen lassen sich dann nur im 32-Bit-Modus einrichten oder können gar nicht gestartet werden. Der Grund dafür ist in der Regel, dass VirtualBox die Virtualisierungstechnik VT nicht nutzen kann, die in viele Intel-CPUs integriert ist.

Dafür kann es mehrere Ursachen geben: Am wahrscheinlichsten ist es, dass Windows die Funktion durch Hyper-V blockiert. Abhilfe: Starten Sie das Programm Windows-Features, suchen Sie nach Hyper-V und deaktivieren Sie die Option. Danach muss der Rechner neu gestartet werden. Sollte Hyper-V nicht schuld sein, ist VT möglicherweise im BIOS/EFI deaktiviert. Bei modernen Computern ist es ziemlich unwahrscheinlich, dass Ihre CPU die Funktion wirklich nicht enthält.

35.2 VirtualBox-Maschinen einrichten

Ist VirtualBox einmal installiert, können Sie mit dem Einrichten virtueller Maschinen beginnen. Dieser Abschnitt berücksichtigt sowohl Linux- als auch Windows-Gäste.

Eine virtuelle Maschine mit Linux einrichten

Dieser Abschnitt beschreibt, wie Sie innerhalb von VirtualBox eine virtuelle Maschine mit Linux einrichten. Dabei spielt es keine Rolle, ob VirtualBox selbst unter Linux, Windows oder macOS läuft.

Beim Einrichten einer neuen virtuellen Maschine unterstützt Sie ein Assistent. Als Betriebssystemtyp stehen neben Windows diverse Linux-Distributionen zur Auswahl. Wenn Ihre Distribution nicht vertreten ist, wählen Sie Linux mit Kernel 2.6 / 3.x / 4.x; diese Einstellung gilt für alle aktuellen Kernelversionen, auch für 5.x. Achten Sie darauf, dass es für jedes Betriebssystem *zwei* Versionen gibt: eine für 32- und eine für 64-Bit-Installationen. Wählen Sie den passenden Eintrag!

VirtualBox sieht standardmäßig 1 GiB RAM für virtuelle Linux-Maschinen vor. Viele Desktop-Distributionen laufen flüssiger, wenn Sie ihr etwas mehr RAM zuweisen.

Als Nächstes müssen Sie eine virtuelle Festplatte einrichten. Der Datenträger wird als Image-Datei im Host-Dateisystem gespeichert. Dazu stehen verschiedene Formate zur Auswahl. Im Regelfall sollten Sie beim VirtualBox-eigenen Format VDI bleiben und auch die Option dynamisch alloziert beibehalten. Damit wird der Speicherplatz für die Festplatte erst nach und nach angefordert. Die Alternative feste Grösse bedeutet, dass der gesamte Speicherplatz sofort vorreserviert wird.

Die vorgeschlagenen 10 GiB sind allerdings arg knapp bemessen. Bei vielen Distributionen reicht das nicht einmal für eine Minimalinstallation aus. Stellen Sie zumindest 20 GiB ein.

Schließlich zeigt VirtualBox eine Zusammenfassung aller Hardware-Komponenten an. Mit Ändern können Sie nun bei Bedarf weitere Einstellungen durchführen, z. B. den Netzwerkzugang verändern oder im Dialogblatt Massenspeicher eine ISO-Datei als Datenquelle für das DVD-Laufwerk auswählen.

Wenn Sie mit der Konfiguration fertig sind, starten Sie die virtuelle Maschine. Das von der ISO-Datei geladene Linux-Installationsprogramm erscheint in einem eigenen Fenster. Dort installieren Sie Linux wie auf einem realen Rechner.

Mögliche Fehlermeldungen beim ersten Start einer virtuellen Maschine

VirtualBox testet erst mit dem Start einer virtuellen Maschine, ob die VirtualBox-Kernelmodule geladen sind und ob Hardware-Virtualisierungsfunktionen zur Verfügung stehen. Ist eine dieser Voraussetzungen nicht erfüllt, wird eine Fehlermeldung oder Warnung angezeigt.

Bei den Kernelmodulen müssen Sie sicherstellen, dass diese installiert sind. Falls Sie VirtualBox manuell von *https://www.virtualbox.org* installiert haben, hilft es oft, das Script `/usr/lib[64]/virtualbox/vboxdrv.sh setup` zum Neukompilieren der Module auszuführen. Denken Sie auch daran, dass die Hardware-Virtualisierungsfunktionen im BIOS oder EFI aktiviert sein müssen.

Eine weitere mögliche Fehlerursache kann darin bestehen, dass ein anderes Virtualisierungssystem läuft und die Virtualisierungsfunktionen der CPU blockiert (z. B. KVM/libvirt).

Host-Tastenkombination

Die virtuelle Maschine erhält automatisch den Tastatur- und Mausfokus, sobald Sie eine Taste drücken. Standardmäßig lösen Sie den Fokus mit der rechten Strg-Taste. Im VirtualBox-Hauptfenster können Sie mit Datei • Einstellungen • Eingabe • Virtuelle Maschine eine andere »Host«-Taste einstellen. Die gerade gültige Kombination wird rechts in der Statusleiste des VirtualBox-Fensters angezeigt. Die wichtigsten Host-Tastenkombinationen sind in Tabelle 35.1 zusammengefasst.

Tastenkürzel	Bedeutung
Host	Tastatur- und Mausfokus lösen
Host + F	Vollbildmodus (de)aktivieren
Host + Entf	Strg + Alt + Entf an das Gastsystem senden
Host + ←	Strg + Alt + ← an das Gastsystem senden
Host + F#	Strg + Alt + F# an das Gastsystem senden
Host + S	Snapshot der virtuellen Maschine erstellen
Host + H	virtuelle Maschine per ACPI ausschalten
Host + R	virtuelle Maschine sofort ausschalten (Reset, Vorsicht!)

Tabelle 35.1 VirtualBox-Tastenkürzel

Gasterweiterungen installieren

Nachdem die eigentliche Installation abgeschlossen ist, sollten Sie in der virtuellen Maschine noch die sogenannten Guest Additions installieren. Sie stellen dem Gastsystem zusätzliche Treiber zur Verfügung und verbessern das Zusammenspiel mit dem Wirt: Die Maus kann nun aus der virtuellen Maschine herausbewegt werden, die virtuelle Bildschirmauflösung des Gasts passt sich automatisch an die Fenstergröße an, der Datenaustausch mit dem Wirtssystem kann über Shared Folders erfolgen, Text kann über die Zwischenablage kopiert werden etc.

Manche Distributionen liefern fertige Pakete mit den VirtualBox-Gasterweiterungen mit. Bei openSUSE werden sie sogar gleich automatisch installiert. Allerdings sind diese Pakete selten auf dem aktuellen Stand. Sie bezahlen die Bequemlichkeit der Installation also möglicherweise mit Inkompatibilitäten zu der von Ihnen eingesetzten aktuelleren VirtualBox-Version.

Debian, Ubuntu:	`virtualbox-guest-dkms`, `virtualbox-guest-utils`, `virtualbox-guest-x11`
Fedora mit RPMFusion:	`VirtualBox-guest-additions`
openSUSE:	`virtualbox-guest-kmp-default`, `virtualbox-guest-tools`, `virtualbox-guest-x11`

Bei anderen Distributionen bzw. dann, wenn Sie die neueste Version der Gasterweiterungen benötigen, müssen Sie eine manuelle Installation durchführen. Dazu werfen Sie eine eventuell eingebundene CD/DVD aus und führen dann im VirtualBox-Fenster Geräte • Gasterweiterungen einlegen aus. Im Regelfall erscheint nach einigen Sekunden in der virtuellen Maschine ein Dateimanager-Fenster, in dem Sie `autorun.sh` starten. Sollte das nicht funktionieren, helfen die folgenden Kommandos weiter:


```
root#  mkdir /media/cdrom
root#  mount /dev/sr0 /media/cdrom
root#  sh /media/cdrom/autorun.sh
```

Das Installationsprogramm richtet nun die drei neuen Kernelmodule `vboxadd`, `vboxvideo` und `vboxvfs` sowie einen neuen X-Treiber ein und fügt einige Init-Scripts hinzu, damit diese Gasterweiterungen beim nächsten Start der virtuellen Maschine auch verwendet werden.

Unter Ubuntu funktioniert die Installation der Gasterweiterungen auf Anhieb. Bei den meisten anderen Linux-Distributionen müssen Sie vor der Installation der Gasterweiterungen diverse Pakete installieren, die den C-Compiler und die Kernel-Header-Dateien enthalten. Führen Sie vorher ein Update und einen Neustart aus, um sicherzustellen, dass die installierte Kernelversion und die Version der Kernel-Header-Dateien zusammenpassen!

```
root#  yum    install gcc make kernel-headers kernel-devel \
                       elfutils-libelf-devel                  (CentOS)
root#  apt    install gcc make linux-headers-amd64            (Debian)
root#  dnf    install gcc make kernel-headers kernel-devel    (Fedora)
root#  zypper install gcc make kernel-source kernel-syms      (openSUSE)
```

Probleme mit der Maus

In seltenen Fällen kann es vorkommen, dass während der Installation (oder auch danach) der Mauszeiger in der virtuellen Maschine nicht sichtbar ist. Versuchen Sie, bei den Eigenschaften der virtuellen Maschine im Dialogblatt Anzeige den virtuellen Grafik-Adapter von VMSVGA (gilt per Default seit VirtualBox 6) auf VBoxSVGA umzustellen.

3D-Grafik

Im Idealfall stehen innerhalb der virtuellen Maschine sogar 3D-Funktionen zur Verfügung. Dazu müssen auf jeden Fall die Gasterweiterungen aktiv sein, außerdem müssen die 3D-Funktionen in den Eigenschaften der virtuellen Maschine aktiviert sein (Dialogblatt Anzeige, Option 3D-Beschleunigung). Gleichzeitig sollten Sie den Grafikspeicher auf zumindest 64 MiB stellen (siehe Abbildung 35.1).

Das allein ist aber nicht in jedem Fall ausreichend – ob 3D-Funktionen an den Gast weitergereicht werden können, hängt auch davon ab, in welchem Host-Betriebssystem VirtualBox an sich läuft und welchen Grafiktreiber Sie im Host-System verwenden. Recht gute Erfahrungen habe ich mit Linux-Hosts in Kombination mit dem Intel-Grafiktreiber gemacht. In vielen anderen Fällen, insbesondere auch, wenn VirtualBox unter macOS läuft, funktionierte die 3D-Unterstützung gar nicht. Und selbst wenn die 3D-Funktionen prinzipiell durchgereicht werden, können fallweise Fehldarstellungen auftreten, z. B. nach der Veränderung der Fenstergröße.

Wenn Sie sich vergewissern möchten, ob alles funktioniert, installieren Sie in der virtuellen Maschine je nach Distribution das Paket `mesa-utils`, `glx-utils` oder `Mesa-`

`demo-x` und führen dann `glxinfo` aus. Das Ergebnis sollte so wie im folgenden Listing aussehen:

```
user$  glxinfo | grep render
...
OpenGL renderer string: Chromium
```

Wenn der `OpenGL renderer string` hingegen `llvmpipe` enthält, dann werden die 3D-Funktionen durch die CPU emuliert, was spürbar langsamer ist.

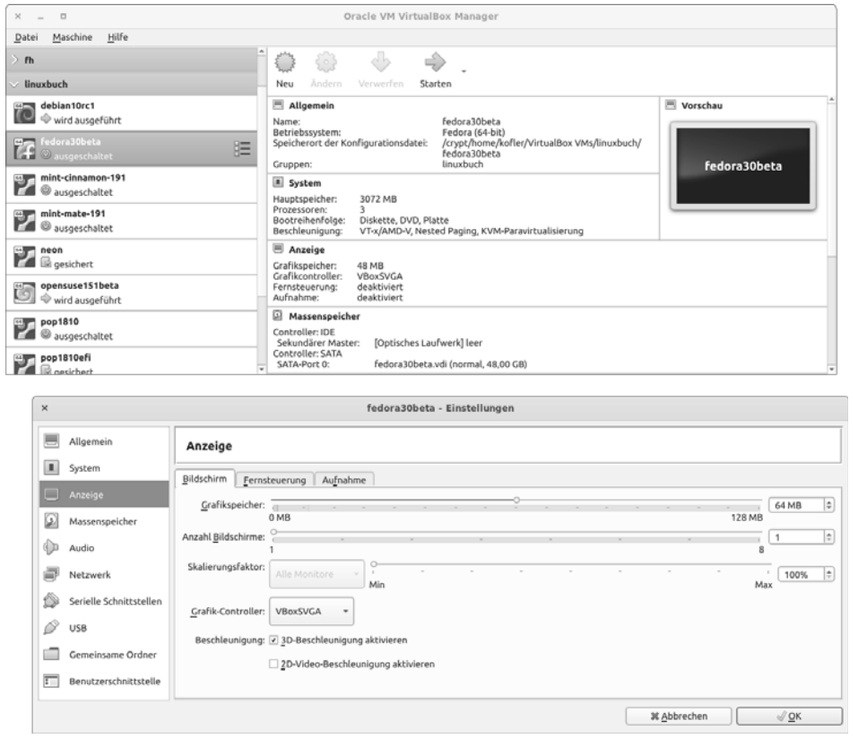

Abbildung 35.1 Überblick über alle virtuellen Maschinen (oben) und deren Einstellungen

Eine virtuelle Maschine mit Windows einrichten

Sofern Sie über eine Installations-CD/DVD bzw. die entsprechende ISO-Datei sowie eine gültige Lizenz und den dazugehörenden Schlüssel verfügen, können Sie in VirtualBox auch Windows installieren (siehe Abbildung 35.2). Die Installation von Windows und der VirtualBox-Gasterweiterungen verlief bei meinen Tests stets problemlos.

Warten Sie mit der Online-Registrierung so lange ab, bis Sie mit der Leistung zufrieden sind. Wenn Sie später in den Einstellungen der virtuellen Maschine das RAM vergrößern oder andere virtuelle Hardware-Parameter ändern, müssen Sie unter Umständen die Registrierung wiederholen!

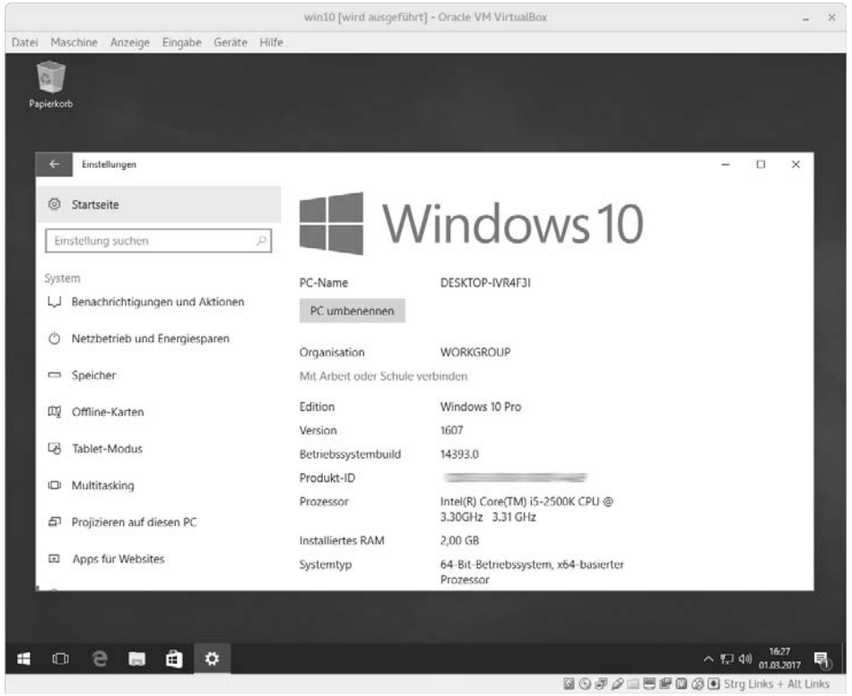

Abbildung 35.2 Windows 10 in einer virtuellen Maschine unter Linux ausführen

35.3 Arbeitstechniken und Konfigurationstipps

Dieser Abschnitt gibt Tipps zur Optimierung virtueller Maschinen sowie zum Datenaustausch zwischen dem Host-System bzw. dem »realen« lokalen Netzwerk und den virtuellen Maschinen.

Netzwerkkonfiguration

VirtualBox stellt seinen Gästen die Netzwerkinfrastruktur des Wirts in Form einer virtuellen Netzwerkkarte zur Verfügung. Dabei existieren unterschiedliche Verfahren, wie der Netzwerkverkehr von der virtuellen Netzwerkkarte in das reale Netzwerk geleitet wird. Die entsprechenden Parameter finden Sie im Einstellungsdialog im

Dialogblatt Netzwerk (siehe Abbildung 35.3). Entscheidend ist die Einstellung des Listenfelds Angeschlossen an, wobei der Vorgabewert NAT lautet.

- **NAT:** Bei der NAT-Variante stellt VirtualBox seinen Gästen einen eigenen DHCP-Server zur Verfügung und realisiert Masquerading (NAT). Auf diese Weise können die Gäste den Internetzugang des Wirtssystems nutzen. Ein Zugang zum lokalen Netzwerk ist wegen der unterschiedlichen Adressbereiche für das lokale Netz und das virtuelle NAT-Netz des Virtualisierungssystems unmöglich. Ebenso wenig können Sie vom Host eine SSH-Verbindung zum Gast herstellen. Die virtuellen Maschinen sind vom Host wie durch eine einfache Firewall getrennt.

  Bei der NAT-Variante verwendet VirtualBox auf dem Host die IP-Adresse 10.0.2.2. Die virtuellen Maschinen erhalten andere 10.0.2.*-Adressen.

- **Netzwerkbrücke:** Bei dieser Variante erscheint der Gast als zusätzlicher Client im lokalen Netz. Diese Variante ist optimal, wenn es im lokalen Netzwerk einen DHCP-Server gibt bzw. wenn der Host-Rechner mit einem ADSL- oder WLAN-Router verbunden ist. Die virtuellen Gäste beziehen ihre Netzwerkkonfiguration dann über diesen Server/Router und können sowohl auf das lokale Netzwerk als auch auf das Internet zugreifen. Wenn Ihr Host-Rechner mehrere Netzwerkschnittstellen besitzt, müssen Sie angeben, welche Schnittstelle die Verbindung zum lokalen Netzwerk herstellt.

  Im Büro ist diese Variante meine bevorzugte Konfiguration: Reale und virtuelle Maschinen sind damit im lokalen Netzwerk gleichwertige Partner, und der Datenaustausch via SSH, Samba etc. funktioniert unkompliziert. Beachten Sie aber, dass die Netzwerkbrücke in manchen (Unternehmens-)WLANs nicht funktioniert. Die besten Erfahrungen habe ich mit dieser Konfigurationsvariante gemacht, wenn der Host-Rechner über ein Ethernet-Kabel (also nicht über WLAN) mit dem lokalen Netzwerk verbunden ist.

- **Host-only Adapter:** Bei dieser Variante kann der Gast über die Netzwerkfunktionen nur mit dem Wirt kommunizieren, nicht aber mit anderen Rechnern im lokalen Netzwerk oder mit dem Internet. Diese Variante ist dann zweckmäßig, wenn Sie ein von außen nicht zugängliches Testsystem aus mehreren virtuellen Maschinen aufbauen möchten.

- **Internes Netzwerk:** Hier bildet VirtualBox ein virtuelles Netzwerk, in dem ausschließlich virtuelle Maschinen kommunizieren können. Sie haben bei dieser Variante weder Zugriff auf das lokale Netzwerk noch auf das Internet.

Sie können virtuelle Maschinen mit bis zu vier Netzwerkadaptern ausstatten. Das gibt Ihnen die Möglichkeit, mehrere Konfigurationsvarianten parallel zu verwenden – z. B. einen NAT-Adapter, damit die virtuellen Maschinen Internetzugang erhalten,

und einen Host-only-Adapter, damit Sie eine SSH-Verbindung zwischen den virtuellen Maschinen und dem Host-Rechner herstellen können.

Die Netzwerkkonfiguration kann im laufenden Betrieb geändert werden! Es ist also nicht erforderlich, die virtuelle Maschine bei jeder Änderung neu zu starten. Der schnellste Weg in den Konfigurationsdialog führt über das Icon Aktivität der Netzwerkadapter in der Statusleiste des VirtualBox-Fensters.

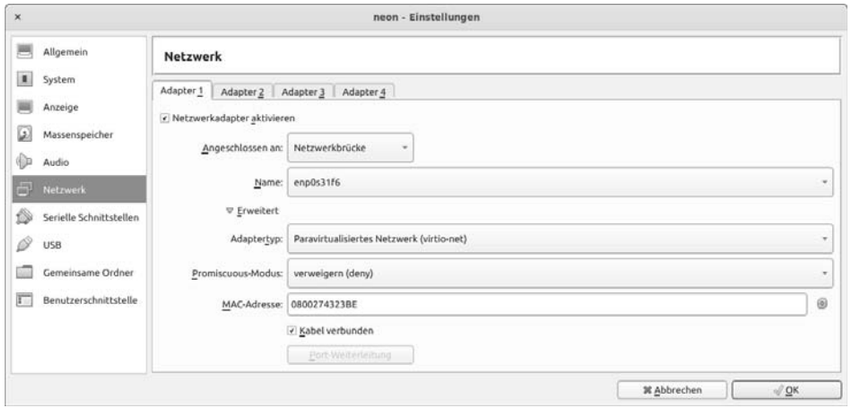

Abbildung 35.3 Netzwerkeinstellungen für die virtuelle Maschine

Datenaustausch über die Zwischenablage

In den Einstellungen der virtuellen Maschine können Sie im Dialogblatt Allgemein • Erweitert für die gemeinsame Zwischenablage und für die Funktion Drag & Drop den Modus Bidirektional aktivieren. Beide Optionen setzen auf jeden Fall voraus, dass in der virtuellen Maschine die Gasterweiterungen installiert sind.

Diese Konfiguration gibt Ihnen die Möglichkeit, über die Zwischenablage Text zwischen dem Host und dem Gast zu kopieren. Außerdem können Sie nun per Drag & Drop Dateien zwischen einem Dateimanager im Host und einem Dateimanager im Gast hin- und herkopieren. Fallweise hat dies bei meinen Tests gut funktioniert, aber wirklich ausgereift wirkt diese Funktion nicht.

Datenaustausch mit einem Shared Folder

Host-Konfiguration

Ein zuverlässigerer Weg zum Datenaustausch zwischen Wirt und Gast sind sogenannte Shared Folder. Zur Konfiguration öffnen Sie mit Ändern den Einstellungsdialog, wechseln in das Dialogblatt Gemeinsame Ordner, wählen dann ein lokales Verzeichnis auf dem Wirtssystem aus und geben dem Ordner einen Namen (z. B. `myshare`). Das

Verzeichnis gilt spezifisch für eine bestimmte virtuelle Maschine. Für Windows-Gäste aktivieren Sie auch gleich die Option AUTOMATISCH EINBINDEN.

Linux-Gäste

Nach einem Neustart eines Linux-Gastsystems ist nun ein manuelles `mount`-Kommando erforderlich, um auf das gemeinsame Verzeichnis zugreifen zu können. Dabei müssen Sie `myshare` durch den Namen ersetzen, den Sie bei der Konfiguration verwendet haben.

```
root@gast#  mkdir /media/vbox-share
root@gast#  mount -t vboxsf myshare /media/vbox-share
```

Wenn Linux die Fehlermeldung *unknown filesystem vboxsf* liefert, sind die VirtualBox-Gasterweiterungen nicht richtig installiert. Abhilfe schafft bei den meisten Distributionen die Installation des Pakets `virtualbox-guest-utils`.

Windows-Gäste

In Windows-Gästen finden Sie das gemeinsame Verzeichnis im Explorer als Netzwerkverzeichnis des virtuellen Rechners `vboxsrv`. Wenn Sie bei der Konfiguration die Option AUTOMATISCH EINBINDEN verwendet haben, dann wird dem Verzeichnis unter Windows auch gleich ein eigener Laufwerksbuchstabe zugeordnet.

USB-Geräte in virtuellen Maschinen

Sofern Sie auf dem Host das VirtualBox Extension Pack installiert haben, können Sie USB-Geräte auch in virtuellen Maschinen nutzen. Das funktioniert nur, wenn das USB-Gerät im Wirtssystem *nicht* verwendet wird. USB-Datenträger werden im Wirtssystem normalerweise automatisch in das Dateisystem eingebunden; Sie müssen sie wieder aus ihm lösen, um sie im Gast verwenden zu können.

Eine weitere Voraussetzung besteht darin, dass der Benutzer, der VirtualBox ausführt, Mitglied der Gruppe `vboxusers` ist. Schließlich müssen Sie darauf achten, dass der USB-CONTROLLER bei den Einstellungen der virtuellen Maschine im Dialogblatt USB aktiviert ist. In diesem Dialogblatt können Sie auch einen Filter definieren, um ein USB-Gerät direkt einer virtuellen Maschine zuzuordnen. Das ist aber keine zwingende Voraussetzung. Sie können das USB-Gerät nach dem Einschalten auch dynamisch in der VirtualBox-Statusleiste beim USB-Icon der virtuellen Maschine zuordnen.

Export/Import virtueller Maschinen

Um eine virtuelle Maschine weiterzugeben, erzeugen Sie mit DATEI • APPLIANCE EXPORTIEREN eine sogenannte Virtual Appliance, also eine zur Weitergabe bestimmte virtuelle Maschine, die üblicherweise aus zwei Dateien besteht: `*.ovf` enthält eine Beschreibung der virtuellen Maschine, `*.vmdk` das Festplatten-Image in komprimierter Form. Diese virtuelle Maschine können Sie nun bei einer anderen VirtualBox-Installation mit DATEI • APPLIANCE IMPORTIEREN wieder einrichten.

Eine virtuelle Maschine auf einen anderen Host übertragen

Wenn es Ihnen nur darum geht, eine oder mehrere virtuelle Maschinen von einem Rechner auf einen anderen zu übertragen, können Sie sich die Umwandlung in eine Virtual Appliance sparen. In diesem Fall reicht es aus, das betreffende Verzeichnis `VirtualBox VMs/vm-name` zu kopieren. Anschließend führen Sie in VirtualBox das Kommando MASCHINE • HINZUFÜGEN aus und wählen die `*.vbox`-Datei aus.

Geschwindigkeitsoptimierung

Mit zwei Optionen bei der Einstellung der virtuellen Hardware können Sie ein klein wenig mehr Geschwindigkeit aus Ihren virtuellen Maschinen herauskitzeln:

- **Host-Caching für die virtuelle Festplatte:** Im Dialogblatt MASSENSPEICHER der virtuellen Maschine können Sie für den SATA-Controller die Option HOST-I/O-CACHE VERWENDEN aktivieren. Sie erreichen damit, dass Schreibzugriffe zwischengespeichert werden, was die Geschwindigkeit I/O-lastiger Vorgänge stark vergrößern kann. Der Nachteil: Sollte der Host-Rechner abstürzen, riskieren Sie ein beschädigtes Dateisystem in der virtuellen Maschine.
- **Paravirtualisierte Netzwerktreiber:** Sofern es sich bei der virtuellen Maschine um eine Linux-Distribution handelt, können Sie im Dialogblatt NETZWERK bei den erweiterten Einstellungen die Option PARAVIRTUALISIERTES NETZWERK (VIRTIO-NET) aktivieren (siehe Abbildung 35.3). VirtualBox spielt der virtuellen Maschine nun nicht mehr die Logik eines Netzwerkadapters vor, sondern spricht direkt mit dem `virtio-net`-Treiber des Linux-Kernels. Das ist deutlich effizienter.

Virtuelle Maschinen gruppieren und unsichtbar ausführen

Gruppen

Wenn Sie in VirtualBox viele virtuelle Maschinen einrichten, können Sie diese per Kontextmenü gruppieren. Abbildung 35.1 zeigt eine ausgeklappte Gruppe mit virtuellen Maschinen, die ich während der Arbeit an diesem Buch eingerichtet habe. Eine weitere Gruppe (nicht ausgeklappt) enthält virtuelle Maschinen für meinen Unterricht an einer Fachhochschule.

Gruppen schaffen nicht nur mehr Übersicht in der Liste der virtuellen Maschinen, sie geben Ihnen auch die Möglichkeit, alle virtuellen Maschinen einer Gruppe gemeinsam zu starten bzw. herunterzufahren. Besonders praktisch ist das, wenn die Gruppe inhaltlich zusammengehört (z. B. zum Test eines Servers mit zwei Clients).

> **Tipp**
>
> Sie können Maschinen per Drag&Drop verschieben, natürlich auch zwischen verschiedenen Gruppen.

Virtuelle Maschinen ohne Fenster

Normalerweise wird jede laufende virtuelle Maschine in einem eigenen Fenster angezeigt. Beim Schließen des Fensters haben Sie die Wahl, den Status der virtuellen Maschine zu speichern (die virtuelle Maschine also gewissermaßen zu pausieren), sie per ACPI herunterzufahren oder sie gewaltsam zu stoppen (wie durch das Lösen eines Netzkabels).

Mitunter wäre es aber praktisch, virtuelle Maschinen unsichtbar, also *ohne* eigenes Fenster auszuführen. Das gilt besonders für Server-Installationen, die ohnedies im Textmodus laufen. Für derartige virtuelle Maschinen können Sie beim Start-Button den Menüeintrag OHNE GUI STARTEN wählen.

Noch mehr Flexibilität gibt der Eintrag ABKOPPELBARER START (siehe Abbildung 35.4). Damit wird die virtuelle Maschine beim Start wie üblich in einem Fenster angezeigt. Mit MASCHINE • GUI ABKOPPELN können Sie das Fenster dann aber bei Bedarf schließen, ohne die virtuelle Maschine zu stoppen. Mit einem Doppelklick auf das Symbol der virtuellen Maschine im VirtualBox-Hauptfenster können Sie die Benutzeroberfläche der virtuellen Maschine bei Bedarf wiederbeleben.

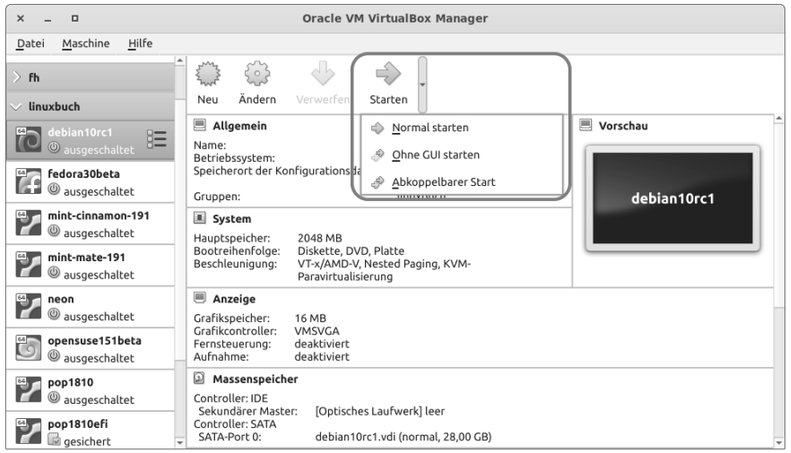

Abbildung 35.4 Das Menü des Start-Buttons enthält zwei versteckte Einträge zum Start der virtuellen Maschine ohne bzw. mit abkoppelbarer Oberfläche.

Leider gibt es keine Möglichkeit, in den Einstellungen der virtuellen Maschine voreinzustellen, dass diese virtuelle Maschine immer ohne Fenster gestartet werden soll.

VirtualBox mit einem hochauflösenden Monitor verwenden

Wenn Sie VirtualBox auf einem Host mit einem 4k-Monitor verwenden, kann es sein, dass die Darstellung der virtuellen Maschine viel zu klein ist. Sie haben zwei Möglichkeiten, das zu ändern:

- In den Eigenschaften der virtuellen Maschine können Sie im Dialogblatt ANZEIGE einen festen Skalierungsfaktor einstellen.
- Oder Sie aktivieren im laufenden Betrieb mit der Host-Taste und C den skalierten Modus. In diesem Modus wird der virtuelle Bildschirm an die Fenstergröße angepasst und entsprechend skaliert. Aus nicht ganz nachvollziehbaren Gründen verliert das VirtualBox-Fenster in diesem Modus seine Menüs und die Statusleiste. Nochmals Host+C schaltet den skalierten Modus wieder aus, Menüs und Statusleiste tauchen wieder auf.

Bei beiden Varianten leiden allerdings die Geschwindigkeit des Bildaufbaus und die Darstellungsqualität.

VirtualBox per Kommando steuern (vboxmanage)

Anstatt VirtualBox über seine Benutzeroberfläche zu bedienen, können Sie sehr viele Operationen auch über das Kommando `vboxmanage` ausführen. Es fehlt hier der Platz, auf die unzähligen Subkommandos und Optionen einzugehen. Stattdessen beschränke ich mich in diesem und dem folgenden Beispiel auf einige wenige Anwendungsmöglichkeiten. Eine umfassende Referenz finden Sie im VirtualBox-Handbuch:

https://www.virtualbox.org/manual/ch08.html

Alle virtuellen Maschinen pausieren, fortsetzen oder herunterfahren

Virtuelle Maschinen, selbst wenn sie sich scheinbar im Leerlauf befinden, verursachen mitunter eine beträchtliche CPU-Last. Das hat unter anderem mit den vielen Hintergrunddiensten in den virtuellen Maschinen zu tun, die nach Updates suchen. Wenn ich die CPU für andere Aufgaben dringender benötige oder wenn mich ganz einfach der Lüfter meines Notebooks irritiert, pausiere ich einfach mit dem folgenden Script sämtliche virtuellen Maschinen:

```
#!/bin/bash
vboxmanage list runningvms | \
  sed -r 's/.*\{(.*)\}/\1/' |
  xargs -L1 -I {} vboxmanage controlvm {} pause
```

`vboxmanage list runningvms` liefert eine Liste aller laufenden virtuellen Maschinen. `sed` extrahiert daraus die ID-Nummern. `xargs` gibt diese an ein zweites `vboxmanage`-Kommando weiter und führt `pause` aus. Um die Ausführung der virtuellen Maschinen fortzusetzen, gibt es ein zweites, gleichartiges Script mit `resume` anstelle von `pause`. Ein

drittes Script rufe ich auf, bevor ich den Rechner herunterfahre. Es sendet mit `vboxmanage controlvm acpipowerbutton` an alle virtuellen Maschinen ein ACPI-Shutdown-Signal.

Virtuelle Festplatten vergrößern

Linux-Gast

Die Benutzeroberfläche von VirtualBox bietet keine Möglichkeit, eine virtuelle Festplatte nachträglich zu vergrößern. Abhilfe schafft wiederum das Kommando `vboxmanage`. Bevor Sie loslegen, müssen Sie Ihre virtuelle Maschine herunterfahren. Außerdem ist ein vollständiges Backup sehr zu empfehlen!

Anschließend suchen Sie die `*.vdi`-Datei der virtuellen Festplatte und wenden darauf das Kommando `vboxmanage` an. Mit der Option `--resize` geben Sie die gewünschte neue Größe in MiB an. Im Regelfall wird das Kommando blitzschnell ausgeführt.

```
root# vboxmanage modifyhd debian.vdi --resize 60000
```

Das ist aber erst die halbe Miete. Die virtuelle Maschine weiß nämlich noch nichts davon, dass ihre Festplatte größer geworden ist. Bei einer Gast-Installation ohne LVM und mit `ext4`- oder `xfs`-Dateisystemen binden Sie nun ein ISO-Image einer Linux-Live-CD in das virtuelle CD/DVD-Laufwerk ein und starten innerhalb der virtuellen Maschine ein Live-System. Dort führen Sie `parted /dev/sda` aus und können nun die Größe der letzten Partition erhöhen. Anschließend müssen Sie auch das darin enthaltene Dateisystem mit `resize2fs` oder `xfs_growfs` vergrößern.

Wenn Sie im Linux-Gast hingegen LVM oder `btrfs`-Dateisysteme verwenden, können Sie das Dateisystem im laufenden Betrieb vergrößern. Diese Eingriffe sind natürlich nicht ganz ungefährlich. Lesen Sie vorher die relevanten Abschnitte aus Kapitel 21, »Administration des Dateisystems«!

Windows-Gast

Analog kann auch ein Windows-Dateisystem vergrößert werden. Auch in diesem Fall fahren Sie die virtuelle Maschine zuerst herunter und vergrößern die `*.vdi`-Datei mit dem Kommando `vboxmanage`. Dann starten Sie Windows, öffen darin ein Eingabeaufforderungsfenster mit Administratorrechten und führen die folgenden Kommandos aus:

```
> Diskpart
list disk
select disk 0
list partition
select partition 2
extend
```

`list disk` liefert eine Liste aller virtuellen Festplatten. Normalerweise muss die erste Platte mit dem Index 0 ausgewählt werden. Nun ermittelt `list partition` die Partitionen. Abermals muss mit `select` eine Partition zur weiteren Bearbeitung ausgewählt

werden – im Regelfall die letzte. Mit `extend` wird diese nun auf die maximale Größe erweitert.

35.4 Vagrant

Das Einrichten einer neuen virtuellen Maschine ist mit relativ viel Handarbeit verbunden. Solange es nur um eine Installation geht, ist das kein großes Problem. Wenn Sie aber regelmäßig virtuelle Maschinen einrichten müssen und dabei womöglich Wert darauf legen, dass die virtuellen Maschinen reproduzierbar exakt gleich konfiguriert sind, sollten Sie sich mit dem Programm *Vagrant* anfreunden. Vagrant ist ein Werkzeug, das beim Einrichten, Ausführen, Steuern und Stoppen von virtuellen Umgebungen hilft.

Vagrant wird zusammen mit einigen weiteren Programmen (Atlas, Packer, Vault, Nomad, Consul) von der Firma Hashicorp entwickelt. Alle Produkte verwenden Open-Source-Lizenzen und stehen kostenlos zur Verfügung. Zum Teil gibt es darüber hinaus Enterprise-Varianten mit Zusatzfunktionen für zahlende Kunden.

https://www.hashicorp.com/#open-source-tools

Vagrant ist unabhängig von der Virtualisierungsplattform!

Auch wenn ich Ihnen Vagrant hier im VirtualBox-Kapitel vorstelle, kommt das Programm auch mit anderen Betriebssystemen (macOS, Windows) und mit diversen Virtualisierungssystemen zurecht, z. B. mit VMware, Hyper-V sowie KVM/libvirt (siehe Kapitel 36, »KVM«).

Nomenklatur

Die Dokumentation zu Vagrant ist leichter zu verstehen, wenn Sie sich zuerst mit einigen Begriffen vertraut machen:

- **Vagrant-Datei:** Vagrant richtet virtuelle Maschinen auf der Basis einer Vagrant-Datei und einer Box ein. Die Textdatei `Vagrantfile` gibt die Quelle der Box-Datei an und beschreibt, welche Operationen auf die Box angewendet werden müssen, um die virtuelle Maschine fertigzustellen. Dieser einmalig durchzuführende Vorgang wird »Provisioning« genannt. Die Anweisungen in der Vagrant-Datei werden in der Syntax der Programmiersprache Ruby angegeben. Die Vagrant-Datei kann beispielsweise Kommandos zum Einrichten der Netzwerkverbindung und des SSH-Servers enthalten. Sie können aber auch externe Scripts aufrufen, die zur Installation von Zusatz-Software oder für Konfigurationsarbeiten in der virtuellen Maschine auf Werkzeuge wie *Puppet* oder *Chef* zurückgreifen.
- **Boxes:** Eine Box ist eine komprimierte Datei, die eine virtuelle Maschine enthält. Box-Dateien sind wegen des inkludierten Festplatten-Images zumeist recht groß

(mehrere Hundert MiB). Auf *https://atlas.hashicorp.com/boxes/search* finden Sie einen Katalog kostenlos verfügbarer Vagrant-Boxes. Vagrant kommt aber auch mit Boxes zurecht, die lokal gespeichert sind oder auf anderen Webservern zugänglich sind.

Beim ersten Start wird die virtuelle Maschine zuerst aus der Box geklont; anschließend führt Vagrant die in `Vagrantfile` aufgezählten Konfigurationsarbeiten durch, führt am Klon also noch Änderungen durch. Die Box selbst bleibt dabei unverändert und kann später neuerlich als Basis verwendet werden, wenn weitere Instanzen erzeugt werden sollen oder die virtuelle Maschine neu eingerichtet werden soll.

- **Vagrant-Kommando:** Die gesamte Administration von Vagrant erfolgt durch das Kommando `vagrant`. Damit starten und stoppen Sie virtuelle Maschinen, stellen SSH-Verbindungen zu ihnen her etc.
- **Provider:** Vagrant verwendet standardmäßig VirtualBox als Virtualisierungssystem. Sogenannte Provider stellen optionale Schnittstellen zu anderen Virtualisierungssystemen her. Einige Provider sind standardmäßig in Vagrant enthalten, andere können extra installiert werden.
- **Plugins:** Vagrant hat einen modularen Aufbau. Selbst etliche Grundfunktionen sind als Plugins realisiert. Zur Realisierung von Zusatzfunktionen können Sie Vagrant durch externe Plugins erweitern (`vagrant plugin install name`).

Installation

Bei vielen Distributionen installieren Sie Vagrant am einfachsten mit den Paketverwaltungskommandos. Allerdings erhalten Sie damit nicht immer die aktuellste Version. Auf der Vagrant-Website *https://www.vagrantup.com* finden Sie aktuelle Pakete im Debian- und RPM-Format, deren Installation in der Regel auf Anhieb aus dem Webbrowser heraus gelingt:

```
user$  vagrant --version
Vagrant 2.2.3
```

Base Boxes

Ich gehe in diesem Buch nur auf die Nutzung und Modifizierung vorgefertiger Boxes ein. Fortgeschrittene Vagrant-Anwender können aber auch vollkommen neue Boxes erzeugen. In der Regel ist es zweckmäßig, dabei eine sogenannte *Base Box* einzurichten, also eine virtuelle Maschine, die auf einer minimalen Installation der jeweiligen Distribution basiert und die speziell für Vagrant vorkonfiguriert ist. Eine typische Vagrant-Konfiguration besteht aus einem SSH-Server, einem `vagrant`-Benutzer mit `sudo`-Rechten ohne Passwort und eventuell der Installation von Gasterweiterungen für das gewünschte Virtualisierungssystem. Eine ausführliche Anleitung, wie Sie Vagrant-kompatible Base Boxes einrichten, finden Sie hier:

https://www.vagrantup.com/docs/boxes/base.html

Hello World!

Um Vagrant auszuprobieren, greifen Sie am besten auf eine der vielen vorgefertigten Vagrant-Boxes zurück. Der Katalog auf *https://app.vagrantup.com/boxes/search* enthält leider nicht viel mehr als den Namen der jeweiligen Box und eine Liste der unterstützten Provider. Unbegreiflicherweise fehlt eine Beschreibung, welche Zielsetzung die jeweilige Box hat. Nicht einmal die Größe der Box ist dokumentiert.

Für erste Experimente können Sie z. B. die Box `ubuntu/bionic64` verwenden. Sie enthält einen tagesaktuellen Build einer Minimalinstallation von Ubuntu 18.04 für den Server-Einsatz (also ohne grafische Benutzeroberfläche):

```
user$  mkdir u1804
user$  cd u1804
user$  vagrant init ubuntu/bionic64
A `Vagrantfile` has been placed in this directory. You are now
ready to `vagrant up` your first virtual environment!
user$  vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Adding box 'ubuntu/bionic64' (v20190508.0.0) for virtualbox
    default: Downloading: https://vagrantcloud.com/ubuntu/boxes/bionic64/
                          versions/20190508.0.0/providers/virtualbox.box
==> default: Preparing network interfaces based on configuration...
    default: Adapter 1: nat
==> default: Forwarding ports...
    default: 22 (guest) => 2222 (host) (adapter 1)
==> default: Mounting shared folders...
    default: /vagrant => /home/kofler/u1804
...
user$  vagrant ssh
Welcome to Ubuntu 18.04.2 LTS (GNU/Linux 4.15.0-48-generic x86_64)
```

Kurz einige Erklärungen zu den obigen Kommandos, deren Ausgaben aus Platzgründen stark gekürzt abgedruckt sind. `vagrant init` lädt vom Hashicorp-Server die Datei `Vagrantfile` für die gewünschte virtuelle Maschine. Das geht schnell, da die Datei nur wenige Kilobyte groß ist. Sie wird im gerade aktuellen Verzeichnis gespeichert. `vagrant init` verwendet standardmäßig VirtualBox als Virtualisierungsplattform. Wenn Sie ein anderes System verwenden möchten, wählen Sie dieses mit `--provider name` aus.

`vagrant up` startet die virtuelle Maschine. Ab dem zweiten Mal wird auch dieses Kommando recht schnell ausgeführt, beim ersten Mal dauert es aber geraume Zeit: Zuerst muss nämlich die Box für die virtuelle Maschine heruntergeladen werden. Diese Box sowie diverse Zusatzdateien werden im Verzeichnis `.vagrant.d/boxes` gespeichert, also getrennt von dem Verzeichnis, in dem sich `Vagrantfile` befindet. Das hat den Vorteil, dass später bei Bedarf weitere virtuelle Maschinen auf Basis der bereits

vorhandenen Box eingerichtet werden können. Für `ubuntu/bionic64` beträgt der Platzbedarf der Box ca. 300 MiB.

Sobald die Box heruntergeladen ist, wird die entsprechende virtuelle Maschine eingerichtet. Im obigen Beispiel verwendet Vagrant den Default-Provider für VirtualBox. Die Dateien der virtuellen Maschine landen daher in dem von VirtualBox vorgesehenen Verzeichnis. Wenn Sie die VirtualBox-Defaulteinstellungen nicht verändert haben, ist das `VirtualBox VMs` in Ihrem Heimatverzeichnis. Damit gibt es nun Dateien an drei verschiedenen Orten:

- in Ihrem eigenen Vagrant-Verzeichnis: Es enthält neben `Vagrantfile` einige weitere Konfigurationsdateien und beansprucht nur wenig Speicherplatz. Alle `vagrant`-Kommandos müssen in diesem Verzeichnis oder in einem seiner Unterverzeichnisse ausgeführt werden.
- in `.vagrant.d/boxes`: Das Verzeichnis enthält je eine Box für alle irgendwann mit Vagrant eingerichteten Maschinen. Der Platzbedarf beträgt typischerweise einige Hundert MiB pro Box.
- in `VirtualBox VMs`: Dieses Verzeichnis enthält die virtuellen Maschinen inklusive der Disk-Images für jede mit Vagrant eingerichtete Maschine. Der Platzbedarf ist hoch und beträgt oft mehrere GiB pro virtueller Maschine.

Die von Vagrant eingerichtete VirtualBox-Maschine wird im VirtualBox-Hauptfenster zwischen selbst erzeugten virtuellen Maschien aufgelistet. Ihr Name endet immer mit einer zufällig generierten Zahl (siehe Abbildung 35.5). `vagrant up` startet die virtuelle Maschine unsichtbar, also ohne ein VirtualBox-Fenster zu öffnen. Zwar ist es möglich, per Doppelklick auf die Liste der virtuellen Maschinen ein entsprechendes Fenster zu öffnen; im Normalfall ist es aber üblich, Vagrant-Maschinen per SSH zu administrieren.

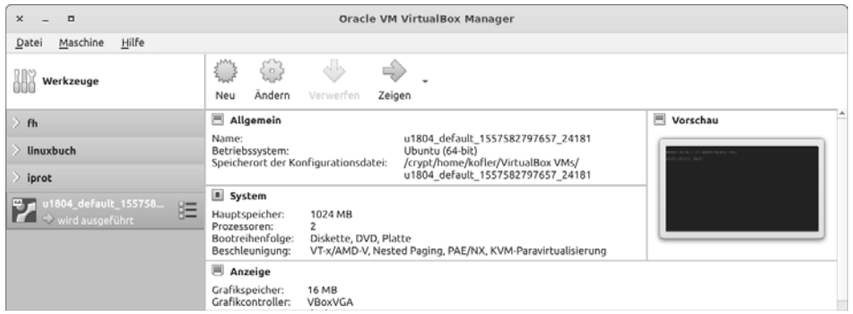

Abbildung 35.5 Die von Vagrant eingerichteten virtuellen Maschinen sind in der Liste der VirtualBox-Maschinen an der Namenserweiterung »default_nnn« zu erkennen.

Bento-Boxes

Das Boxes-Angebot auf *https://app.vagrantup.com/boxes/search* ist leider ziemlich unübersichtlich. Wenn Sie auf der Suche nach kleinen, vernünftig vorkonfigurierten Boxes für die wichtigsten Linux-Distributionen sind, lohnt sich ein Blick auf die Webseite *https://app.vagrantup.com/bento*! Zur Verwendung einer derartigen Box führen Sie einfach `vagrant init bento/<name>` aus.

Das Bento-Projekt (*https://chef.github.io/bento*) hat es sich zur Aufgabe gemacht, minimale Vagrant-Boxes zu erzeugen und zu verwalten.

Netzwerkkonfiguration

Vagrant-Maschinen verwenden in VirtualBox einen NAT-Netzwerkadapter. Das ist aus Sicherheitsgründen zweckmäßig, weil in Vagrant-Maschinen üblicherweise der Account `vagrant` mit einem gleichnamigen Passwort eingerichtet ist. Wäre die virtuelle Maschine im lokalen Netz oder gar im Internet öffentlich erreichbar, würde sie unweigerlich das Ziel von Hacker-Angriffen.

Damit zwischen dem Host-Rechner und der virtuellen Maschine eine SSH-Verbindung möglich ist, richtet Vagrant standardmäßig eine Port-Umleitung zwischen dem Port 22 der virtuellen Maschine und dem Port 2222 des Hosts ein. Ist dieser Port schon von einer anderen Box belegt, sucht `vagrant up` selbstständig einen anderen freien Port mit der Nummer 22*nn*.

Um eine SSH-Verbindung zur virtuellen Maschine herzustellen, führen Sie einfach das Kommando `vagrant ssh` aus. Vagrant startet den SSH-Client dann mit den richtigen Optionen. Sie brauchen kein Passwort anzugeben. In der virtuellen Maschine werden Sie in der Regel als Benutzer `vagrant` bzw. beim hier vorgestellten Beispiel als Benutzer `ubuntu` angemeldet. Dank einer in `/etc/sudoers.d` vorgesehenen Konfigurationsdatei erlangen Sie mit `sudo -s` ohne Passwort `root`-Rechte:

```
user@hostsystem$  vagrant ssh
Welcome to Ubuntu 18.04.2 LTS (GNU/Linux 4.15.0-48-generic x86_64)
vagrant@ubuntu-bionic:~$  sudo -s
root@ubuntu-bionic:~#  cat /etc/sudoers.d/90-cloud-init-users
# User rules for ubuntu
ubuntu ALL=(ALL) NOPASSWD:ALL
```

Bei vielen Boxes richtet Vagrant darüber hinaus ein gemeinsames Verzeichnis zwischen Host und virtueller Maschine ein. Auf dem Host wird dazu das Verzeichnis verwendet, in dem sich `Vagrantfile` befindet. Auf dem Client ist der Shared Folder in Linux-Gästen üblicherweise unter `/vagrant` zugänglich.

Beim vorgestellten Ubuntu-Beispiel ist das zum Datenaustausch vorgesehene Verzeichnis als VirtualBox Shared Folder realisiert. In der Box sind dazu standardmäßig die VirtualBox-Gasterweiterungen installiert. Andere von Vagrant unterstützte Verfahren zur Realisierung des gemeinsamen Verzeichnisses basieren auf Rsync (Synchronisierung nur beim Start), NFS oder SSHFS (`vagrant-sshfs`-Plugin).

Default-Login

Es ist üblich, dass Vagrant-Maschinen einen Default-Account mit dem Login-Namen `vagrant` und einem gleichnamigen Passwort haben. Dieser Benutzer wird auch für SSH-Verbindungen verwendet, wobei die Authentifizierung über eine Schlüsseldatei erfolgt.

Die in diesem Beispiel vorgestellte Ubuntu-Box widerspricht leider den Vagrant-Empfehlungen: In diesem Fall lautet der Default-Login `ubuntu`. Als Passwort wird ein zufälliger hexadezimaler Code verwendet. Um ein eigenes Passwort einzustellen, stellen Sie mit `vagrant ssh` eine Verbindung zur virtuellen Maschine her und führen dann `passwd` aus.

Kommando	Bedeutung
`vagrant box list`	heruntergeladene Vagrant-Boxes auflisten
`vagrant box update`	Vagrant-Box aktualisieren
`vagrant destroy`	Vagrant-Maschine löschen
`vagrant halt`	Vagrant-Maschine herunterfahren
`vagrant init name`	vorgefertige Vagrant-Datei herunterladen
`vagrant login`	Login zu eigenem Atlas-Account durchführen
`vagrant plugin install name`	Plugin installieren
`vagrant provision`	Provisioning wiederholen
`vagrant resume`	pausierte Vagrant-Maschine wieder aktivieren
`vagrant share`	Vagrant-Maschine öffentlich zugänglich machen
`vagrant ssh`	SSH-Verbindung zur Vagrant-Maschine herstellen
`vagrant status`	Status der Vagrant-Maschine anzeigen
`vagrant suspend`	Vagrant-Maschine pausieren
`vagrant up`	Vagrant-Maschine starten

Tabelle 35.2 Wichtige vagrant-Kommandos

Administration

Die gesamte Administration von Vagrant erfolgt mit dem gleichnamigen Kommando (siehe Tabelle 35.2). Soweit sich die gewünschte Operation auf eine Box bezieht, sucht `vagrant` zuerst im aktuellen Verzeichnis nach `Vagrantfile`, danach in allen übergeordneten Verzeichnissen.

`vagrant` wird in der Regel ohne `root`-Rechte ausgeführt. `vagrant -h` liefert eine Liste aller Kommandos. `vagrant kommando -h` zeigt weiterführende Informationen zum betreffenden Kommando an.

VagrantFile

Die Datei `VagrantFile` beschreibt die Konfiguration der virtuellen Maschine, die Vagrant einrichten soll. Im einfachsten Fall sind drei Zeilen ausreichend, die einfach den Ort der zugrunde liegenden Vagrant-Box auf dem Hashicorp-Server angeben. `"2"` bedeutet, dass die Vagrant-Datei die Syntax von Version 2 verwendet. `config.vm.box` gibt den Namen der Box an. Vagrant sucht üblicherweise im Hashicorp-Katalog nach der Box und lädt sie von dort herunter. Wenn sich die Box auf einem anderen Server oder in einem lokalen Verzeichnis befindet, geben Sie diesen Ort zusätzlich mit `config.vm.box_url` an. Für lokale Dateien verwenden Sie dabei die Syntax `"file:///pfad/name.box"`.

```
# VagrantFile für ubuntu/xenial64
Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/bionic64"
end
```

Nicht explizit in `VagrantFile` aufgeführt sind die Operationen zum Einrichten der Port-Umleitung für den SSH-Server sowie für die Synchronisation des gemeinsamen Verzeichnisses. Darum kümmert sich Vagrant automatisch, sofern dies nicht durch anderslautende Optionen verhindert wird.

Im Folgenden stelle ich Ihnen exemplarisch einige Optionen für `VagrantFile` vor. Eine vollständige Referenz finden Sie in der Vagrant-Dokumentation:

https://www.vagrantup.com/docs/vagrantfile

VagrantFile-Änderungen aktivieren

Einfache Änderungen an `VagrantFile` werden wirksam, wenn Sie die virtuelle Maschine einfach nur neu starten:

```
root#  vagrant reload
```

Alle mit `config.vm.provision` definierten Konfigurationsarbeiten erfordern aber die Option `--provision`. Damit erzwingen Sie eine neuerliche Konfiguration der virtuel-

len Maschine. (Normalerweise wird das sogenannte Provisioning ja nur beim ersten Start durchgeführt.)

```
user$  vagrant reload --provision
```

Alternativ können Sie das Provisioning auch im laufenden Betrieb durchführen bzw. wiederholen. Dabei können Sie mit der Option `--provision-with` einschränken, welchen Typ von Provisioning-Maßnahmen (z. B. `shell` oder `file`) bzw. welche benannte Provisioning-Anweisung Sie ausführen möchten:

```
user$  vagrant provision --provision-with shell
```

Bei komplexen Änderungen, für die Vagrant in der richtigen Reihenfolge mehrere Scripts ausführen muss, kann es sogar erforderlich sein, dass Sie die virtuelle Maschine mit `vagrant destroy` löschen und dann vollständig neu einrichten müssen:

```
user$  vagrant destroy
user$  vagrant up
```

Hostname

`config.vm.hostname` legt den Hostnamen der virtuellen Maschine fest:

```
config.vm.hostname = "vagrant-u1804"
```

Portumleitung

Mit `config.vm.network` können Sie diverse Parameter der Netzwerkkonfiguration verändern. Die folgende Zeile bewirkt eine Port-Umleitung vom Port 80 der virtuellen Maschine auf den Port 8080 des Hosts:

```
config.vm.network "forwarded_port", guest: 80, host: 8080
```

Gemeinsame Verzeichnisse

Standardmäßig teilt Vagrant das Projektverzeichnis, also das Verzeichnis des Hosts, in dem sich `Vagrantfile` befindet, im Gast als `/vagrant`. Bei Bedarf können Sie das verhindern:

```
config.vm.synced_folder ".", "/vagrant", disabled: true
```

Umgekehrt können Sie mit `config.vm.synced_folder` weitere gemeinsame Verzeichnisse einrichten. Dabei bezieht sich der erste Parameter auf den Host-Rechner (relativ zum Projektverzeichnis), der zweite Parameter auf den Gast:

```
config.vm.synced_folder "html/", "/var/www/html"
```

VirtualBox-spezifische Konfiguration

Wenn Sie Parameter der VirtualBox-Konfiguration verändern möchten, müssen Sie dazu einen eigenen `config.vm.provider`-Block definieren. Das folgende Listing gibt dafür drei Beispiele:


```
Vagrant.configure("2") do |config|
  ...
  config.vm.provider "virtualbox" do |vb|
    # RAM in MiB für die virtuelle Maschine (Default: laut Box)
    vb.memory = 1024
    # CPU-Cores (Default: laut Box)
    vb.cpus = 2
    # beim Start VirtualBox-Fenster anzeigen (Default: false)
    vb.gui = true
  end
end
```


Scripts

Mit `vagrant.vm.provision "shell" ...` erreichen Sie, dass Vagrant im Zuge des Provisionings das angegebene Script mit `root`-Rechten in der virtuellen Maschine ausführt. Kleinere Scripts können Sie direkt als Zeichenkette mit dem Schlüsselwort `inline` angeben:

```
config.vm.provision "shell", inline: "echo $(date)"
```

Auch mehrzeilige Scripts können Sie direkt in die Vagrant-Datei einbetten:

```
$myscript = <<END
apt-get update
apt-get install -y joe
END

Vagrant.configure("2") do |config|
  ...
  config.vm.provision "shell", inline: $myscript
end
```

Längere Scripts sind besser in eigenen Dateien untergebracht. Diese können Sie z. B. direkt im Vagrant-Projektverzeichnis speichern. Mit `path` geben Sie einfach den relativen Ort der Datei an. Anders als bei lokal auszuführenden Scripts ist es übrigens nicht erforderlich, die Datei mit `chmod a+x` ausführbar zu machen.

```
config.vm.provision "shell", path: "my-long-script.sh"
```

Wenn Sie in die Vagrant-Datei mehrere Scripts einbauen, ist es zweckmäßig, diese zu benennen. Dabei gilt die folgende Syntax:

```
config.vm.provision "script1", type: "shell", inline: "echo $(date)"
config.vm.provision "script2", type: "shell", inline: $myscript
```

Das hat zwei Vorteile: Zum einen können Sie damit die Ausgaben des Vagrant-Kommandos klarer einzelnen Scripts zuordnen, zum anderen ist es so möglich, nur ein bestimmtes Script auszuführen:

```
user$  vagrant provision --provision-with script2
```

Scripts für virtuelle Windows-Maschinen müssen übrigens in der Syntax von PowerShell formuliert werden – aber die PowerShell ist in diesem Buch ohnedies kein Thema.

Beispiel: CentOS-Webserver

Der Ausgangspunkt für das folgende Beispiel ist die Box `centos/7` aus dem Hashicorp-Katalog. (Als ich dieses Kapitel verfasst habe, gab es noch keine CentOS-8-Box.) Ähnlich wie mit `ubuntu/xenial64` erhalten Sie damit eine minimale Server-Installation: Die Box ist ca. 400 MiB groß und kompatibel mit vier Providern: VirtualBox, VMWare Workstation, VMWare Fusion und libvirt. Unter VirtualBox beansprucht `centos/7` anfänglich ca. 1 GiB Platz im Verzeichnis der VirtualBox-Maschinen. Die Eckdaten und einige Konfigurationsdetails sind hier dokumentiert:

https://app.vagrantup.com/centos/boxes/7

Um die Maschine im originalen Zustand einzurichten, führen Sie die folgenden Kommandos aus:

```
user$  mkdir centos7
user$  cd centos7
user$  vagrant init centos/7
user$  vagrant up
```

Im Gegensatz zu `ubuntu/bionic64` sind in der CentOS-Maschine die VirtualBox-Gasterweiterungen nicht standardmäßig installiert. Die Synchronisation des gemeinsamen Vagrant-Verzeichnisses erfolgt daher mit `rsync` beim Start der virtuellen Maschine. Beachten Sie, dass die Synchronisation einseitig ist: Es werden Dateien vom Host zum Gast übertragen, aber keine Änderungen vom Gast zurück zum Host synchronisiert.

Minimal ist auch die Vagrant-Datei von `centos/7`. Ohne Kommentare verbleiben nur drei Zeilen:

```
Vagrant.configure("2") do |config|
  config.vm.box = "centos/7"
end
```

Webserver installieren und starten

Das Ziel dieses Beispiels ist es, in der virtuellen Maschine automatisiert einen Webserver einzurichten. Dazu erstellen Sie im Vagrant-Projektverzeichnis die folgende Script-Datei:


```
#!/bin/bash
# Datei /home/kofler/centos7/install-webserver.sh
yum install -y httpd
systemctl enable httpd
systemctl start httpd
```


Anschließend ergänzen Sie die Vagrant-Datei um die folgenden beiden Zeilen:

```
config.vm.provision "shell", path: "install-webserver.sh"
config.vm.network "forwarded_port", guest: 80, host: 8080
```

Die erste Zeile gibt an, dass das Script `install-webserver.sh` im Rahmen des Provisionings in der virtuellen Maschine mit `root`-Rechten ausgeführt werden soll. Die zweite Zeile leitet den Port 80 der virtuellen Maschine auf den lokalen Port 8080 um, sodass der neu installierte Webserver direkt auf dem Hostrechner ausprobiert werden kann.

Um die Installation durchzuführen, starten Sie die virtuelle Maschine mit der Option `--provision` neu. Dabei werden sämtliche Ausgaben des `yum`-Kommandos angezeigt. Im folgenden Listing habe ich die Ausgaben aus Platzgründen stark gekürzt:

```
user$  vagrant reload --provision
==> default: Running provisioner: shell...
    default: Running: script
...
==> default: Install  1 Package (+4 Dependent packages)
...
==> default: Created symlink
    from /etc/systemd/system/multi-user.target.wants/httpd.service
    to /usr/lib/systemd/system/httpd.service.
```

Um den Webserver auszuprobieren, öffnen Sie auf dem Hostrechner in einem Webbrowser die Seite *http://localhost:8080*. Sie sollten darin die Testseite des Webservers sehen.

Eigene HTML-Dateien

Wenn Sie anstelle der Testseite eigene Webseiten anzeigen möchten, können Sie im Vagrant-Projektverzeichnis ein Unterverzeichnis mit den gewünschten HTML-Dateien einrichten:

```
user$  mkdir html
user$  cat > html/index.html << END
> <html>
> <body>
> <h1>Hello World!</h1>
> </body>
> </html>
> END
```

Damit alle Dateien aus dem lokalen `html`-Verzeichnis mit dem Verzeichnis `/var/www/html` in der virtuellen Maschine synchronisiert werden, ist die folgende Ergänzung in `VagrantFile` erforderlich. Die erste Zeile deaktiviert die Default-Synchronisierung mit dem Verzeichnis `/vagrant` in der virtuellen Maschine, die zweite Zeile richtet eine neue Synchronisierungsregel für `/var/www/html` ein:

```
config.vm.synced_folder ".", "/vagrant", disabled: true
config.vm.synced_folder "html/", "/var/www/html", type: "rsync"
```

Beachten Sie, dass dieses Beispiel nur statische Webseiten berücksichtigt. Wenn Sie eine dynamische Webseite einrichten möchten, müssen Sie in der virtuellen Maschine auch einen Datenbank-Server sowie geeignete Apache-Erweiterungen installieren, also z. B. MySQL und PHP. Dazu ist ein komplexeres Provisioning-Script erforderlich.