

Michael Kofler

Linux

Das umfassende Handbuch

30 Jahre
Linux-
Handbuch

- ▶ Von der Bash bis zum KI-Server: Das Standardwerk für alle Linux-Themen
- ▶ Desktop und Server installieren, konfigurieren und administrieren
- ▶ Für den Einstieg, die Fortbildung und den professionellen Einsatz

19., aktualisierte Auflage

 **Rheinwerk**
Computing

Vorwort

Linux begann als Hobby-Projekt des finnischen Programmierers Linus Torvalds und dominiert heute viele Segmente des IT-Markts. Dieser Siegeszug ist nicht jedem bewusst: Milliarden Menschen verwenden Android-Smartphones, nutzen Server der Cloud-Infrastruktur, WLAN-Router und IoT-Geräte, ohne zu wissen, dass auf fast allen diesen Geräten Linux läuft.

Die schwache öffentliche Wahrnehmung hat mit dem Versagen im Desktop-Segment zu tun: Die Masse ärgert sich mit Windows herum, eine Minderheit leistet sich Geräte mit macOS. Und Linux? Läuft (fast) nur auf den Notebooks von Technikstudenten, Wissenschaftlerinnen oder Administratoren.

Der ausbleibende Erfolg am Desktop hat viele Gründe: Linux hat nicht *ein* Desktop-System, sondern mehrere. Wegen dieser Zersplitterung funktioniert kein System perfekt. Probleme gibt es auch bei der Hardware: Linux läuft zwar auf vielen Geräten problemlos, aber gerade neue Notebooks bereiten oft Ärger: Auf dem einen Rechner funktioniert Bluetooth nicht richtig, auf einem anderen macht die Installation des Grafiktreibers Probleme usw.

Nun will ich Sie natürlich nicht schon im Vorwort abschrecken! Im Gegenteil, in diesem Buch zeige ich Ihnen, wie effizient Sie mit Linux arbeiten können. Linux vereint Vielseitigkeit und Sicherheit in einem Ausmaß, mit der andere Betriebssysteme schwer mithalten können.

Der Umstieg auf Linux ist zugegebenermaßen unbequem. Sie müssen sich an neue Regeln und Arbeitsweisen gewöhnen. Vieles ist aber nur eine Frage der Perspektive: Ich wundere mich oft, wie merkwürdig dieses oder jenes Detail in Windows gelöst ist. Mir sind die Linux-Techniken eben vertrauter.

Warum Linux?

Aus privater Perspektive spricht die Unabhängigkeit für Linux: Kein milliardenschweres Unternehmen entscheidet, wie lange Sie Ihren Rechner benutzen können/dürfen. Sie entscheiden selbst und verlängern so die Nutzungsdauer Ihrer Geräte. Außerdem benötigen Sie zur Installation keine Microsoft-, Apple- oder Google-ID.

Beim Server-Einsatz gibt es auch finanzielle Argumente: Linux ist kostenlos, ebenso wie die darauf ausgeführten Programme. Das reduziert nicht nur die Kosten, sondern auch den Ärger mit der Verwaltung und Aktivierung komplizierter Lizenzen.

In vielen Unternehmen steht das Sicherheitsargument an erster Stelle. Natürlich hat auch Linux Sicherheitsprobleme, aber es sind definitiv weniger als unter Windows. Eine Infrastruktur, die nicht zu 100 Prozent auf Windows setzt, kann im Fall eines erfolgreichen Angriffs nicht zu 100 Prozent zusammenbrechen wie ein Kartenhaus.

30 Jahre »Linux – Das umfassende Handbuch«

Das Linux-Buch feiert mit der hier vorliegenden Auflage sein 30-jähriges Jubiläum. Als ich an der ersten Auflage dieses Buchs schrieb, hatten die meisten Privatanwenderinnen und -anwender noch gar keinen Internetzugang, und wenn doch, dann über ein unzuverlässiges Modem. Das World Wide Web war gerade im Entstehen. In der ersten Auflage des Buchs habe ich *Mosaic* als Linux-tauglichen Webbrowser empfohlen. Erst in der zweiten Auflage konnte ich auf *Netscape* eingehen, der damals als erster »richtiger« Browser kostenlos für Linux zur Verfügung stand. (Aus Netscape wurde später Mozilla Firefox.)

Das wichtigste Medium zur Verbreitung von Linux war in den 90er-Jahren die CD. Die ersten Auflagen dieses Buchs enthielten deswegen CDs (später DVDs) mit Linux-Distributionen. Der Siegeszug des Internets veränderte den Charakter dieses Buchs: Es war nicht mehr notwendig, alle Optionen diverser Kommandos bis ins letzte Detail zu beschreiben; jetzt reicht ein Link auf eine Webseite mit vertiefenden Informationen.

In den Vordergrund des Buchs rückte die Vermittlung von Unix/Linux-Grundlagen. Ja, alles steht im Internet, aber Leserinnen und Leser schätzen den geordneten Überblick über Linux, die strukturierte Sammlung von Basiswissen so sehr, dass sich regelmäßige Neuauflagen lohnen. Die altmodische, aber werbefreie Präsentation auf Papier (oder in einem E-Book), frei von blinkenden Bannern und lästigen Werbevideos, ist ein entscheidender Vorteil.

Parallel zur Entwicklung des Internets bekam dieses Buch einen neuen inhaltlichen Fokus. Immer mehr Organisationen und Firmen betreiben selbst Linux-Server, sei es zu Hause, auf gemieteten Root-Servern oder in virtuellen Maschinen (Cloud-Instanzen). Dementsprechend erklärt dieses Buch, wie Sie selbst SSH-, Web-, Mail- und Datenbank-Server einrichten und sicher betreiben.

Ein weiterer Meilenstein der Linux-Geschichte war die Vorstellung des Raspberry Pi vor gut einem Jahrzehnt. Dieser preisgünstige Linux-basierte Mini-Computer bietet viele Anwendungen, von elektronischen Bastelprojekten bis zur Basisstation für die eigene Home Automation.

Seit 2022 zeichnet sich eine weitere IT-Zeitenwende ab: Mit ChatGPT wird der erste brauchbare KI-Chatbot frei verfügbar. Es lässt sich trefflich darüber streiten, wie »intelligent« dieses und konkurrierende Systeme sind. Fest steht, dass KI-Tools eine großartige Hilfe bei Linux-Problemen aller Art sind.

Nachdem ich mich vor zwei Jahrzehnten gefragt habe, ob das Internet IT-Bücher obsolet macht, stellt sich diese Frage jetzt wieder. Und neuerlich glaube ich, dass dies nicht der Fall sein wird. KI-Tools helfen (auch) bei der Lösung von Linux-Problemen. Dennoch brauchen Sie einiges an Grundwissen, um funktionierende Prompts zu formulieren. Und noch mehr Wissen zur Abschätzung, ob die Antworten von ChatGPT & Co. plausibel, veraltet oder frei erfunden sind. Genau dieses Wissen vermittele ich hier – seit 30 Jahren!

Neu in dieser Auflage

Dieses Buch ist mit der ersten Auflage kaum noch zu vergleichen. Auch wenn sich das Buch mit jeder Auflage nur ein kleines Stück geändert hat, hat es – so wie Linux in seiner Gesamtheit – in 30 Jahren einen weiten Weg zurückgelegt.

Für diese Auflage habe ich das Buch einmal mehr vom ersten bis zum letzten Kapitel aktualisiert. Ich habe die aktuellsten Distributionen ausprobiert und Konfigurationsanleitungen an neue Versionen angepasst. Gleichzeitig habe ich das Buch an vielen Stellen gestrafft und von Altlasten befreit. Das hat Platz für neue Inhalte gemacht, z. B. rund um die folgenden Themen:

- ▶ Die Shell fish (neues Kapitel)
- ▶ Swap on ZRAM
- ▶ Geoblocking mit nft (neuer Abschnitt)
- ▶ Samba im Zusammenspiel mit Windows 11 24H2
- ▶ Monitoring mit Prometheus und Grafana (neues Kapitel, Docker-Setup mit Traefik)
- ▶ KI-Sprachmodelle ausführen (neues Kapitel)
- ▶ Berücksichtigung von CachyOS

The Linux way to do it

Mit diesem Buch lernen Sie Linux von Grund auf. Die Themenpalette reicht von der Installation von Linux auf einem Notebook über die Desktop-Anwendung bis hin zum Server- und Cloud-Einsatz.

Besonders wichtig ist mir, dass Sie Linux nicht nur anwenden, sondern auch verstehen lernen: Ausführliche Grundlagenkapitel erklären, wie Sie Linux im Terminal bedienen, wie Sie Linux optimal konfigurieren und warum Linux so funktioniert. Nach der Lektüre dieser Kapitel kennen Sie nicht nur Linux an sich, sondern auch die Philosophie von Unix/Linux – also gewissermaßen *the Linux way to do it*.

Je mehr Sie sich in die Linux-Welt einarbeiten, desto mehr wird Linux *Ihr* Betriebssystem. Ich wünsche Ihnen viel Freude beim Experimentieren und Arbeiten mit Linux!

Michael Kofler (<https://kofler.info>)

Konzeption

Das Buch ist in acht Teile gegliedert:

- ▶ **Teil I** erklärt, was Linux eigentlich ist, und vermittelt das Grundlagenwissen, das Sie für eine optimale und sichere **Installation** brauchen. Hier finden Sie konkrete Installationsanleitungen für etliche Distributionen: Debian, Fedora, Ubuntu und CachyOS.
- ▶ **Teil II** behandelt Linux auf dem **Desktop**. Hier lernen Sie die Desktop-Systeme Gnome und KDE kennen. Außerdem stelle ich Ihnen Programme vor, um E-Mails und Fotos zu verwalten und Audio-Dateien und Filme abzuspielen. Ein umfassendes Kapitel zum Minicomputer Raspberry Pi zeigt Ihnen, wie Sie diesen als Plattform für Bastelprojekte einsetzen.
- ▶ In **Teil III** lernen Sie das **Terminal** kennen. In mehreren Kapiteln lernen Sie, mit welchen Kommandos Sie das Dateisystem durchsuchen, wie Sie Dokumente und Bilder in andere Formate konvertieren und wie Sie den Kommandointerpreter bash nutzen.
- ▶ In **Teil IV** stehen verschiedene **Texteditoren** im Mittelpunkt. Neben den Urgesteinen Vi und Emacs stelle ich Ihnen auch Visual Studio Code ausführlich vor.
- ▶ **Teil V** widmet sich der **Systemkonfiguration**. Egal, ob es gerade bei Ihrer Hardware Probleme gibt oder ob Sie ganz besondere Anforderungen stellen – hier erfahren Sie, wie Sie das Dateisystem administrieren, das Grafiksystem konfigurieren, Software-Pakete installieren und aktualisieren, den Systemstart konfigurieren sowie den Kernel und seine Module einrichten bzw. neu kompilieren.
- ▶ In **Teil VI** geht es um den Einsatz von **Linux als Server**. Ich erläutere die Server-Installation von Red Hat Enterprise Linux (RHEL) und dessen Klonen sowie von Debian und Ubuntu Server, sowohl auf physischen Rechnern als auch in der Cloud. Im Anschluss zeige ich Ihnen die Konfiguration wichtiger Server-Dienste, unter anderem: SSH, Apache, MySQL/-MariaDB, Postfix und Dovecot, Nextcloud und Samba.
- ▶ **Teil VII** hat verschiedene Aspekte der **Sicherheit** zum Thema. Dort erfahren Sie, wie Sie Backups durchführen und Ihre Server durch Firewalls, Monitoring, SELinux oder AppArmor schützen.
- ▶ **Teil VIII** beschäftigt sich mit verschiedenen Arten der **Virtualisierung**: Hier lernen Sie VirtualBox sowie das Server-Virtualisierungssystem KVM kennen. Ein weiteres Kapitel stellt die Container-Systeme Docker und Podman vor. Mit dem Windows Subsystem for Linux (WSL) integrieren Sie Linux *innerhalb* von Windows. Ein ganz neues Kapitel beschreibt, wie Sie unter Linux lokale KI-Sprachmodelle ausführen.

Formales

Wenn in Listings Kommandos und deren Ergebnisse dargestellt werden, befindet sich dazwischen meistens eine leere Zeile. Die Ergebnisse werden immer eingerückt:

```
ls *.md

    vorwort.md intro.md ...
```

Manche Kommandos können nur vom Systemadministrator `root` ausgeführt werden. In diesem Fall ist dem Kommando `sudo` vorangestellt. Sie können das Kommando wie angegeben ausführen. Wenn Sie mehrere Kommandos mit `root`-Rechten ausführen, ist es aber bequemer, vorübergehend mit `sudo -s` eine `root`-Shell zu öffnen. Hintergrundinformationen zu dem äußerst wichtigen Kommando `sudo` finden Sie in Abschnitt 13.3, »`sudo`«.

```
sudo systemctl restart apache2
```

Falls einzelne Kommandos nicht in einer Zeile Platz finden, werden sie mit dem Zeichen `\` auf zwei oder mehr Zeilen verteilt. `\` ist ein unter Linux zulässiges Zeichen, um mehrzeilige Kommandoangaben durchzuführen. Sie können das Kommando aber natürlich auch einzellig ohne `\` eintippen.

```
sudo nft add chain inet mytable mychain \
    '{ type filter hook input priority 0; }'
```

Oft folgen dem eigentlichen Kommando Kommentare, die – der Syntax der Shell `bash` entsprechend – mit dem Zeichen `#` eingeleitet werden:

```
sudo apt install openssh-server      # gilt für Debian und Ubuntu
sudo dnf install openssh-server      # gilt für Fedora und RHEL
```

Distributionen

Die unzähligen Linux-Distributionen lassen sich nach ihrer Herkunft zu Familien zusammenfassen. Viele Informationen in diesem Buch gelten deswegen für mehrere Distributionen:

- ▶ Arch Linux: gilt auch für CachyOS und Manjaro
- ▶ Debian: gilt größtenteils auch für Raspberry Pi OS, teilweise auch für Ubuntu
- ▶ RHEL/Fedora: RHEL und Fedora sind miteinander verwandt, die meisten Details gelten wechselseitig. Alle RHEL-Informationen gelten auch für alle Klone, z. B. für AlmaLinux, Oracle Linux und Rocky Linux.
- ▶ Ubuntu: gilt größtenteils auch für Kubuntu, Linux Mint, Pop!_OS etc.

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.

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.

Schon lange vor Windows, Linux und den Smartphones 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, stellte hohe Hardware-Anforderungen und lief nur auf teuren Workstations.

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>).

Genau genommen bezeichnet der Begriff Linux nur den Kernel: Er ist der innerste Teil (also der Kern) eines Betriebssystems mit ganz elementaren Funktionen, wie Speicherverwaltung, Prozessverwaltung und Steuerung der Hardware. Die Informationen in diesem Buch beziehen sich auf den Kernel 6.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:

- **CPUs:** Linux ist kompatibel zu den meisten marktüblichen CPUs. Das gilt auch für ARM-CPU's. Linux läuft auf Milliarden von Android-Smartphones und Millionen von Raspberry Pis!

Zu den Ausnahmen zählen aber die 2024 vorgestellten Notebooks auf der Basis von Qualcomm-ARM-CPU's (z. B. »Snapdragon X«). Auf solchen Geräten läuft Linux zwar prinzipiell, für den Praxisbetrieb gibt es (Stand Mitte 2025) aber noch zu viele Probleme. Lesenswert ist dieser Testbericht:

<https://www.phoronix.com/review/snapdragon-x-elite-linux-benchmarks>

Ähnliche Einschränkungen gelten für Apple-Notebooks mit den CPU's M1, M2 usw. Asahi Linux (<https://asahilinux.org>) hat sich den Betrieb von Linux auf modernen Apple-Geräten zum Ziel gesetzt. Stand Mitte 2025 ist Asahi Linux aber nur mit den ersten beiden CPU-Generationen kompatibel und auch dann mit vielen Einschränkungen verbunden.

- **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- oder AMD-CPU's 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 Bluetooth-Adapter:** Für relativ neue WLAN-, LAN- und Bluetooth-Controller gibt es mitunter keine Linux-Treiber.
- **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 zudem Testberichte der Zeitschrift c't:

Deren Redakteure machen sich bei den meisten Geräten die Mühe, auch die Linux-Kompatibilität zu testen.

Lieber etwas älter

Um ganz neue Notebooks mache ich beim Kauf in der Regel einen großen Bogen, auch wenn die Spezifikationen noch so verlockend sind. Sie verursachen allzu oft Treiberprobleme und verursachen Zusatzarbeit bei Installation und Konfiguration. Sie sparen Geld und vermeiden es, sich beim Konfigurieren zu ärgern, wenn Sie sich für ein Vorjahresmodell entscheiden!

1.3 Distributionen

Noch immer ist die einleitende Frage – Was ist Linux? – nicht ganz beantwortet. Viele Anwender interessiert der Kernel nämlich herzlich wenig. Für sie umfasst der Begriff Linux, wie er umgangssprachlich verwendet wird, neben dem Kernel auch das riesige Bündel mitgelieferter Programme: Dazu zählen unzählige Kommandos, ein Desktop-System (z. B. KDE oder Gnome), LibreOffice, Firefox, GIMP sowie zahllose Programmiersprachen und Server-Programme (Webserver, Mail-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.

- **Zielpattform (CPU-Architektur):** Viele Distributionen sind nur für x86- und ARM-kompatible Prozessoren erhältlich. Es gibt aber auch Distributionen für andere Prozessorplattformen (SPARC etc.). Besonders viele Plattformen unterstützt Debian.
- **Lizenz:** Die meisten Distributionen sind kostenlos erhältlich. Bei einigen Distributionen gibt es 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 rundherum.

Sie können eine Linux-Distribution nur so lange sicher betreiben, wie Sie Updates bekommen. Danach sollten Sie auf eine neue Version der Distribution wechseln. Deswegen ist es bedeutsam, wie lange es für eine Distribution Updates gibt. Hier gilt meist die Grundregel: je teurer der kommerzielle Support, desto länger der Zeitraum (siehe Tabelle 1.1).

Distribution	Wartungszeitraum
Debian	3 Jahre (mit Einschränkungen 5 Jahre)
Fedora	13 Monate
Red Hat Enterprise Linux (RHEL)	10 Jahre (mit Einschränkungen 13 Jahre)
RHEL-Klone	bis zu 10 Jahre
SUSE Enterprise Server	10 Jahre (mit Einschränkungen 13 Jahre)
Ubuntu LTS	3 bis 5 Jahre
Ubuntu LTS mit Pro-Upgrade	10 Jahre (mit Einschränkungen 12 Jahre)
Ubuntu (sonstige Versionen)	9 Monate

Tabelle 1.1 Maximale »Lebenszeit« verschiedener Linux-Distributionen (Stand: Sommer 2025)

Live-System

Manche Distributionen ermöglichen den Linux-Betrieb direkt von einem USB-Stick. Das ermöglicht ein einfaches Ausprobieren. Außerdem bieten derartige Live-Systeme eine gute Möglichkeit, um ein defektes Linux-System zu reparieren bzw. die betreffende Distribution neu zu installieren.

Rolling-Release-Konzept

Bei den meisten Linux-Distributionen werden mit den regulären Updates nur Sicherheitsprobleme und offensichtliche Fehler behoben, aber keine Versionssprünge durchgeführt. Bei-

spielsweise wurde Ubuntu 24.04 mit dem Versionsverwaltungsprogramm `git` in der Version 2.43 ausgeliefert. Sie können einen Ubuntu-Server mit Version 24.04 bis in die 2030er-Jahre sicher betreiben, aber `git` wird bei Version 2.43 bleiben. Zu einer neueren Version kommen Sie nur, wenn Sie alternative Paketquellen einrichten oder ein Upgrade auf eine neuere Distributions-Version durchführen (z. B. auf Ubuntu 26.04).

Es gibt aber auch Distributionen, die das *Rolling-Release*-Modell anwenden, z. B. Arch Linux oder openSUSE Tumbleweed: Dort erhalten Sie mit Updates stets die neueste Version *jeder* installierten Software-Komponente. Das klingt praktisch, kann aber zu Stabilitätsproblemen führen. Deswegen sind Rolling-Release-Distributionen im Server-Bereich unüblich. Sie sprechen eher fortgeschrittene Linux-Anwender an, die Software entwickeln oder Systeme administrieren und die kein Problem damit haben, nach einem Update die eine oder andere Konfigurationsdatei anzupassen, wenn etwas nicht mehr funktioniert.

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.

AlmaLinux ist ein RHEL-Klon, also eine zu Red Hat Enterprise Linux kompatible Distribution. AlmaLinux hat zusammen mit Rocky Linux die Nachfolge von CentOS Linux angetreten.

Android ist eine von Google entwickelte Plattform für Mobilfunkgeräte und Tablets. Android hat damit Linux zu der Welt dominanz verholfen, über die Linux-Entwickler in der Vergangenheit gescherzt haben. Android ist aber ungeeignet für eine PC-Installation und insofern keine »echte« Distribution.

Arch Linux ist eine für technische Anwender optimierte Rolling-Release-Distribution. Wegen der relativ komplizierten, im Textmodus durchzuführenden Installation machen Einsteigerinnen und Einsteiger zumeist einen großen Bogen um Arch Linux. Dafür zählen <https://wiki.archlinux.org> und <https://wiki.archlinux.de> zu den besten Quellen für Linux-Konfigurationsdetails im Netz. ArchLinux ist auf x86-Architekturen fokussiert. Es gibt zwar eine ARM-Variante, diese war zuletzt aber in keinem guten Zustand.

Die Arch-Linux-Derivate **CachyOS**, **Manjaro** und **EndeavourOS** mit grafischen Installations- und Konfigurationsprogrammen haben Arch Linux in der Top-10-Liste von *distrowatch.com* verankert.

CentOS war eine kostenlose Variante zu Red Hat Enterprise Linux (RHEL) und hatte eine riesige Installationsbasis. Allerdings hat Red Hat im Dezember 2020 das Ende von CentOS in seiner bisherigen Form verkündet. Sein Nachfolger **CentOS Stream** unterscheidet sich in zwei wichtigen Details vom ursprünglichen CentOS: Zum einen ist der Wartungszeitraum wesentlich kürzer und beträgt nur 4 bis 5 Jahre anstelle von bisher 10 Jahren; zum anderen werden die meisten Paket-Updates (ausgenommen sind Sicherheits-Updates, die einem *Non-*

disclosure Agreement unterliegen) zuerst für CentOS freigegeben, bevor sie für RHEL zum Einsatz kommen. Das scheint auf den ersten Blick ein Vorteil zu sein. Tatsächlich geht damit aber die vollständige Kompatibilität zu RHEL verloren. CentOS Stream ist für den längerfristigen Produktiveinsatz ungeeignet.

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.

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 nicht die neuesten Programmversionen. Dafür steht Debian für neun Hardware-Plattformen zur Verfügung. Viele andere Distributionen sind von Debian abgeleitet, z. B. Raspberry Pi OS und Ubuntu.

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.

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 Sicherheitsexperten.

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.

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

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 AlmaLinux, CentOS Stream, Fedora, Oracle Linux oder Rocky Linux an.

SUSE ist nach diversen Übernahmen die wichtigste deutsche Linux-Firma. Ihr Hauptprodukt, **SUSE Enterprise**, ist vor allem im europäischen Markt verankert. **openSUSE** ist eine kostenlose Variante, die sich aber speziell an Privatanwender und Entwicklerinnen wendet.

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 Versionen mit Long Time Support (LTS) gibt es sogar 3 bis 5 Jahre lang Updates.

Für kommerzielle Kunden bietet die Firma Canonical diverse Support-Angebote, unter anderem **Ubuntu Pro**: Dieses zahlungspflichtige Upgrade erweitert den Update-Zeitraum von LTS-Versionen auf zehn Jahre. Canonical hat sich damit vor allem im Server- und Cloud-Sektor zu den weltweit wichtigsten Linux-Firmen entwickelt.

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 ist insofern bei KDE-Fans beliebt.

Mini-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 **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 oder Ubuntu zu entscheiden. 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, Ubuntu (mit Pro-Abo) und SUSE. Andernfalls sind AlmaLinux, Debian, Rocky Linux und Ubuntu (ohne Abo) attraktive kostenlose Alternativen.

Linux Standard Base

Egal, für welches Linux Sie sich entscheiden – es gibt einen riesigen gemeinsamen Nenner. Auch wenn jede Distribution ihre Eigenheiten hat, gibt es noch viel mehr Gemeinsamkeiten. Das Linux-Standard-Base-Projekt (LSB) definiert dafür die Regeln. Fast alle Distributionen sind LSB-konform:

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

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) und Lesser General Public License (LGPL)

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 gegrün-

det, um hochwertige Software frei verfügbar zu machen. Richard Stallman ist übrigens auch der Autor des Editors Emacs, der in Kapitel 18 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>

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! Der Kernel und diverse Linux-Tools unterliegen der GPL, aber für andere Komponenten und Programme gilt eine Fülle anderer Lizenzen.

- **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 manche 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.

- **Business Source Licenses (BSL) und Server Side Public Licenses (SSPL):** Traditionelle Doppellizenzen beschränken die Weitergabe von Software. Bei *Software as a Service* (SAAS) kommt es aber nie zu einer Weitergabe. Für Server-Software ist das zum Problem geworden (z. B. für Datenbanksysteme wie MongoDB). SAAS-Firmen verdienen Geld mit Dienstleistungen auf der Basis von Software, nicht nur ohne zu zahlen, sondern auch ohne jede Notwendigkeit, sich an der Weiterentwicklung des Codes zu beteiligen. Dieses Geschäftsmodell wird von vielen Entwicklerinnen und Entwicklern als unfair betrachtet.

Abhilfe schaffen Lizenzen, bei denen der Code zwar öffentlich publiziert wird, seine Anwendung aber an Bedingungen geknüpft wird. Bei manchen Lizenzen wird der Quellcode nach einer längeren Zeitspanne (z. B. nach vier oder fünf Jahren) automatisch frei von allen Einschränkungen. Nach dem OSI-Standard (siehe den folgenden Kasten) handelt es sich dabei *nicht* um echte Open-Source-Produkte.

- **Kommerzielle Lizenzen:** Es gibt im Linux-Umfeld einige Treiber und Programme, die zwar kostenlos genutzt werden dürfen, deren Quellcode aber ein Firmengeheimnis bleibt. Die NVIDIA-Grafiktreiber sind ein Beispiel dafür. (NVIDIA hat zuletzt damit begonnen, »echte« Open-Source-Treiber zu entwickeln. Der Erfolg/Abschluss dieses Projekts bleibt abzuwarten.)

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. Groß ist 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. Eine gute Einführung in das Thema finden Sie hier:

<https://opensource.org>

<https://heise.de/-221957>

Was ist die Open Source Initiative (OSI)?

Die Organisation *Open Source Initiative* hat Kriterien dafür aufgestellt, was als Open-Source-Lizenz gilt und was nicht. Lizenzen, die *OSI approved* sind, entsprechen dem Ideal der Open-Source-Bewegung:

<https://opensource.org/licenses>

Lizenzkonflikte zwischen Open- und Closed-Source-Software

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.

Manche proprietären 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.

Die Frage ist allerdings, ob man der Open-Source-Idee mit dieser engen Auslegung der GPL-Regeln nützt oder schadet: Optimisten glauben, dass die Hardware-Firmen dadurch gezwungen wären, selbst Open-Source-Treiber zu entwickeln oder zumindest die erforderlichen Informationen an die Entwicklergemeinschaft freizugeben. Pessimisten befürchten, dass derartige Hardware dann unter Linux einfach nicht mehr nutzbar wäre, oft ohne gute Alternativen.

1.5 Die Geschichte von Linux

In den folgenden Punkten habe ich versucht, die wichtigsten Meilensteine aufzuzeigen, die zu dem Linux geführt haben, das wir heute kennen.

- **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. Stattdes-

sen 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 schrieben. 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 Tux zum Linux-Logo.
- ▶ **1998 – Microsoft sieht Linux als Bedrohung:** Mit dem rasanten Siegeszug des Internets stieg auch die Verbreitung von Linux, vor allem auf Servern. Gewissermaßen zum Ritter Schlag 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.
- ▶ **2000er-Jahre – Fundament für Internet und Cloud:** Der gemeinsame Nenner von Amazon (gegründet 1994), Google (1998), Wikipedia (2001), Facebook (2006) und Dropbox (2007) besteht darin, dass all diese Firmen bzw. Organisationen für ihren Server-Betrieb auf Linux setzen. Das Internet, wie es sich seit den 2000er-Jahren entwickelt, und die daraus entwachsene Cloud-Infrastruktur sind ohne Linux undenkbar.
- ▶ **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 Firma Red Hat für stattliche 34 Mrd. US-Dollar. Gleichzeitig wandte 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. Mit dem Kauf von GitHub, der ebenfalls 2018 über die Bühne ging, beherrscht Microsoft nun das wichtigste Repository für Open-Source-Projekte.

Ist seither nichts mehr passiert? Ja und nein! Linux und Open-Source-Software gehören mittlerweile so sehr zum IT-Mainstream, dass die markanten Meilensteine seltener wurden. Vielmehr hat sich Linux inkrementell weiterentwickelt und neue Marktsegmente erobert, beispielsweise in Form von Containern (Docker, Kubernetes) oder als Infrastruktur für KI-Firmen.

Eine technisch sehr spannende Entwicklung geht von **Immutable** oder **Atomic Distributions** aus, die ähnlich wie Container funktionieren. Beispiele sind *Vanilla OS*, *Fedora Silverblue*, *Image Mode for RHEL* oder *SUSE Adaptable Linux Platform*. Die ganze Distribution oder zumindest größere Komponenten sind dabei wie statische Blöcke zu sehen, die im Betrieb nie verändert werden. Für ein Update wird ein komplett neues Image vorbereitet und per Reboot gestartet. Die Konfigurationsdateien und Daten befinden sich außerhalb der eigentlichen Distribution in eigenen Dateisystemen.

Eine Sonderstellung nimmt NixOS ein, das durch ein funktionales Paketmanagement und vollständig deklarative Systemkonfiguration ähnliche Ziele verfolgt, technisch aber einen anderen Weg geht. Alle genannten Systeme ermöglichen einfache Rollbacks auf frühere Systemzustände und versprechen höhere Zuverlässigkeit. Der große Durchbruch solcher Systeme, am ehesten im Cloud-Segment, steht noch aus.

Kapitel 26

Kernel und Module

Dieses Kapitel beschäftigt sich mit dem Linux-Kernel und seinen Modulen. Module sind Teile des Kernels, die bei Bedarf geladen werden – etwa wenn eine bestimmte Hardware-Komponente zum ersten Mal angesprochen wird. Vorweg ein Überblick:

- ▶ **Kernelmodule:** Abschnitt 26.1 erklärt, warum Kernelmodule automatisch geladen werden und was Sie tun müssen, wenn dieser Automatismus versagt.
- ▶ **Device Trees:** Auf Smartphones und in Embedded Devices gelten für die Verwaltung der Kernelmodule ganz andere Voraussetzungen als auf PCs. Dort haben sich Device Trees zur Verwaltung von Kernelmodulen durchgesetzt. Eine Einführung in diese auch auf dem Raspberry Pi übliche Technik gibt Abschnitt 26.2.
- ▶ **Kernelmodule selbst kompilieren:** Wenn Sie spezielle Hardware einsetzen, müssen Sie eventuell ein eigenes Kernelmodul kompilieren. Wie das gelingt, verrät Abschnitt 26.3.
- ▶ **Den ganzen Kernel kompilieren:** Auch wenn eher selten die Notwendigkeit besteht, den ganzen Kernel neu zu kompilieren, beweist Abschnitt 26.4, dass dies durchaus keine Hexerei ist.
- ▶ **Kernel-Live-Patches:** Die Installation eines neuen Kernels wird erst mit einem Neustart wirksamer. Für Sicherheits-Updates ist es eleganter, diese im laufenden Betrieb durchzuführen. Abschnitt 26.5 stellt Mechanismen zur Aktivierung derartiger Live-Patches vor.
- ▶ **/proc- und /sys-Dateisystem:** Abschnitt 26.6 zeigt, wie Sie aus dem /proc- bzw. /sys-Dateisystem aktuelle Informationen über den Kernel ermitteln.
- ▶ **Kerneloptionen und -parameter:** Abschnitt 26.7 und Abschnitt 26.8 erklären, wie Sie während des Rechnerstarts Optionen an den Kernel übergeben bzw. im laufenden Betrieb Kernelparameter ändern.
- ▶ **Spectre, Meltdown & Co.:** Abschnitt 26.9 beschreibt schließlich, mit welchen Maßnahmen der Kernel Sie vor Sicherheitslücken in aktuellen CPUs schützt.

Dieses Kapitel richtet sich explizit an fortgeschrittene Linux-Anwenderinnen und -Anwender. Einsteiger sind gut beraten, nur den für ihre Distribution vorgesehenen Kernel zu verwenden! Die Informationen in diesem Kapitel gelten für alle Kernelversionen ab 4.n.

26.1 Kernelmodule

Der Kernel ist jener Teil von Linux, der für elementare Funktionen wie Speicherverwaltung, Prozessverwaltung, Zugriff auf SSDs und Netzwerkkarten etc. zuständig ist. Der Kernel verfolgt dabei ein modularisiertes Konzept: Anfänglich – also beim Hochfahren des Rechners – wird ein Basiskernel geladen, der nur jene Funktionen enthält, die zum Rechnerstart erforderlich sind.

Wenn im laufenden Betrieb Zusatzfunktionen benötigt werden, z. B. für spezielle Hardware, wird der erforderliche Code als Modul mit dem Kernel verbunden. Werden diese Zusatzfunktionen eine Weile nicht mehr benötigt, kann das Modul wieder aus dem Kernel entfernt werden. Dieses modularisierte Konzept hat viele Vorteile:

- ▶ Kernelmodule können nach Bedarf eingebunden werden. Wenn ein bestimmtes Modul auf Ihrem Rechner nicht benötigt wird, kann so Speicher gespart werden, d. h., der Kernel ist nicht größer als unbedingt notwendig und optimal an Ihre Hardware angepasst.
- ▶ Bei einer Änderung der Hardware (z. B. nach dem Einbau einer neuen Netzwerkkarte) muss kein neuer Kernel kompiliert, sondern nur das entsprechende Modul geladen werden.
- ▶ Bei der Entwicklung eines Kernelmoduls muss nicht ständig der Rechner neu gestartet werden. Es reicht, ein Modul neu zu kompilieren. Anschließend kann es bei laufendem Betrieb getestet werden.

Eine Menge Hintergrundinformationen zum Umgang mit Kernelmodulen finden Sie im Module-HOWTO-Dokument. Es ist zwar bald 20 Jahre alt, an den Prinzipien der Modulverwaltung hat sich seither aber wenig geändert.

<https://www.tldp.org/HOWTO/Module-HOWTO>

Module automatisch laden

Dafür, dass Kernelmodule tatsächlich automatisch geladen werden, sobald sie benötigt werden, ist die in den Kernel integrierte Komponente `kmod` verantwortlich. `kmod` wird durch die Dateien `/etc/modprobe.conf` und `/etc/modprobe.d/*.conf` gesteuert. Diese Konfigurationsdateien werden weiter unten genauer beschrieben.

In den Anfangstagen von Linux mussten der Kernel und seine Module exakt zusammenpassen: Es war nicht möglich, ein Modul zu laden, das für eine andere, vielleicht nur geringfügig veränderte Kernelversion kompiliert wurde. Aus diesem Grund gab und gibt es bis heute für jede Kernelversion ein eigenes Modulverzeichnis `/lib/modules/n.n`.

2006 hat man mit *Module Versioning* versucht, zusammen mit einem Modul Zusatzinformationen zu speichern, die Aufschluss darüber geben, ob eine Zusammenarbeit zwischen dem Modul und dem Kernel auch bei unterschiedlicher Versionsnummer möglich ist. Damit konnten nicht exakt zur Kernelversion passende Module genutzt werden. Dieser Mechanismus funktioniert allerdings nur, wenn es zwischen der Kernel- und der Modulversion keine

Änderungen an den Schnittstellen gegeben hat. In der Praxis hat sich dieser Mechanismus nicht besonders gut bewährt, weswegen seine Anwendung heute unüblich ist.

Kommandos zur Modulverwaltung

Alle gängigen Distributionen sind so eingerichtet, dass Module bei Bedarf automatisch geladen werden. Ein Beispiel: Sie binden mit `mount` das Dateisystem eines USB-Sticks in den Verzeichnisbaum ein. Daraufhin wird automatisch das `vfat`-Modul aktiviert, das zum Lesen des Dateisystems erforderlich ist.

Im Regelfall erfolgt die Modulverwaltung also automatisch und transparent, ohne dass Sie mit den im Folgenden beschriebenen Kommandos zur manuellen Modulverwaltung eingreifen müssen. Dennoch sollten Sie die Modulkommandos kennen, um Module zur Not auch manuell laden zu können.

Alle Module befinden sich im Verzeichnis `/lib/modules/n.n`. Dabei ist `n.n` die Version des laufenden Kernels. Moduldateien haben die Dateiendung `*.ko` bzw. `.ko.xz`, wenn sie mit `xz` komprimiert sind. Das Kommando `uname -r` liefert die Versionsnummer des laufenden Kernels:

```
uname -r

6.15.10-200.fc42.aarch64

ls /lib/modules/6.15.10-200.fc42.aarch64

config
dtb/
kernel/
modules.alias
modules.alias.bin
modules.block
modules.builtin
...
```

`modprobe` lädt das angegebene Modul in den Kernel. Dabei muss nur der Modulname angegeben werden. Zusätzlich können Parameter (Optionen) an das Modul übergeben werden. Hexadezimalen Werten müssen Sie `0x` voranstellen, also etwa `option=0xff`.

```
sudo modprobe cifs
```

Im Vergleich zum Low-Level-Kommando `insmod`, auf das ich hier nicht eingehe, agiert `modprobe` relativ intelligent:

- `modprobe` sucht die Moduldatei selbst im verzweigten Modulverzeichnisbaum des gerade aktiven Kernels. Bei Modulen, die schon beim Kompilieren in den Kernel integriert wurden, endet `modprobe` ohne Fehlermeldung.

- `modprobe` lädt gegebenenfalls auch alle Module, die als Voraussetzung für das gewünschte Modul benötigt werden.
- `modprobe` berücksichtigt alle in `/etc/modprobe*` angegebenen Modulooptionen.

`modprobe` setzt eine korrekte Modulkonfiguration durch die Dateien `/etc/modprobe*` und `/lib/modules/n.n/modules.dep` voraus.

`lsmod` liefert eine normalerweise recht lange Liste aller momentan geladenen Kernelmodule. Beachten Sie, dass `lsmod` in den Kernel einkompilierte Module nicht anzeigt, sondern nur solche Module, die nachträglich geladen wurden!

```
lsmod | sort
```

Module	Size	Used by
8250_dw	24576	0
ac97_bus	16384	1 snd_soc_core
acpi_pad	24576	0
acpi_thermal_rel	16384	1 int3400_thermal
aesni_intel	401408	12
...		

Sehr hilfreich bei der Zuordnung von Kernelmodulen ist das Kommando `lspci` mit der Option `-k` (*Kernel driver in use*). Es listet alle über den PCI-Bus angeschlossenen Hardware-Komponenten auf und zeigt an, welche Treiber geeignet wären, um das Gerät zu steuern, und welcher Treiber tatsächlich verwendet wird:

```
lspci -k
```

```
00:00.0 Host bridge: Intel Corporation 8th Gen ...  
  Subsystem: Lenovo 8th Gen Core Processor Host ...  
  Kernel driver in use: skl_uncore  
00:01.0 PCI bridge: Intel Corporation Xeon ...  
  Kernel driver in use: pcieport  
00:02.0 VGA compatible controller ...  
  Subsystem: Lenovo UHD Graphics 630 (Mobile)  
  Kernel driver in use: i915  
  Kernel modules: i915  
...
```

`modinfo` liefert eine Menge Informationen über ein Modul. Das Modul muss sich nicht im Kernel befinden.

```
sudo modinfo cifs
```

```
filename:      /lib/modules/6.15.10-200.fc42.aarch64/kernel/fs/smb/  
               client/cifs.ko.xz  
version:      2.54
```

```

description:   VFS to access SMB3 servers e.g. Samba ...
license:       GPL
author:        Steve French
...
parm:          cifs_min_rcv: Network buffers in pool.
                Default: 4 Range: 1 to 64 (uint)
parm:          cifs_min_small: Small network buffers in pool.
                Default: 30 Range: 2 to 256 (uint)
...

```

`rmmod` entfernt das angegebene Modul wieder aus dem Kernel und gibt den belegten Speicher frei. Das Kommando kann nur erfolgreich ausgeführt werden, wenn das Modul gerade nicht verwendet wird.

```
sudo rmmod cifs
```

Modulkonfiguration

Die Modulverwaltung funktioniert scheinbar wie von Zauberhand:

- ▶ Wenn Sie eine zusätzliche Partition in das Dateisystem einbinden und dabei ein bisher nicht genutztes Dateisystemformat zum Einsatz kommt, wird automatisch das Modul für dieses Dateisystem geladen.
- ▶ Wenn sich die Partition auf einem USB-Laufwerk befindet, werden auch die USB-Module aktiviert, sofern diese nicht ohnedies schon geladen sind.
- ▶ Während der Initialisierung von Netzwerkfunktionen wird automatisch der erforderliche Treiber für Ihre Netzwerkkarte geladen etc.

Für das automatische Laden von Kernelmodulen ist die in den Kernel integrierte Komponente `kmod` verantwortlich. Dafür, dass all das funktioniert, sorgen unterschiedliche Konfigurationsmechanismen:

- ▶ **Für den Rechnerstart erforderliche Module:** Manche Kernelmodule werden sofort beim Start des Rechners benötigt – etwa Module zum Zugriff auf das Dateisystem. Sofern diese Module nicht integrale Bestandteile des Kernels sind, müssen sie in einer `Initrd`-Datei durch GRUB beim Rechnerstart an den Kernel übergeben werden (siehe Abschnitt 24.1, »GRUB-Grundlagen«).
- ▶ **Module für Grundfunktionen:** Die Module für die Basisverwaltung von Hardware-Komponenten (z. B. für das USB-System) werden von verschiedenen Scripts des Init-Prozesses direkt durch `modprobe`-Anweisungen geladen.
- ▶ **Module für Schnittstellen:** Eine Reihe weiterer Module wird dann geladen, wenn eine Schnittstelle zum ersten Mal benutzt wird. Hier tritt allerdings das Problem auf, dass es für manche Schnittstellen je nach der eingesetzten Hardware unterschiedliche Module gibt.

Wenn Sie also die Schnittstelle `eth0` für die erste Netzwerkkarte im Rechner ansprechen, muss das zu dieser Karte passende Modul geladen werden.

Da der Kernel nicht hellsehen kann, benötigt er eine Information darüber, welches Modul das richtige ist. Diese Information befindet sich in `/etc/modprobe.conf` sowie in den Dateien der Verzeichnisse `/etc/modprobe.d` und `/etc/modules-load.d`. Dort befinden sich installations- oder distributionsspezifische Optionen sowie Anweisungen, welche Module *nicht* automatisch zu laden sind (blacklist-Datei). Bei systemd-Distributionen werden diese Informationen durch die Service-Datei `systemd-modules-load.service` ausgewertet. Auf die Syntax der `modprobe`-Dateien gehe ich gleich im Detail ein.

Auch die automatische Device-Verwaltung durch das `udev`-System lädt bei Bedarf die notwendigen Module. Die entsprechenden Regeln finden Sie in `/etc/udev/rules.d`.

- ▶ **Module für USB-Geräte etc.:** Derartige Hardware-Komponenten nehmen eine Sonderrolle ein. Die Datei `/lib/modules/n.n/modules.alias` enthält eine Referenz mit Identifikationscodes der Komponenten und entscheidet darüber, welches Modul geladen wird.
- ▶ **Modulabhängigkeiten:** Eine Menge Module sind voneinander abhängig. Beispielsweise funktioniert das Modul `nfs` für das NFS-Dateisystem nur, wenn auch die Module `lockd`, `nfs_acl` und `sunrpc` geladen sind. Derartige Modulabhängigkeiten sind zentral in der Datei `/lib/modules/n.n/modules.dep` verzeichnet.

Module beim Rechnerstart laden

Manchmal wollen Sie unabhängig von den hier zusammengefassten Konfigurationswegen erreichen, dass beim Rechnerstart ein bestimmtes Kernelmodul geladen wird – und das, ohne sich auf irgendwelche Automatismen zu verlassen. Die optimale Vorgehensweise hängt von Ihrer Distribution ab.

Besonders einfach ist es bei Debian und Ubuntu: Dort kümmert sich das durch `systemd` einmalig ausgeführte Kommando `kmod` darum, alle in `/etc/modules` zeilenweise aufgelisteten Module zu laden. Sie müssen also lediglich das gewünschte Modul in einer neuen Zeile in `/etc/modules` angeben.

Bei vielen anderen Distributionen sowie bei aktuellen Debian- und Ubuntu-Versionen existiert zusätzlich das Verzeichnis `/etc/modules-load.d`. Es ist anfänglich zumeist leer. Wenn Sie Module automatisch laden möchten, erzeugen Sie in dem Verzeichnis eine Datei mit der Kennung `*.conf` und speichern dort die Modulnamen zeilenweise.

Viele Kernelmodule werden ganz früh im Bootprozess geladen. Damit Ihre Änderungen in `/etc/modules` oder in der `modprobe`-Konfiguration wirksam werden, müssen Sie die `initrd`-Datei aktualisieren (siehe auch Abschnitt 24.2, »Initrd-Dateien«)!

```
sudo update-initramfs -u      # Debian, Ubuntu <= 25.04
sudo dracut --force           # Fedora, RHEL, Ubuntu >= 25.10
sudo mkinitcpio -P            # Arch Linux, CachyOS
```

modprobe-Syntax

Die folgenden Absätze beschreiben die wichtigsten Schlüsselwörter für die Dateien `modprobe.conf`, `/etc/modprobe.d/*` sowie `/lib/modules/n.n/modules.alias`. Weitere Details liefert man `modprobe.conf`.

- `alias`-Anweisungen geben an, welche Kernelmodule für welche Devices eingesetzt werden. Dazu ein Beispiel: Für das Device `/dev/eth0` soll das Modul `8139too` verwendet werden:

```
alias eth0 8139too
```

Der Zugriff auf viele Hardware-Komponenten erfolgt durch block- und zeichenorientierte Device-Dateien im `/dev`-Verzeichnis. Aus der Sicht des Kernels werden diese Device-Dateien nicht durch ihren Namen, sondern durch die Major- und Minor-Device-Nummer charakterisiert (siehe auch Abschnitt 12.9, »Device-Dateien«). Zahlreiche `alias`-Anweisungen stellen den Zusammenhang zwischen Device-Nummern und Modulen her.

Analog sieht auch die Definition von Netzwerkprotokollen aus: Um ein bestimmtes Protokoll zu nutzen, sucht der Kernel nach einer Protokollfamilie mit dem Namen `net-pf-N`, wobei `N` die ID-Nummer des Protokolls ist. Das folgende Beispiel bewirkt, dass für die Protokollfamilie 5 das `AppleTalk`-Modul geladen wird:

```
alias net-pf-5 appletalk
```

Wenn Sie dieses veraltete Protokoll nicht brauchen und womöglich das entsprechende Modul gar nicht installiert ist, dann erspart Ihnen die folgende Anweisung lästige Fehlermeldungen:

```
alias net-pf-5 off
```

- `options`-Anweisungen geben an, mit welchen Optionen ein bestimmtes Modul geladen werden soll. Die folgende Anweisung bewirkt, dass das Modul `ne` (für NE-2000-kompatible Ethernet-Karten) mit der Option `io=0x300` geladen wird:

```
options ne io=0x300
```

- `include`-Anweisungen laden weitere Konfigurationsdateien.
- Mit `install`-Anweisungen geben Sie Kommandos an, die ausgeführt werden, anstatt das betreffende Modul einfach zu laden. Auch hierzu sehen Sie ein Beispiel, das aus Platzgründen auf zwei Zeilen verteilt wurde. Wenn das `ALSA`-Modul `snd` benötigt wird, sollen die folgenden Kommandos ausgeführt werden:

```
install snd modprobe --ignore-install snd $CMDLINE_OPTS && \
{ modprobe -Qb snd-ioc132 ; : ; }
```

- Mit `remove` geben Sie Kommandos an, die beim Entfernen eines Moduls ausgeführt werden sollen.
- `blacklist` bewirkt, dass modulinterne Alias-Definitionen nicht berücksichtigt werden. `blacklist`-Anweisungen befinden sich üblicherweise in der Datei `/etc/modprobe.d/`

blacklist. Sie enthält Module, die beispielsweise wegen Kompatibilitätsproblemen oder aufgrund von besseren Alternativen *nicht* geladen werden sollen. Beispielsweise verhindert die folgende Zeile, dass das Modul `usbmouse` geladen wird:

```
blacklist usbmouse
```

26.2 Device Trees

Der Begriff »Device Tree« bezeichnet die hierarchische Darstellung von Hardware-Komponenten. Der Device Tree wird während des Boot-Vorgangs vom Kernel geladen und teilt diesem mit, welche Hardware-Komponenten zur Verfügung stehen und über welche Anschlüsse diese Komponenten genutzt werden.

Device Trees wurden von den Linux-Kernelentwicklern ersonnen, um der Vielfalt von Chips und Geräten (sprich: Smartphones) auf Basis von ARM-CPUs Herr zu werden. Dank Device Trees ist es möglich, dass ein für ARM-Geräte kompilierter Kernel auf unterschiedlichen Geräten mit unterschiedlichen Zusatzkomponenten laufen kann. Bei PCs ist das selbstverständlich; in der ARM-Welt war dies aufgrund der Vielfalt von CPU- und Hardware-Varianten aber bisher unmöglich.

Sofern Sie nicht gerade ein Kernelentwickler für Smartphones sind, kommen Sie am ehesten bei der Arbeit mit Minicomputern wie dem Raspberry Pi mit Device Trees in Kontakt. Device Trees werden beispielsweise in Raspberry Pi OS standardmäßig eingesetzt. Beim Raspberry Pi beschreibt der Device Tree den Chip BCM27xx/28xx mit all seinen vielen Steuerungsmöglichkeiten, Bus-Systemen, GPIOs etc. Die Beschreibung erfolgt in einem Textformat, das dann aus Platz- und Effizienzgründen in ein binäres Format umgewandelt wird. Die Details dieses Formats sind aus Anwendersicht nicht relevant. Wenn Sie sich dennoch dafür interessieren, finden Sie auf den folgenden Seiten eine umfassende technische Referenz:

<https://devicetree.org>

https://elinux.org/Device_Tree

<https://linux-magazin.de/Ausgaben/2013/06/Kern-Technik>

Device Trees sind auch für den Betrieb von Notebooks mit ARM-CPU relevant (im Unterschied zu Notebooks mit Intel- und AMD-CPU, deren Hardware-Eigenschaften per ACPI Discovery (*Advanced Configuration and Power Interface*) ermittelt werden). Die Integration der korrekten Device-Tree-Daten in einen Boot-Prozess, der allgemeingültig für verschiedene Notebook-Modelle funktioniert, hat sich als großes Problem für Linux-Distributionen herausgestellt. Ein Lösungsansatz ist das Project *Stubble* von Canonical/Ubuntu, das die gerätespezifischen Device Trees in das Kernel-Image integriert. Stubble ist kompatibel zu `systemd-stub` und `ukify`. Mehr Details können Sie hier nachlesen:

<https://github.com/ubuntu/stubble>

<https://lwn.net/Articles/1034579>

Device-Tree-Konfiguration beim Raspberry Pi

Die Device-Tree-Konfiguration des Raspberry Pi ist über mehrere Orte verteilt. Das Verzeichnis `/boot/firmware` enthält Device-Tree-Beschreibungen für alle momentan gängigen Raspberry-Pi-Modelle. Die Dateikennung `*.dtb` steht dabei für *Device Tree Blob*, wobei ein *Blob* einfach ein binäres Objekt ist. Die Nummern 2708 bis 2711 sind Broadcom-interne Nummern zur Beschreibung der Chip-Familie. Die auf dem Raspberry Pi eingesetzten Modelle BCM2835 bis BCM2837 sind konkrete Implementierungen (»Familienmitglieder«, wenn Sie so wollen).

- ▶ `bcm2708-rpi-0-w.dtb`: Raspberry Pi Zero W/WH
- ▶ `bcm2710-rpi-3-b.dtb/bcm2710-rpi-3-b-plus.dtb`: Raspberry Pi Modell 3B und 3B+
- ▶ `bcm2711-rpi-4-b.dtb`: Raspberry Pi Modell 4B
- ▶ `bcm2711-rpi-400.dtb`: Raspberry Pi 400
- ▶ `bcm2712-rpi-5-b.dtb`: Raspberry Pi 5
- ▶ `bcm2711-rpi-500.dtb`: Raspberry Pi 500

Während des Boot-Prozesses übergibt das in der Boot-Datei `start.elf` enthaltene Programm den für das jeweilige Raspberry-Pi-Modell geeigneten Device Tree an den Kernel.

Das Verzeichnis `/boot/firmware/overlays` enthält fast 400 Device-Tree-Blob-Overlays (DTBOs), von denen ich hier nur einige wenige exemplarisch nenne:

- ▶ `hifiberry-dacplus-overlay.dtbo`: für die HiFiBerry-Erweiterung DAC+
- ▶ `lirc-rpi-overlay.dtbo`: für Infrarot-Fernbedienungen
- ▶ `i2c-rtc-overlay.dtbo`: für I²C-Komponenten mit Real Time Clock
- ▶ `w1-gpio-overlay.dtbo`: für 1-Wire-Temperatursensoren

Overlays sind also Ergänzungen zum Haupt-Device-Tree `bcmxxx`. Diese DTBs werden *nicht* automatisch geladen, sondern nur, wenn die Konfigurationsdatei `config.txt` entsprechende Hinweise enthält. Sie ergänzen den Device Tree des BCM28xx bzw. BCM2711.

Die Raspberry-Pi-spezifische Datei `/boot/firmware/config.txt` kennt drei Schlüsselwörter zum Umgang mit Device Trees:

- ▶ `dtparam=i2c_arm=on/off,i2s=on/off,spi=on/off` aktiviert oder deaktiviert die Bussysteme I²C, I²S und SPI. Die Beschreibung dieser Bussysteme ist im Haupt-Device-Tree bereits enthalten. Standardmäßig sind alle drei Bussysteme inaktiv (entspricht `dtparam=i2c_arm=off,i2s=off,spi=off`). Wenn ein Bussystem oder mehrere genutzt werden sollen, müssen sie explizit aktiviert werden.
 - `dtoverlay=name,key1=val1,key2=val2...` bewirkt, dass die genannte Overlay-Datei geladen wird, wobei die übergebenen Parameter berücksichtigt werden.
 - `device_tree=`, also ohne die Zuweisung eines konkreten Werts, deaktiviert das gesamte Device-Tree-System.

Die Schlüsselwörter `dtparam` und `dtoverlay` können mehrfach verwendet werden.

Anhand der Device-Tree-Konfiguration entscheidet der Linux-Kernel, welche Module er lädt. Daher ersetzt die Device-Tree-Konfiguration die bei älteren Raspbian-Installationen erforderlichen Einstellungen in `/etc/modules` bzw. in `/etc/modprobe.d/*`.

In einfachen Fällen können Sie `config.txt` mit dem Programm `raspi-config` im Untermenü **ADVANCED OPTIONS** korrekt einrichten. Das gilt insbesondere, wenn Sie die Bussysteme I²C und SPI verwenden möchten.

In allen anderen Fällen müssen Sie die entsprechenden Zeilen hingegen selbst in `config.txt` eintragen. Das folgende Listing illustriert die Syntax anhand einiger Beispiele. Beachten Sie, dass mehrere Optionen nur durch Kommata, aber nicht durch Leerzeichen voneinander getrennt werden dürfen (siehe z. B. `dtoverlay=...`).

```
# Datei /boot/firmware/config.txt
# Beispiele zur Steuerung des Device-Tree-Systems (weitere Details
# siehe /boot/overlays/README in einer Raspberry-Pi-OS-Installation)

# Audio-System aktivieren
dtparam=audio=on

# SPI-Bus aktivieren
dtparam=spi=on

# I2C-Bus aktivieren
dtparam=i2c_arm=on

# HiFiBerry DAC+ verwenden
dtoverlay=hifiberry-dacplus

# 1-Wire-Temperatursensor mit Standardeinstellungen verwenden
dtoverlay=w1-gpio-pullup

# 1-Wire-Temperatursensor verwenden, der mit
# GPIO X verbunden ist (per Default GPIO 4),
# und dabei den internen Pull-up-Widerstand aktivieren
dtoverlay=w1-gpio-pullup,gpiopin=X,pullup=y

# Echtzeituhr-Modell ds1307 verwenden
dtoverlay=rtc-i2c,ds1307

# IR-Empfänger verwenden
dtoverlay=lirc-rpi
```

26.3 Kernelmodule selbst kompilieren

Wenn Sie Linux in Kombination mit VirtualBox einsetzen, die proprietären Grafiktreiber von NVIDIA nutzen möchten oder ein anderes hardware-spezifisches Kernelmodul brauchen, das im Kernel Ihrer Distribution fehlt, dann müssen Sie das Modul passend zum laufenden Kernel kompilieren.

Zum Kompilieren eines Moduls sind neben dem C-Compiler `gcc` und `make` auch weitere grundlegende Entwicklungswerkzeuge erforderlich. Die meisten Distributionen erleichtern die Sache durch fertige Paketselektionen oder Meta-Pakete, die auf alle relevanten Pakete verweisen (siehe Tabelle 26.1).

Distribution	Kommando
Arch Linux	<code>pacman -S base devel</code>
Debian, Ubuntu	<code>apt install build-essential</code>
Fedora, RHEL	<code>dnf group install development-tools</code>
SUSE	<code>zypper install -t pattern devel_basis</code>

Tabelle 26.1 Kommandos zur Installation grundlegender Entwicklungswerkzeuge

Außerdem brauchen Sie die Include-Dateien (Header-Dateien) zum aktuellen Kernel (siehe Tabelle 26.2). Diese Dateien sind Teil des Kernelcodes. Bei einigen Distributionen befinden sich die Include-Dateien und der Rest des Codes in zwei getrennten Paketen. Das hat den Vorteil, dass Sie nicht gleich den riesigen Kernelcode installieren müssen, wenn Sie nur die vergleichsweise kleinen Include-Dateien brauchen.

Distribution	Kommando
Arch Linux	<code>pacman -S linux-headers</code>
Debian, Ubuntu	<code>apt install linux-headers-\$(uname -r)</code>
Fedora, RHEL	<code>dnf install kernel-devel</code>
SUSE	<code>zypper install kernel-devel</code>

Tabelle 26.2 Kommando zur Installation der Kernel-Header-Dateien

Natürlich landen die Header-Dateien je nach Distribution in unterschiedlichen Verzeichnissen (siehe Tabelle 26.3). `n.n` ist dabei ein Platzhalter für die installierte Kernelversion. Diese Information ermitteln Sie mit dem Kommando `uname -r`.

Distribution	Pfad
Arch Linux	/usr/lib/modules/n.n/build
Debian, Ubuntu	/usr/src/linux-headers-n.n
Fedora, RHEL	/usr/src/kernels/n.n
SUSE	/usr/src/linux-n.n-obj/arch/default

Tabelle 26.3 Installationsort der Kernel-Header-Dateien

Wenn Sie den Kernel selbst kompilieren (siehe Abschnitt 26.4), landen die zum Kernel passenden Include-Dateien automatisch im Verzeichnis `/lib/modules/n.n/build/include`.

Modul kompilieren

Die meisten Programme, die eigene Kernelmodule benötigen, enthalten ein Installations-Skript, das sich um das Kompilieren und Einrichten des Moduls kümmert. Das gilt beispielsweise für VirtualBox oder die Grafiktreiber von NVIDIA. Bei manchen Distributionen ist der Prozess sogar dahin gehend automatisiert, dass nach jedem Kernel-Update automatisch das Modul neu kompiliert wird (siehe *DKMS* weiter unten).

Wenn Sie dagegen den Quellcode für eine noch nicht offiziell unterstützte Hardware-Komponente heruntergeladen haben, müssen Sie sich um den Kompilierprozess selbst kümmern. Dazu führen Sie in der Regel die folgenden Kommandos aus. Nur das `make`-Kommando erfordert root-Rechte.

```
cd quellcodeverzeichnis
make clean
make
sudo make install
```

Modul-Updates mit DKMS automatisieren

DKMS steht für *Dynamic Kernel Module Support* und hilft dabei, nach einem Kernel-Update selbst kompilierte Kernelmodule automatisch zu aktualisieren. DKMS besteht aus einigen Shell-Skripts und wurde von Dell entwickelt. Die entsprechenden `dkms`-Pakete stehen gegenwärtig für die Distributionen Debian, Fedora und Ubuntu zur Verfügung.

Um DKMS zu nutzen, muss der Quellcode des Moduls in einem Verzeichnis der Form `/usr/src/NAME-VERSION` installiert werden. Das Verzeichnis muss die Datei `dkms.conf` enthalten, die DKMS erklärt, wie es mit dem Code umgehen soll. Die folgenden Zeilen stammen vom NVIDIA-Treiber für Ubuntu, wobei ich die Formatierung des Listings geändert habe, um die Lesbarkeit zu verbessern. Tatsächlich sind vor und nach dem Zeichen `=` keine Leerzeichen erlaubt!

```
# Datei /usr/src/nvidia-580.76.05/dkms.conf
PACKAGE_NAME          = "nvidia#"
PACKAGE_VERSION        = "580.76.05"
CLEAN                  = "make clean"
BUILT_MODULE_NAME[0]   = "nvidia"
DEST_MODULE_LOCATION[0] = "/kernel/drivers/char/drm"
PROCS_NUM              = `nproc`
[ $PROCS_NUM -gt 16 ] && PROCS_NUM=16
MAKE[0]                = "unset ARCH; [ ! -h /usr/bin/cc ] && ..."
BUILT_MODULE_NAME[1]   = "nvidia-modeset"
DEST_MODULE_LOCATION[1] = "/kernel/drivers/char/drm"
BUILT_MODULE_NAME[2]   = "nvidia-drm"
DEST_MODULE_LOCATION[2] = "/kernel/drivers/char/drm"
...
```

Sind diese Voraussetzungen erfüllt, übergeben Sie das Kernelmodul mit `dkms add` der Kontrolle von DKMS, kompilieren es mit `dkms build` für den aktuellen Kernel und installieren es mit `dkms install`. Die folgenden Beispiele beziehen sich wieder auf den NVIDIA-Kerneltreiber. Die Kommandos werden normalerweise nach dem Update des Kernels oder eines Treibers automatisch ausgeführt (Kommando `dkms autoinstall`). Nur wenn dieser Automatismus nicht funktioniert, müssen Sie manuell nachhelfen und sich bei eventuellen Fehlern auf die Suche nach deren Ursachen machen. (Mitunter sind die Versionen des Kernels und des Treibers nicht kompatibel zueinander.)

```
sudo dkms add      -m nvidia -v 580.76.05
sudo dkms build    -m nvidia -v 580.76.05
sudo dkms install  -m nvidia -v 580.76.05
```

Die obigen Kommandos gelten für den gerade laufenden Kernel. Um Kernelmodule für eine andere Kernelversion zu kompilieren und zu installieren, fügen Sie den obigen Kommandos die Option `-k n.n` hinzu, wobei `n.n` die Versionsnummer eines auf Ihrem Rechner installierten Kernels ist.

`dkms status` bzw. ein Blick in das Verzeichnis `/var/lib/dkms` verrät, welche Kernelmodule sich momentan unter der Kontrolle von DKMS befinden.

Bei Debian und Ubuntu gibt es eine ganze Reihe von `xxx-name-dkms`-Paketen, mit denen Kernelmodule für Hardware-Treiber kompiliert werden können:

```
sudo apt-cache --names-only search '.*-dkms' | sort
```

Bei Debian befinden sich die Pakete zum Teil in den Paketquellen *contrib* und *non-free*, die Sie eventuell vorher aktivieren müssen. Die Installation eines Kernelmoduls sieht dann wie folgt aus, wobei Sie einfach `nvidia` durch den Namen des gewünschten Treibers und `amd64` durch Ihre CPU-Architektur ersetzen:

```
sudo apt update
sudo apt install linux-headers-n.n-amd64 nvidia-nnn-dkms
```

Im Rahmen der Installation werden alle abhängigen Pakete installiert sowie das Kernelmodul kompiliert und als DKMS-Modul eingerichtet. Bei zukünftigen Kernel-Updates wird somit automatisch eine neue Version des Moduls erzeugt.

DKMS oder module-assistant?

Vor allem in Debian kamen zum Kompilieren von Kernelmodulen in der Vergangenheit häufig die Werkzeuge aus dem Paket `module-assistant` zum Einsatz. Das gleichnamige Kommando bzw. dessen Kurzform `m-a` existiert weiterhin, muss aber extra installiert werden; nach Möglichkeit sollten Sie DKMS-Pakete vorziehen!

akms

Die RPMFusion-Paketquelle für Fedora verwendet anstelle von DKMS einen eigenen Mechanismus zum Kompilieren von Kernelmodulen: Das Kommando `akms` wird nach Kernel-Updates aufgerufen. Es kompiliert zu allen RPM-Source-Paketen, die sich in `/usr/src/akmods` befinden, die entsprechenden Kernelmodule und installiert diese.

<https://rpmfusion.org/Packaging/KernelModules/Akmods>

26.4 Kernel selbst konfigurieren und kompilieren

Der durchschnittliche Linux-Anwender muss seinen Kernel nicht selbst kompilieren. Bei allen aktuellen Distributionen werden ein brauchbarer Standardkernel und eine umfangreiche Sammlung von Modulen mitgeliefert. Dennoch kann es Gründe geben, den Kernel neu zu kompilieren:

- ▶ Sie wollen Ihr System besser kennenlernen. Das Motto dieses Buchs ist es ja, Ihnen auch einen Blick hinter die Linux-Kulissen zu ermöglichen.
- ▶ Sie brauchen besondere Funktionen, die weder in den mitgelieferten Kernel integriert sind noch als Modul vorliegen.
- ▶ Sie möchten eine aktuellere Version des Kernels verwenden als die, die mit Ihrer Distribution mitgeliefert wurde.
- ▶ Sie möchten selbst an der Kernelentwicklung teilnehmen und daher mit dem neuesten Entwicklerkernel experimentieren.
- ▶ Sie wollen in Ihrem Bekanntenkreis mit Insider-Wissen auftrumpfen: »Ich habe den neuesten Linux-Kernel selbst kompiliert!«

Hürden

Es gibt allerdings gewichtige Gründe, die gegen das Kompilieren eines eigenen Kernels sprechen:

- Viele Distributionen verwenden nicht den Originalkernel, wie er von Linus Torvalds freigegeben wird, sondern eine gepatchte Version mit diversen Zusatzfunktionen, wobei natürlich jede Distribution andere Patches verwendet.

An sich ist das eine feine Sache für den Anwender: Er bekommt auf diese Weise Zusatzfunktionen, von denen der Distributor glaubt, dass sie schon ausreichend stabil funktionieren. Wenn Sie sich nun aber selbst den Quellcode des Originalkernels herunterladen, fehlen diese Patches. Einzelne Funktionen Ihrer Distribution, die bisher einwandfrei gearbeitet haben, machen plötzlich Probleme oder funktionieren gar nicht mehr.

- Das Kompilieren eines eigenen Kernels ist nicht schwierig. Schwierig ist aber die vorherige Konfiguration des Kompilationsprozesses. Dabei stehen Tausende von Optionen zur Auswahl. Sie können mit diesen Optionen beeinflussen, welche Funktionen direkt in den Kernel integriert werden, welche als Module und welche gar nicht zur Verfügung stehen sollen.

Wenn Sie sich – mangels Detailwissen – für die falschen Optionen entscheiden, ist das Ergebnis wie oben: Einzelne Funktionen verweigern den Dienst, und es ist relativ schwierig, die Ursache herauszufinden. Gerade für Linux-Einsteiger ist es praktisch unmöglich, die passenden Einstellungen für alle Optionen richtig zu erraten.

Aus diesen Gründen verweigern die meisten Distributoren jeden Support, wenn Sie nicht den mit der Distribution mitgelieferten Kernel verwenden. Lassen Sie sich von diesen Warnungen aber nicht abschrecken, es einmal selbst zu versuchen. Wenn Sie nach der in diesem Abschnitt präsentierten Anleitung vorgehen, können Sie Ihren Rechner anschließend sowohl mit dem alten als auch mit dem neuen Kernel hochfahren – es kann also nichts passieren!

Entwicklungswerkzeuge

Zur Kompilierung des Kernels sind dieselben Entwicklungswerkzeuge wie zum Kompilieren eines einzelnen Moduls erforderlich (siehe Abschnitt 26.3).

Je nach Distribution sind darüber hinaus weitere Pakete notwendig. Unter Debian und Ubuntu müssen Sie beispielsweise in `/etc/apt/sources.list` die `deb-src`-Paketquellen aktivieren und dann die folgenden Kommandos ausführen:

```
sudo apt update
sudo apt install flex bison
sudo apt build-dep linux
```

Unter Fedora sind häufig die folgenden Pakete nicht automatisch installiert:

```
sudo dnf install dwarves ncurses-devel zstd
```

Kernelversionen

Die Weiterentwicklung des Linux-Kernels erfolgt seit Jahren im gleichen Rhythmus: Sobald Linus Torvalds befindet, dass die gerade in Entwicklung befindliche Kernelversion ausreichend stabil ist, wird diese offiziell freigegeben. Gleichzeitig beginnen die Arbeiten an der nächsten Version: Entwickler können in den nächsten zwei Wochen Patches für neue Funktionen bei Linus Torvalds einreichen. Danach dauert es typischerweise fünf bis sieben Wochen, bis alle Probleme und Fehler im neuen Code behoben sind und die nächste Kernelversion fertig wird.

Die Versionsnummer des Kernels besteht aus drei Teilen: *Major Release Number*, *Minor Release Number* und *Bugfix Release Number*. Als ich dieses Kapitel Ende August 2025 überarbeitete, war die Kernelversion 6.16 aktuell. Diese hatte Linus Torvalds am 27. Juli freigegeben. Bis Ende August gab es nur ein kleineres Update zur Behebung von Bugs und Sicherheitsproblemen. Die vollständige Versionsnummer lautete zu diesem Zeitpunkt 6.16.2.

Long Term Releases

Grundsätzlich wird die Wartung alter Kernelversionen mit der Fertigstellung der jeweils neuesten Version beendet. Ausgenommen von dieser Regel sind ausgewählte Kernelversionen, die durch die Kernelentwicklergemeinde über mehrere Jahre gepflegt werden. Im August 2025 genossen die folgenden Kernelversionen Langzeitunterstützung: 5.4, 5.10, 5.15, 6.1, 6.6 und 6.12.

Für Linux-Distributoren gibt es drei Varianten, mit Kernel-Updates umzugehen:

- ▶ Vordergründig am einfachsten wäre es, stets die gerade aktuellste Kernelversion als Update anzubieten. Das kann aber zu Stabilitätsproblemen führen (vor allem dann, wenn weitere Komponenten, z. B. das Grafiksystem, nicht ebenfalls aktualisiert werden), weswegen die meisten Distributoren vor diesem Weg zurückschrecken. Zu den wenigen Ausnahmen zählen Fedora sowie Rolling-Release-Distributionen wie Arch Linux oder openSUSE Tumbleweed.
- ▶ Die nächstbeste Variante besteht darin, für die Distribution die gerade aktuellste Long-Term-Linux-Version zu verwenden. Das hat für den Distributor den großen Vorteil, dass er sich um die Pflege des Kernelcodes nicht selbst kümmern muss.
- ▶ Am aufwendigsten ist es, wenn ein Distributor nicht auf einen Langzeit-Kernel zurückgreift, sondern den ursprünglich mit der Distribution ausgelieferten Kernel selbst pflegt. Dazu müssen Sicherheits-Updates und fallweise auch Zusatzfunktionen aus aktuellem Kernelcode »rück-portiert« werden. Besonders bemerkenswert ist diesbezüglich Red Hat, das für seine Enterprise-Distributionen mit immensem Arbeitsaufwand ein ganzes Jahrzehnt lang alte Kernelversionen mit Sicherheits-Updates versorgt.

Statistik und Links

Der Kernel besteht zurzeit aus über 40 Millionen Zeilen Code. Der Großteil davon ist in C geschrieben, ein kleiner Teil in Assembler sowie in der Sprache Rust. Wenn Sie wissen möchten, wer bzw. welche Firmen zur Kernelentwicklung beitragen, verfolgen Sie einfach die Linux-News-Site <https://lwn.net>. Dort finden Sie zu jedem Kernel-Release eine statistische Aufarbeitung, wer die meisten Änderungen durchgeführt hat:

<https://lwn.net/Articles/1031161> (für Version 6.16)

Tipps zur Kompilierung des Kernels finden Sie auf der folgenden Seite:

<https://kernelnewbies.org/FAQ>

Wenn Sie sich für technische Interna interessieren, sind die Dokumentationsdateien des Kernelcodes sehr aufschlussreich. Gerade neue Funktionen des Kernels werden zuerst an dieser Stelle beschrieben, noch bevor die entsprechenden man-Seiten aktualisiert werden:

<https://www.kernel.org/doc/Documentation>

Kernelcode installieren

Der Quellcode für den Kernel befindet sich üblicherweise im Verzeichnis `/usr/src/linux`; nur bei Red Hat und Fedora gibt es abweichende Gepflogenheiten, die weiter unten behandelt werden. Falls dieses Verzeichnis leer ist, haben Sie den Kernelcode nicht installiert. Sie können nun wahlweise den Kernelquellcode Ihrer Distribution installieren oder den gerade aktuellen offiziellen Kernelcode herunterladen. Weniger Probleme bereitet zumeist die erste Variante, insbesondere für Einsteiger.

Ist genug Platz auf der SSD?

Der Platzbedarf für den Kernelcode ist beachtlich! Ich habe für meine Tests das offizielle Git-Repository geklont, das an sich schon ca. 8 GiB Platz beansprucht. Beim Kompilieren kommen unzählige Binärdateien hinzu, der Platzbedarf steigt auf ca. 35 GiB. Zuletzt können Sie mit `make clean` zahllose Objektdateien wieder löschen, aber zwischenzeitlich brauchen Sie genug freien Speicherplatz.

Kernelcode der Distribution installieren

Bei den meisten Distributionen gibt es ein eigenes Paket, das den Kernelquellcode enthält (siehe Tabelle 26.4). Dabei ist `n.n` ein Platzhalter für die installierte Kernelversion.

Bei Debian und Ubuntu wird der Kernelcode als tar-Archiv in das Verzeichnis `/usr/src` installiert. Sie müssen das Archiv selbst mit `tar xjf linux-n.n.tar.bz2` auspacken. Die Kennung `.bz2` deutet darauf hin, dass der Quellcode mit dem besonders effizienten bzip2-Verfahren komprimiert wurde.

Bei RHEL finden Sie nach `dnf download --source kernel` das Paket `kernel-n.n.src.rpm` im aktuellen Verzeichnis. Mit `rpm -iv` packen Sie es im Verzeichnis `rpmbuild/SOURCE` aus.

Distribution	Kommando
Debian, Ubuntu	<code>apt install linux-source</code>
Fedora	<code>fedpkg clone -a kernel</code> (Details siehe unten)
RHEL	<code>dnf download --source kernel</code> (Quellcodepaket!)
SUSE	<code>zypper install kernel-source</code>

Tabelle 26.4 Installation des distributionsspezifischen Kernelquellcodes

Den aktuellen Kernel-Code für Arch Linux laden Sie am besten mit `git` aus dem entsprechenden Arch-Linux-Repo herunter:

```
git clone \
  https://gitlab.archlinux.org/archlinux/packaging/packages/linux.git
```

Fedora ist unter Kernelentwicklern eine besonders beliebte Distribution. Bevor Sie loslegen, müssen Sie diverse Entwicklerpakete installieren und den aktuellen User-Account für `pesign` freischalten. (`pesign` ist ein Kommando zum Signieren von UEFI-Programmen.)

```
sudo dnf install fedpkg fedora-packager rpmdevtools \
  ncurses-devel pesign openssl
```

```
sudo pesign
```

```
pesign: Nothing to do.
```

Die Fedora-Entwickler empfehlen, den Quellcode des Kernels sowie aller Fedora-Patches mit `fedpkg clone -a kernel` aus einem Git-Repository herunterzuladen. Details können Sie hier nachlesen:

https://fedoraproject.org/wiki/Building_a_custom_kernel

Offiziellen Kernelcode installieren

Der mit der Distribution mitgelieferte Kernel ist oft schon veraltet. Den aktuellen Kernelcode in Form von komprimierten tar-Archiven finden Sie hier:

<https://www.kernel.org>

Anstatt eine bestimmte Kernelversion herunterzuladen und mit ihr zu arbeiten, kann es sinnvoller sein, einen Klon des offiziellen Git-Repositorys für stabile Kernelversionen zu erzeugen. `git tag` ermittelt in Kombination mit `sort -V` (numerische Sortierordnung) die aktuellsten

zehn im Code enthaltenen Versionen. `git checkout` aktiviert die Version, die Sie anschließend tatsächlich nutzen (kompilieren) möchten – hier den Release Candidate 3 der damals noch in Entwicklung befindlichen Version 6.17:

```
git clone \
    git://git.kernel.org/pub/scm/linux/kernel/git/stable/linux-stable.git

cd linux-stable

git tag -l | sort -V | tail -n 10

v6.16-rc4
v6.16-rc5
v6.16-rc6
v6.16-rc7
v6.16.1
v6.16.2
v6.16.3
v6.17-rc1
v6.17-rc2
v6.17-rc3

git checkout v6.17-rc3
```

Anfänglich wird durch `git clone` zwar deutlich mehr Code heruntergeladen (Download-Umfang knapp 6 GiB, Platzbedarf auf der SSD ca. 8 GiB); dafür verlaufen spätere Updates bzw. Versionswechsel aber einfacher und schneller – elementare Grundkenntnisse des `git`-Kommandos einmal vorausgesetzt.

Beachten Sie, dass es sich hier um den originalen Kernelcode handelt, wie ihn Linus Torvalds freigegeben hat – also ohne distributionsspezifische Patches. Dieser Kernel wird oft *Vanilla Kernel* genannt.

Varianten und Entwicklerzweige

Es gibt im Internet unzählige weitere Git-Repositorys mit diversen Varianten und Entwicklungszweigen des Kernelcodes. Werfen Sie insbesondere einen Blick in das Blog des Kernelentwicklers Greg Kroah-Hartman (vierter Link)!

<https://git.kernel.org>

<https://github.com/torvalds/linux>

<https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git>

<http://www.kroah.com/log/blog/2019/06/15/linux-stable-tree-mirror-at-github>

Kernelkonfiguration

Der Kernel besteht aus Tausenden von Einzelfunktionen bzw. Komponenten. Bei nahezu allen Funktionen können Sie vor dem Kompilieren angeben, ob sie direkt in den Kernel integriert werden, als Modul kompiliert werden oder gar nicht verfügbar sein sollen. Dieser Vorgang heißt den »Kernel konfigurieren«.

Die Kernelkonfiguration wird durch die Datei `.config` im Verzeichnis mit dem Kernelquellcode bestimmt. Dabei handelt es sich um eine aktuell ca. 13.000 Zeilen lange Textdatei, die angibt, ob eine Funktion direkt in den Kernel integriert (`name=y`) oder als Modul kompiliert werden soll (`name=m`). Nicht benötigte Funktionen erscheinen in der Konfigurationsdatei nicht bzw. nur in Kommentarzeilen. Die Datei kann auch zusätzliche Einstellungen enthalten (`name=wert`). Die folgenden Zeilen zeigen einige Zeilen einer `.config`-Datei:

```
CONFIG_64BIT=y
CONFIG_X86_64=y
CONFIG_X86=y
CONFIG_INSTRUCTION_DECODER=y
CONFIG_OUTPUT_FORMAT="elf64-x86-64"
...
```

Wenn Sie bei der manuellen Kernelkonfiguration (siehe den folgenden Abschnitt) keinen Ausgangspunkt haben, müssen Sie sich wirklich um alle Kerneloptionen kümmern. Gerade beim ersten Mal ist es so gut wie sicher, dass Sie irgendetwas übersehen werden. Sie sparen eine Menge Zeit und Mühe, wenn Sie die mit Ihrer Distribution mitgelieferte Kernelkonfigurationsdatei als Startbasis verwenden:

```
cp old-config /pfad/zum/code/linux-n.n/.config
```

Dieses Verfahren hat leider einen Nachteil: Wenn der ursprüngliche Kernelcode andere Patches enthält als der neu zu kompilierende Code, enthält auch die ursprüngliche Konfigurationsdatei Optionen, die im neuen Code nicht vorgesehen sind. Das kann zu Problemen führen. Manche Distributoren bauen Patches in ihren Kernel ein, die im Standardkernel nicht enthalten sind. Deswegen müssen Sie anschließend in das Quellcodeverzeichnis wechseln und dort das folgende Kommando ausführen:

```
cd /pfad/zum/code/linux-n.n
make oldconfig                # bei Unklarheiten rückfragen
```

`make oldconfig` wertet die vorhandene `.config`-Datei aus. Fehlen dort Optionen, die der aktuelle Kernelcode vorsieht, dann werden entsprechende Rückfragen angezeigt. In der Regel können Sie einfach mit  antworten; dann gelten für diese Optionen die von den Entwicklern vorgeschlagenen Defaulteinstellungen. Genau das macht – ohne jede Rückfrage – das `make`-Kommando `olddefconfig`:

```
make olddefconfig            # bei Unklarheiten Defaults verwenden
```

Aktuelle Konfiguration feststellen

Jetzt bleibt noch die Frage offen, woher Sie die aktuelle Kernelkonfigurationsdatei für das Kommando `cp old-config` nehmen. Bei nahezu allen Distributionen befindet sich im Verzeichnis `/boot` die zum laufenden Kernel passende Konfigurationsdatei, also z. B. `/boot/config-n.n`. Somit wird aus `cp old-config` beispielsweise:

```
cd /pfad/zum/code/linux-n.n
cp /boot/config-6.2.11-300.fc38.x86_64.config .config
make oldconfig
```

Der mit SUSE mitgelieferte Kernel verwendet die `cloneconfig`-Option (Gruppe *General setup*). Das bedeutet, dass `/proc/config.gz` den komprimierten Inhalt der `.config`-Datei enthält, mit der der gerade laufende Kernel kompiliert wurde. Mit `make cloneconfig` kopieren Sie die zuletzt verwendete Konfiguration in die Datei `.config`.

Kernel manuell konfigurieren

Egal, ob Sie mit einer Grundkonfiguration oder bei null starten – mit `make xxxconfig` (auf die verschiedenen Varianten gehe ich im nächsten Abschnitt ein) können Sie nun sämtliche Kerneloptionen nach Ihrem Bedarf verändern bzw. einstellen. Dabei müssen Sie sich zwischen zwei prinzipiellen Kerneleypen entscheiden: einem monolithischen Kernel oder einem modularisierten Kernel. Ein monolithischer Kernel enthält alle benötigten Treiber und unterstützt keine Module. Modularisierte Kernel sind über die integrierten Treiber hinaus in der Lage, im laufenden Betrieb zusätzliche Module aufzunehmen. Ein modularisierter Kernel ist in fast allen Fällen die bessere Entscheidung.

Bei den meisten Kernel-Komponenten und -Treibern haben Sie die Wahl zwischen drei Optionen: YES, MODULE und NO.

- ▶ YES bedeutet, dass diese Komponente direkt in den Kernel integriert wird.
- ▶ MODULE bedeutet, dass diese Komponente als Modul kompiliert wird (nur sinnvoll bei einem modularisierten Kernel).
- ▶ NO bedeutet, dass die Komponente überhaupt nicht kompiliert wird.

Es gibt auch eine Reihe von Funktionen, die nicht als Module zur Verfügung gestellt werden können – dort reduziert sich die Auswahl auf YES oder NO.

Die übliche Vorgehensweise besteht darin, in den modularisierten Kernel nur relativ wenige elementare Funktionen zu integrieren und alle anderen Funktionen als Module verfügbar zu machen. Der Vorteil: Der Kernel an sich ist relativ klein, Module werden nur nach Bedarf nachgeladen.

Eine alternative Strategie besteht darin, einen monolithischen Kernel möglichst exakt für die eigenen Hard- und Software-Ansprüche zu optimieren. Alle Funktionen, die genutzt werden

sollen, integrieren Sie direkt in den Kernel. Bei allen anderen Komponenten entscheiden Sie sich für NO.

Generell wird ein monolithischer Kernel immer etwas größer als ein modularisierter Kernel. Dafür funktioniert er ohne die dynamische Modulverwaltung, und der Rechnerstart ist unter Umständen ohne Initrd-Datei möglich. (Das ist abhängig von den Randbedingungen. Eine Initrd-Datei kümmert sich, losgelöst von den Kernelmodulen, auch um die initiale Einstellung der Tastatur, die eventuell notwendige Eingabe von LUKS-Passwörtern für verschlüsselte Dateisysteme etc. Nur wenn Sie all diese Funktionen nicht brauchen, ist ein Start ohne Initrd-Datei denkbar.)

Der Nachteil eines monolithischen Kernels ist offensichtlich: Wenn Sie eine bestimmte Funktion später brauchen, müssen Sie den Kernel neu kompilieren. Und nur echte Linux-Profis können abschätzen, welche Funktionen sie nutzen werden.

Werkzeuge zur manuellen Kernelkonfiguration

Um abweichend von der aktuellen Konfiguration einzelne Einstellungen zu verändern, können Sie `.config` manuell editieren. Das ist aber fehleranfällig und erfordert eine gute Kenntnis der Namen der diversen Optionen. Besser ist es daher, mit `make xxxconfig` ein spezielles Konfigurationsprogramm zu starten. Dabei stehen unterschiedliche Varianten zur Verfügung, die Sie mit einem der aufgelisteten `make`-Kommandos starten:

```
cd /pfad/zum/code/linux-n.n
make menuconfig      # dialoggeführte Konfiguration im Textmodus
make nconfig         # dialoggeführte Konfiguration im Textmodus
make localmodconfig  # autom. Konfiguration für die aktuelle Hardware
```

`make menuconfig` und `make nconfig` setzen voraus, dass Sie vorher das Paket `ncurses-devel` bzw. `libncurses6-dev` installiert haben. Die Konfiguration erfolgt ebenfalls im Textmodus. Der große Vorteil im Vergleich zu `make config` besteht darin, dass die Einstellung der unzähligen Optionen durch verschachtelte Dialoge strukturiert ist (siehe Abbildung 26.1). Die Gestaltung der Dialoge und die Tastenkürzel variieren ein wenig, in ihrer Funktionsweise sind beide Varianten aber recht ähnlich.

`make localmodconfig` ist eine interessante Kompilervariante für alle, die es eilig haben. Dabei werden nur die Module kompiliert, die im gerade laufenden Kernel tatsächlich genutzt werden. Das hat Vor- und Nachteile: Der offensichtliche Vorteil besteht darin, dass wirklich nur der Teil des Kernelcodes übersetzt wird, der tatsächlich benötigt wird. Das kann die Übersetzungszeit auf ein Drittel senken!

Allerdings läuft der so kompilierte Kernel auf einem anderen Rechner unter Umständen nicht, wenn für seine Hardware-Komponenten relevante Treiber fehlen. Auch das Nachladen eines Moduls, das zur Kompilierzeit nicht aktiv war, wird scheitern. Der Kernel ist also nur

zu Testzwecken geeignet, nicht aber für eine längerfristige Nutzung. Detailinformationen zu dieser make-Variante können Sie hier nachlesen:

<https://heise.de/-1402386>

Wenn Sie nur einzelne Optionen an einer vorgegebenen Kernelkonfiguration ändern möchten, können Sie auf `make xxxconfig` ganz verzichten. Stattdessen geben Sie die gewünschten Einstellungen in der getrennten Datei `config-local` an. Die hier gewählten Optionen haben Vorrang gegenüber `.config`.

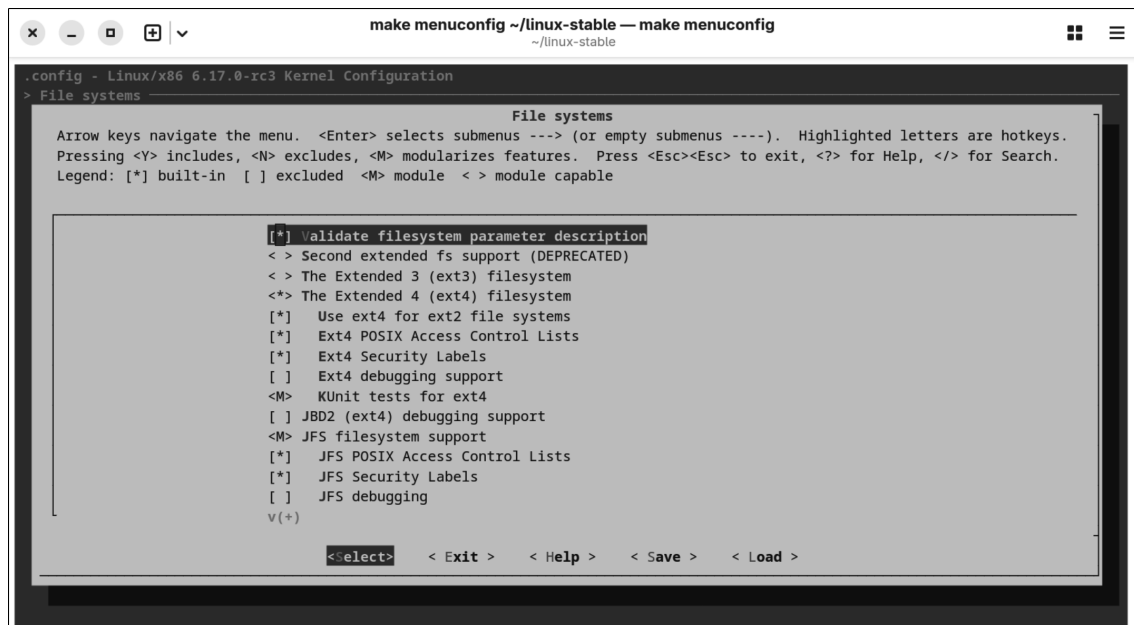


Abbildung 26.1 Kernelkonfiguration mit »make menuconfig«

Kernel kompilieren und installieren

Nachdem Sie mit der Konfiguration des Kernels vermutlich einige Zeit verbracht haben, muss jetzt der Rechner arbeiten. Die folgenden Kommandos beschäftigen einen zeitgemäßen Rechner circa 15 bis 30 Minuten. Die Option `-j N` gibt an, wie viele Prozesse `make` parallel starten soll. Das Kommando `nproc` ermittelt, wie viele virtuelle Cores zur Verfügung stehen. (Ich habe meine Tests auf einem Mini-PC mit der AMD-CPU 8745H durchgeführt. Die CPU besitzt 8 Cores und nutzt Hyperthreading, d. h. `nproc` liefert 16. Das `make`-Kommando nutzt somit alle Threads und führte zu einer nahezu hundertprozentigen CPU-Auslastung für 18 Minuten.)

```
cd /pfad/zum/code/linux-n.n
make -j$(nproc) all          # alles kompilieren, alle CPU-Cores nutzen
```

Das Ergebnis am Ende dieses Prozesses ist die Datei bzImage im Verzeichnis /pfad/zum/code/linux-n.n/arch/x86/boot. Die Größe der Datei liegt meist zwischen 10 und 20 MiB und hängt davon ab, wie viele Funktionen direkt in den Kernel inkludiert sind und wie viele als Module bzw. überhaupt nicht kompiliert wurden.

Module und Kernel installieren

`make modules_install` kopiert die Moduldateien dorthin, wo die Kommandos zur Modulverwaltung (etwa `insmod`) diese erwarten: in das Verzeichnis `/lib/modules/n.n`. Dabei ist `n.n` die Versionsnummer des soeben kompilierten Kernels. Während das Kompilieren ohne root-Rechte klappt, muss dieses Kommando mit `sudo` ausgeführt werden.

```
sudo make modules_install          # Module samt Debug-Infos installieren
```

Standardmäßig enthalten die neu erzeugten Kernelmodule eine Menge Debugging-Informationen. Sie sind deswegen viel größer als »gewöhnliche« Module und führen in der Folge zu riesigen Initrd-Dateien (bei meinen Tests z.B. 270 MiB anstatt 80 MiB!). Sofern Sie nicht vorhaben, sich auf die Suche nach Kernelfehlern zu begeben, sollten Sie die Debugging-Informationen aus den Modulen entfernen. (In der Kerneldokumentation wird dieser Vorgang »stripping« genannt.) Anstelle des obigen Kommandos führen Sie dazu die folgende Variante aus:

```
sudo make INSTALL_MOD_STRIP=1 modules_install # nur Module installieren
```

Der frisch erzeugte neue Kernel ist natürlich noch nicht aktiv. Bisher wurde nur eine Menge neuer Dateien erstellt, sonst nichts! Der neue Kernel kann erst beim nächsten Start von Linux aktiviert werden und auch dann nur, wenn Sie den Kernel in das `/boot`-Verzeichnis kopieren und eine dazu passende Initrd-Datei erzeugen. Außerdem ist es üblich, die beim Kompilieren verwendete Datei `.config` unter dem Namen `config-n.n` im `/boot`-Verzeichnis zu speichern. Damit wird dokumentiert, wie Ihr Kernel kompiliert wurde.

Dankenswerterweise kümmert sich `make install` um sämtliche gerade zusammengefassten Punkte:

```
sudo make install          # Kerneldateien installieren
```

Falls Ihr System Kernelmodule benötigt, deren Code außerhalb des offiziellen Kernelcodes liegt (typischerweise der NVIDIA-Treiber), müssen auch diese Module neu kompiliert und berücksichtigt werden. Das erfolgt bei vielen Distributionen mittels DKMS. Beachten Sie, dass das Kompilieren des NVIDIA-Treibers bei neuen Kernelversionen oft scheitert bzw. die allerneueste Version des NVIDIA-Treibers voraussetzt!

Ich habe meine Tests auf einem Rechner mit deaktiviertem UEFI Secure Boot durchgeführt. Wenn Sie Secure Boot nutzen möchten, müssen Sie einen *Machine Owner Key* (MOK) erstellen, mit `mokutil` in die MOK-Datenbank einfügen und den Kernel sowie alle Module damit signieren.

Details zum Signiervorgang können Sie hier nachlesen:

https://docs.fedoraproject.org/en-US/quick-docs/kernel-build-custom/#_secure_boot
https://wiki.archlinux.org/title/Unified_Extensible_Firmware_Interface/Secure_Boot
https://wiki.debian.org/SecureBoot#MOK_-_Machine_Owner_Key

Zuletzt müssen Sie die GRUB-Konfiguration aktualisieren:

```
sudo update-grub # Debian, Ubuntu
sudo grub2-mkconfig -o /boot/grub2/grub.cfg # Fedora, RHEL
```

Neustart und Aufräumarbeiten

Ob alles funktioniert hat, merken Sie beim Neustart:

```
uname -a

Linux fedora 6.17.0-rc3 ...
```

Sollte der neue Kernel aus irgendeinem Grund nicht funktionieren, starten Sie den Rechner einfach mit dem bisherigen Kernel und unternehmen einen weiteren Versuch, den Kernel richtig zu konfigurieren und neu zu kompilieren. Läuft der neue Kernel dagegen zufriedenstellend, sollten Sie die nun nicht mehr benötigten Objekt-Dateien des Compilers aufräumen. Sie gewinnen auf diese Weise 25 bis 30 GiB Platz auf Ihrer SSD zurück! Allerdings verlängert sich ein eventueller weiterer Kompiliervorgang dadurch erheblich.

```
cd /pfad/zum/code/linux-n.n
make clean
```

26.5 Kernel-Live-Patches

Die meisten Updates eines Linux-Systems können im laufenden Betrieb erfolgen. Aktualisierte Netzwerkdienste müssen zwar anschließend neu gestartet werden, aber es besteht keine Notwendigkeit, den ganzen Rechner neu zu starten. Die Ausnahme von dieser Regel ist der Kernel: Damit Sicherheits-Updates im Kernel wirksam werden, müssen Sie einen neuen Kernel und neue Module installieren und anschließend den ganzen Rechner neu starten.

Auf Rechnern, die jeden Tag oder zumindest einmal pro Woche ein- und ausgeschaltet werden, ist das egal. Aber bei Servern, die möglichst ohne Unterbrechung ständig verfügbar sein sollen, ist ein Neustart zumeist unerwünscht. Und selbst Administratoren, die Updates automatisieren, scheuen in der Regel davor zurück, auch die erforderlichen Neustarts zu automatisieren. Zu groß ist die Gefahr, dass etwas schiefgeht und dann (zumeist mitten in der Nacht) keiner Zeit hat, das Problem sofort zu beheben.

Ksplice (Oracle)

Die erste Lösung für dieses Problem bot die Funktion *Ksplice*: Bei vielen Updates ist es möglich, die betreffende Kernelfunktion im laufenden Betrieb zu deaktivieren und durch neuen Code zu ersetzen. Die nicht ganz trivialen technischen Hintergründe des Verfahrens sind auf den folgenden Seiten beschrieben:

<https://ksplice.oracle.com>

<https://lwn.net/Articles/340477>

<https://en.wikipedia.org/wiki/Ksplice>

2011 übernahm Oracle die Firma Ksplice. Mit der Funktion Ksplice durchgeführte Kernel-Updates waren über mehrere Jahre ein durchaus gewichtiges Unterscheidungsmerkmal zu anderen Enterprise-Linux-Versionen. Oracle bietet Ksplice allerdings nur zahlenden Kunden an. Ksplice steht zwar als Open-Source-Code zur Verfügung, ist aber nicht Bestandteil des offiziellen Linux-Kernels.

kPatch und kGraft (Red Hat und SUSE)

Red Hat und SUSE wollten in dieser Hinsicht natürlich nicht zurückstecken und entwickelten unter den Namen *kPatch* und *kGraft* vergleichbare Update-Mechanismen. Die beiden Mechanismen stehen seit 2014 in den Enterprise-Versionen von Red Hat und SUSE zur Verfügung. Die für kPatch und kGraft erforderlichen Funktionen (gewissermaßen der größte gemeinsame Nenner) wurden von Linus Torvalds 2015 in den offiziellen Linux-Kernel aufgenommen. Ob Ihr Kernel die Funktionen enthält, erkennen Sie am Vorhandensein des Verzeichnisses `/sys/kernel/livepatch`. Allerdings stellen auch Red Hat und SUSE geeignete Live-Kernel-Patches nur zahlenden Kunden zur Verfügung.

<https://github.com/dynup/kpatch>

<https://en.wikipedia.org/wiki/KGraft>

Ubuntu

Canonical trat zuletzt in das Live-Patching-Geschäft ein. Seit Ende 2016 bietet Canonical seinen kommerziellen Kunden für kritische Sicherheitsprobleme in Ubuntu-LTS-Versionen Live-Patches an, zuerst im Rahmen der Landscape-Dienste, mittlerweile unter der Bezeichnung »Ubuntu Pro«. Das zugrunde liegende Programm `canonical-livepatch` greift dabei auf die vorhin erwähnte Live-Patch-Infrastruktur im Kernel zurück.

Erfreulicherweise stellt Canonical die Live-Patches in beschränktem Umfang auch nichtkommerziellen Anwenderinnen und Anwendern zur Verfügung: Sofern Sie über ein kostenloses Ubuntu-One-Konto verfügen, erhalten Sie einen Token, mit dem Sie fünf Rechner in das Ubuntu-Pro-Programm aufnehmen können:

```

sudo apt install pro      # ist in der Regel bereits installiert

sudo pro attach <token>

...
This machine is attached to 'Ubuntu Pro - free personal subscription'

SERVICE      ENTITLED  STATUS    DESCRIPTION
esm-apps       yes      enabled   Expanded Security Maintenance for ...
esm-infra      yes      enabled   Expanded Security Maintenance for ...
fips           yes      disabled  NIST-certified core packages
fips-updates   yes      disabled  NIST-certified core packages with ...
livepatch      yes      enabled   Canonical Livepatch service
usg            yes      disabled  Security compliance and audit tools

```

Im Rahmen der Ubuntu-Pro-Aktivierung wird auch der Live-Patch-Dienst eingerichtet (siehe die vorletzte Zeile im obigen Listing). Die erforderliche Software steht ausschließlich als Snap-Paket zur Verfügung. Den Live-Patch-Status können Sie mit `canonical-livepatch status` ermitteln:

```

sudo canonical-livepatch status

last check: 22 minutes ago
kernel: 6.8.0-64.67-generic
server check-in: succeeded
kernel state: kernel series 6.8 is covered by Livepatch
patch state:  no livepatches available for kernel 6.8.0-64.67-generic
tier: updates (Free usage; This machine beta tests new patches.)

```

Achten Sie auf das Kleingedruckte!

Die Statusmeldung enthält einen Hinweis, der leicht zu übersehen ist: *This machine beta tests new patches*. Konkret bedeutet das, dass Sie als nicht zahlender Ubuntu-Pro-Nutzer ein Beta-Tester für Kernel-Updates sind! Canonical liefert Kernel-Patches zuerst an die nicht zahlenden Kunden aus. Wenn das zu keinen Problemen führt, werden die gleichen Updates später auch den zahlenden Kunden zur Verfügung gestellt.

Grundsätzlich ist diese Vorgehensweise nachvollziehbar. Ich nutze das kostenlose Ubuntu-Pro-Angebot auf mehreren Rechnern/Servern und bin bisher sehr zufrieden damit. Ärgerlich ist aber die intransparente Informationspolitik von Canonical. Obwohl ich eine Weile danach gesucht habe, bin ich im Internet auf kein offizielles Dokument gestoßen, das deutlich auf den Beta-Charakter der Kernel-Updates bei der kostenlosen Nutzung hinweist.

Wenn Sie detaillierte Informationen zu den behobenen Sicherheitslücken wünschen, geben Sie zusätzlich die Option `--verbose` an. `dmesg | grep livepatch` liefert Meldungen über zuletzt durchgeführte Live-Patches.

Live-Patches nur für Notfälle

Den Code des laufenden Kernels zu verändern, ist ein diffiziler Vorgang (wenn Sie dramatische Vergleiche mögen: wie die Operation am offenen Herzen). Deswegen wird dieser Mechanismus aktuell von allen Distributoren nur für gefährliche Sicherheitslücken im Kernel verwendet.

Bei sonstigen Korrekturen wird wie bisher ein neuer Kernel installiert, auf die Aktivierung der dort durchgeführten Änderungen via Live-Patches aber verzichtet. Somit kann es trotz aktivierter Live-Patches sein, dass Ihre Distribution nach der Installation von (Kernel-)Updates meldet, dass ein Neustart erforderlich ist. Lassen Sie sich davon nicht verunsichern.

26.6 Die Verzeichnisse `/proc` und `/sys`

Die Verzeichnisse `/proc` und `/sys` werden während des Systemstarts in das Dateisystem eingebunden. Sie dienen dazu, Informationen über den Kernel, laufende Prozesse, geladene Module und viele andere Parameter auf eine transparente Art und Weise sichtbar zu machen. Intern sind die Verzeichnisse `/proc` und `/sys` als virtuelle Dateisysteme realisiert. Sie enthalten also keine echten Dateien und beanspruchen daher auch keinen Platz auf der Festplatte. Das gilt auch für die scheinbar sehr große Datei `/proc/kcore`, die den Arbeitsspeicher abbildet.

Die meisten der `/proc`- und `/sys`-Dateien liegen im Textformat vor. Um die Dateien zu lesen, müssen Sie unter Umständen `cat` statt `less` verwenden, weil manche `less`-Versionen mit virtuellen Dateien nicht zurechtkommen.

Das `/proc`-Verzeichnis liefert interne Kernelinformationen sowie Daten zu allen laufenden Prozessen (siehe Tabelle 26.5). Unter anderem ist dort jedem Prozess ein eigenes Unterverzeichnis zugeordnet. Innerhalb des Prozessverzeichnisses befinden sich dann einige Dateien mit diversen Verwaltungsdaten (z. B. die zum Start verwendete Kommandozeile). Diese Daten werden von diversen Kommandos zur Prozessverwaltung (z. B. `top`, `ps` etc.) ausgewertet.

Datei	Bedeutung
<code>/proc/N/*</code>	Informationen zum Prozess mit der PID=N
<code>/proc/asound</code>	ALSA (Advanced Linux Sound Architecture)
<code>/proc/bus/usb/*</code>	USB-Informationen

Tabelle 26.5 Wichtige `/proc`-Dateien

Datei	Bedeutung
/proc/bus/pccard/*	PCMCIA-Informationen
/proc/bus/pci/*	PCI-Informationen
/proc/cmdline	an den Kernel übergebene Parameter
/proc/config.gz	Kernelkonfigurationsdatei (SUSE)
/proc/cpuinfo	CPU-Informationen
/proc/devices	Nummern von aktiven Devices
/proc/fb	Informationen zum Frame-Buffer
/proc/filesystems	im Kernel enthaltene Dateisystemtreiber
/proc/interrupts	Nutzung der Interrupts
/proc/lvm/*	Nutzung des Logical Volume Managers
/proc/mdstat	RAID-Zustand
/proc/modules	aktive Module
/proc/mounts	aktive Dateisysteme
/proc/net/*	Netzwerkzustand und -nutzung
/proc/partitions	Partitionen der Festplatten
/proc/scsi/*	SCSI/SATA-Geräte und -Controller
/proc/sys/*	System- und Kernelinformationen
/proc/uptime	Zeit in Sekunden seit dem Rechnerstart
/proc/version	Kernelversion

Tabelle 26.5 Wichtige /proc-Dateien (Forts.)

Das /sys-Verzeichnis enthält teilweise dieselben Informationen wie /proc, allerdings sind die Daten systematischer organisiert (siehe Tabelle 26.6). Das Ziel des /sys-Verzeichnisses ist es, den Zusammenhang zwischen dem Kernel und der Hardware abzubilden.

Datei	Bedeutung
/sys/block/*	Informationen über alle Block-Devices (Festplatten etc.)
/sys/bus/*	Informationen über alle Bus-Systeme (PCI, SCSI, USB etc.)

Tabelle 26.6 Wichtige /sys-Dateien

Datei	Bedeutung
/sys/class/*	Informationen über Device-Klassen (Bluetooth, Grafik etc.)
/sys/devices/*	Informationen über angeschlossene Hardware-Komponenten
/sys/firmware/*	Informationen über Hardware-Treiber und -Firmware
/sys/kernel/*	Informationen über den Kernel
/sys/module/*	Informationen über geladene Module
/sys/power/*	Informationen über die Energieverwaltung

Tabelle 26.6 Wichtige /sys-Dateien (Forts.)

26.7 Kernel-Boot-Optionen

Nicht immer, wenn ein Detail im Kernel geändert werden soll, muss der Kernel gleich neu kompiliert werden! Es gibt zwei Möglichkeiten, ohne ein Neukompilieren auf den Kernel Einfluss zu nehmen:

- Zum einen können Sie mit dem Bootloader während des Systemstarts Parameter an den Kernel übergeben. Dieser Mechanismus ist Thema dieses Abschnitts.
- Zum anderen können Sie eine Reihe von Kernelfunktionen dynamisch – also im laufenden Betrieb – verändern. Diese Art des Eingriffs ist insbesondere zur Steuerung von Netzwerkfunktionen gebräuchlich und wird in Abschnitt 26.8, »Kernelparameter verändern«, beschrieben.

GRUB

Bei der Konfiguration von GRUB können Sie in der Zeile `linux` bzw. in der Datei `/etc/default/grub` Kernel-Boot-Optionen angeben. Derartige Optionen können Sie auch interaktiv beim Start eines Linux-Installationsprogramms oder beim Start von GRUB über die Tastatur eintippen (siehe Abschnitt 24.3, »GRUB-Bedienung (Anwendersicht)«). Die Syntax für die Angabe von Optionen sieht so aus:

```
linux /boot/vmlinuz-n.n optionA=parameter optionB=parameter1,parameter2
```

Die Parameter zu einer Option müssen ohne Leerzeichen angegeben werden. Mehrere Optionen müssen durch Leerzeichen voneinander getrennt werden, nicht durch Kommata. Hexadezimale Adressen werden in der Form `0x1234` angegeben. Ohne vorangestelltes `0x` wird die Zahl dezimal interpretiert.

Kernel-Boot-Optionen helfen oft dabei, Hardware-Probleme zu umgehen. Wenn der Linux-Kernel beispielsweise aufgrund eines fehlerhaften BIOS nicht erkennt, wie viel RAM Ihr Rechner hat, geben Sie den korrekten Wert mit dem Parameter `mem=` an.

Beachten Sie, dass die beim Linux-Start angegebenen Parameter nur Einfluss auf die in den Kernel integrierten Treiber haben! Parameter für Kernelmodule müssen dagegen in der Datei `/etc/modprobe.conf` bzw. in den Verzeichnissen `/etc/modprobe.d` oder `/etc/modules-load.d` angegeben werden.

Dieser Abschnitt beschreibt nur die wichtigsten Kernel-Boot-Optionen. Weitere Informationen erhalten Sie mit `man bootparam` sowie auf den folgenden Seiten:

<https://tldp.org/HOWTO/BootPrompt-HOWTO.html>

<https://www.kernel.org/doc/Documentation/admin-guide/kernel-parameters.txt>

Wichtige Kernel-Boot-Optionen

- `root=/dev/sdb3`: Die `root`-Option gibt an, dass nach dem Laden des Kernels die dritte primäre Partition des zweiten SATA-Laufwerks als Systempartition für das Root-Dateisystem verwendet werden soll. Analog können natürlich auch andere Laufwerke und Partitionen angegeben werden.

Wenn die Partition mit einem Label bezeichnet ist, kann die Systempartition auch in der Form `root=LABEL=xxx` angegeben werden. Bei ext-Partitionen ermitteln Sie den Partitionsnamen mit `e2label` bzw. verändern ihn mit `tune2fs`.

Eine weitere Variante ist die Angabe der Systempartition durch `root=UUID=nnn`, wobei `nnn` die UUID des Dateisystems ist. Im interaktiven Betrieb von GRUB ist die Eingabe von UUIDs allerdings sehr mühsam. Die UUID ermitteln Sie im laufenden Betrieb mit `blkid <device>`.

- `ro`: Die Option `ro` gibt an, dass das Dateisystem vorerst *read-only* gemountet werden soll. Das ist (in Kombination mit einer der beiden folgenden Optionen) praktisch, wenn ein defektes Dateisystem manuell repariert werden muss.
- `init`: Nach dem Kernelstart wird bei den meisten Distributionen `systemd` gestartet (siehe Kapitel 25). Wenn Sie dies nicht wollen, können Sie mit der Option `init` ein anderes Programm angeben.

Mit `init=/bin/sh` erreichen Sie beispielsweise, dass eine Shell gestartet wird. Die Option kann Linux-Profis helfen, ein Linux-System wieder zum Laufen zu bringen, wenn bei der Init-Konfiguration etwas schiefgegangen ist. Beachten Sie, dass das root-Dateisystem nur *read-only* zur Verfügung steht. (Das können Sie mit `mount -o remount` ändern, siehe Abschnitt 23.8, »mount und `/etc/fstab`«.) Beachten Sie außerdem, dass in der Konsole das US-Tastaturlayout gilt und dass die `PATH`-Variable noch leer ist.

- ▶ `single` oder `emergency`: Wenn Sie eine dieser Optionen verwenden, startet der Rechner im Single-User-Modus (Init-V) bzw. im Rescue-Modus (systemd). Genau genommen werden diese Optionen nicht vom Kernel ausgewertet, sondern so wie alle unbekannten Optionen an das erste vom Kernel gestartete Programm weitergegeben (siehe Kapitel 25, »systemd«).
- ▶ `initrd=name`: `initrd` gibt den Namen der zu ladenden Initial-RAM-Disk-Datei an. Wenn Sie *keine* Initrd-Datei verwenden möchten, geben Sie `initrd=` oder `noinitrd` an.
- ▶ `ipv6.disable=1`: Diese Option deaktiviert alle IPv6-Funktionen des Kernels.
- ▶ `reserve=0x300,0x20`: Diese Option gibt an, dass die 32 Bytes (hexadezimal 0x20) zwischen 0x300 und 0x31F von keinem Hardware-Treiber angesprochen werden dürfen, um darin nach irgendwelchen Komponenten zu suchen. Die Option ist bei manchen Komponenten notwendig, die auf solche Tests allergisch reagieren. Sie tritt im Regelfall in Kombination mit einer zweiten Option auf, die die exakte Adresse der Komponente angibt, die diesen Speicherbereich für sich beansprucht.
- ▶ `pci=bios|nobios|nommconf`: Diese Option steuert, wie die Hardware-Erkennung von PCI-Komponenten erfolgt. Sollten dabei Probleme auftreten, hilft manchmal `pci=bios` oder `pci=nommconf`.
- ▶ `quiet`: Diese Option bewirkt, dass während des Kernelstarts keine Meldungen auf dem Bildschirm dargestellt werden.
- ▶ `video=1024x768`: Mit dieser Option kann per *Kernel Mode Setting* (KMS) die gewünschte Grafikauflösung eingestellt werden, falls der Kernel nicht selbst die optimale Auflösung wählt, z. B. wenn das Video-Signal über einen KVM-Switch geleitet wird. Die Option kann auch helfen, die Installation einer Linux-Distribution in einer geringeren Auflösung durchzuführen (praktisch bei HiDPI-Monitoren, wenn die Schrift unleserlich klein ist).

Wenn Sie auch die Farbtiefe (z. B. 24 Bit) und die Bildfrequenz angeben möchten, sieht die Syntax so aus: `video=1280x800-24@60` Die Einstellung der Grafikdaten funktioniert nur bei KMS-kompatiblen Treibern. Die `video`-Einstellung gilt normalerweise für alle angeschlossenen Monitore. Wenn Sie die Auflösung nur für einen einzelnen Monitor ändern möchten, geben Sie den entsprechenden Signalausgang an, z. B. `video=VGA-1:1024x768`.

- ▶ `nomodeset`: Diese Option deaktiviert das Kernel Mode Setting (KMS). Sie verhindert, dass bei einem Notebook mit NVIDIA-Grafikkarte, aber ohne die erforderlichen NVIDIA-Treiber der Bildschirm beim Start des Grafiksystems schwarz wird.
- ▶ `mitigations=auto|auto,nosmt|off`: Diese Option steuert, welche Schutzmaßnahmen der Kernel gegen CPU-Fehler ergreifen soll (siehe Abschnitt 26.9, »Spectre und Meltdown«).
- ▶ `dis_ucode_ld`: Diese Option verhindert, dass der Kernel Microcode-Updates an die CPU weitergibt (siehe Abschnitt 21.7, »Firmware-, BIOS- und EFI-Updates«). Sie sollte nur verwendet werden, wenn es aufgrund eines derartigen Updates zu Abstürzen oder Instabilitäten kommt.

SMP-Optionen

SMP steht für *Symmetric Multiprocessing* und bezeichnet die Fähigkeit des Kernels, mehrere CPUs bzw. CPU-Cores gleichzeitig zu nutzen. Sollten dabei Probleme auftreten, können die folgenden Optionen hilfreich sein:

- ▶ `maxcpus=1`: Wenn Sie bei einem Multiprozessorsystem Boot-Probleme haben, können Sie mit dieser Option die Anzahl der genutzten Prozessoren auf 1 reduzieren. Der Wert 0 entspricht der Option `nosmp`.
- ▶ `nosmp`: Diese Option deaktiviert die SMP-Funktionen. Der Kernel nutzt nur eine CPU.
- ▶ `noht`: Diese Option deaktiviert die Hyper-Threading-Funktion. Dank dieser Funktion verhalten sich viele CPUs so, als stünden mehrere Cores zur Verfügung. Daraus ergibt sich eine etwas höhere Rechenleistung, wenngleich die Steigerung nicht so hoch ist wie bei echtem SMP.
- ▶ `lapic`: APIC steht für *Advanced Programmable Interrupt Controller* und bezeichnet ein Schema, um Hardware-Interrupts an die CPUs weiterzuleiten. Bei aktuellen Kernelversionen wird APIC immer aktiviert. Wenn Sie Probleme mit APIC vermuten, verhindern Sie durch `lapic`, dass der Kernel den lokalen APIC aktiviert bzw. nutzt.
- ▶ `noapic`: Diese Option reicht etwas weniger weit als `lapic` und deaktiviert nur den IO-Teil von APIC.
- ▶ `lapic`: Diese Option aktiviert APIC explizit. Das ist dann notwendig, wenn APIC durch das BIOS deaktiviert ist, aber dennoch genutzt werden soll.

ACPI-Optionen

Das Energiemanagementsystem ACPI (*Advanced Configuration and Power Interface*) ist nicht nur für das Ein- und Ausschalten verantwortlich, sondern auch für den sparsamen Umgang mit Energie, für die Verwaltung verschiedener Hibernat-Modi etc. Im Folgenden sind die wichtigsten Optionen zur Steuerung der ACPI-Funktionen des Kernels zusammengefasst:

- ▶ `acpi=on/off`: Diese Option (de)aktiviert die ACPI-Funktionen im Kernel.
- ▶ `acpi=oldboot`: Damit werden die ACPI-Funktionen nur während des Boot-Vorgangs genutzt. Sobald der Rechner läuft, werden die ACPI-Funktionen aber nicht mehr verwendet.
- ▶ `pci=noacpi`: Diese Option deaktiviert die Interrupt-Zuweisungen durch ACPI.
- ▶ `noresume`: Diese Option bewirkt, dass vorhandene Hibernat-Daten in der Swap-Partition ignoriert werden. Sie ist also dann sinnvoll, wenn der Rechner nicht mehr richtig aufwacht, z. B., weil die Hibernat-Daten defekt sind.

26.8 Kernelparameter verändern

Viele Parameter des Kernels können im laufenden Betrieb über das `/proc`-Dateisystem verändert werden. Das folgende Beispiel zeigt, wie Sie die Masquerading-Funktion aktivieren, um den Rechner als Internet-Gateway für andere Rechner einzusetzen:

```
sudo sh -c 'echo 1 > /proc/sys/net/ipv4/ip_forward'
```

Einen eleganteren Weg bietet das Kommando `sysctl`, das mit den meisten aktuellen Distributionen mitgeliefert wird. Das analoge Kommando, um das Masquerading wieder abzuschalten, würde so aussehen:

```
sudo sysctl -w net.ipv4.ip_forward=0
```

`sysctl -a` liefert eine Liste aller Kernelparameter zusammen mit ihren aktuellen Einstellungen. Mit `sysctl -p` können die in einer Datei gespeicherten `sysctl`-Einstellungen aktiviert werden. Als Dateiname wird üblicherweise `/etc/sysctl.conf` verwendet.

Die Syntax ist in der Manual-Seite zu `sysctl.conf` beschrieben. Viele Distributionen sehen vor, dass diese Datei während des Init-Prozesses automatisch ausgewertet und ausgeführt wird.

26.9 Spectre und Meltdown

2018 wurden gravierende Design-Schwächen in den CPUs mehrerer Hersteller bekannt. Besonders betroffen war Intel. Durch Exploits konnte ein Prozess unerlaubterweise die Daten anderer Prozesse lesen. Die ersten derartigen Sicherheitslücken erhielten die klingenden Namen *Spectre* und *Meltdown*. Seither wurden diverse weitere verwandte Probleme entdeckt: *Fallout*, *Foreshadow*, *RIDL*, *ZombieLoad* usw.:

https://en.wikipedia.org/wiki/Transient_execution_CPU_vulnerability

Da ein Austausch aller betroffenen CPUs weder technisch noch wirtschaftlich möglich ist, wird seither versucht, die Fehler auf verschiedene Arten zu umgehen. Die vier Hauptansatzpunkte sind dabei:

- ▶ Microcode-Updates für die CPU (siehe auch Abschnitt 21.7, »Firmware-, BIOS- und EFI-Updates«)
- ▶ Änderungen an den Compilern, um Code zu generieren, der eine Ausnutzung der Fehler verhindert
- ▶ Änderungen im Kernel (das betrifft neben Linux auch Windows, macOS etc.) mit derselben Zielsetzung
- ▶ Änderungen an besonders betroffenen Programmen, z. B. an Webbrowsern und an Virtualisierungssystemen

Tatsächlich ist es mit diesen Maßnahmen gelungen, viele (wenn auch nicht alle) Fehler zu umgehen. Der Preis dafür ist allerdings hoch: Benchmarktests von Michael Larabel haben gezeigt, dass die Performance der so abgesicherten Rechner je nach Anwendung und CPU erheblich gesunken ist:

<https://www.phoronix.com/search/Spectre>

Das führt zur absurden Situation, dass die CPU-Hersteller nicht etwa für ihre Designfehler verantwortlich gemacht werden, sondern indirekt sogar davon profitieren: Die verminderte Rechenleistung macht Hardware-Updates früher erforderlich als geplant.

Um festzustellen, von welchen Problemen Ihr Rechner bzw. Ihre CPU betroffen ist und welche Sicherheitslücken durch den Kernel behoben werden, werfen Sie einen Blick in die Dateien des Verzeichnisses `/sys/devices/system/cpu/vulnerabilities`. Die folgenden Ergebnisse stammen von einem etwas älteren Notebook mit einer CPU von Intel (i7-8750H). Ich habe die Spalten des Listings eingerückt, um die Ergebnisse besser lesbar zu präsentieren.

```
cd /sys/devices/system/cpu/vulnerabilities
```

```
grep "" *
```

```
gather_data_sampling: Mitigation: Microcode
ghostwrite:          Not affected
indirect_target_sel: Not affected
itlb_multihit:KVM:   Mitigation: Split huge pages
l1tf:                Mitigation: PTE Inversion;
                     VMX: conditional cache flush, SMT vulnerable
mds:                 Mitigation: Clear CPU buffers; SMT vulnerable
meltdown:            Mitigation: PTI
mmio_stale_data:     Mitigation: Clear CPU buffers; SMT vulnerable
old_microcode:       Not affected
reg_file_data_sampling: Not affected
retbleed:            Mitigation: IBRS
spec_rstack_overflow: Not affected
spec_store_bypass:   Mitigation: Speculative Store Bypass disabled
                     via prctl
spectre_v1:          Mitigation: usercopy/swapgs barriers and
                     __user pointer sanitization
spectre_v2:          Mitigation: IBRS;
                     IBPB: conditional; STIBP: conditional; RSB
filling;
                     PBRSE-eIBRS: Not affected;
                     BHI: Not affected
srbds:Mitigation:    Microcode
tsa:                 Not affected
tsx_async_abort:     Not affected
```

Wesentlich mehr Details verrät das Script `spectre-meltdown-checker.sh`:

```
git clone https://github.com/speed47/spectre-meltdown-checker.git
```

```
cd spectre-meltdown-checker
```

```
sudo ./spectre-meltdown-checker.sh
```

```
Spectre and Meltdown mitigation detection tool v0.46-29-g34c6095
```

```
Checking for vulnerabilities on current system
```

```
Kernel is Linux 6.16.0-5-cachyos #1 SMP PREEMPT_DYNAMIC
```

```
CPU is Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz
```

```
Hardware check
```

- * Hardware support (CPU microcode) for mitigation techniques
 - * Indirect Branch Restricted Speculation (IBRS)
 - * SPEC_CTRL MSR is available: YES
 - * CPU indicates IBRS capability: YES (SPEC_CTRL feature bit)
 - * Indirect Branch Prediction Barrier (IBPB)
 - * CPU indicates IBPB capability: YES (SPEC_CTRL feature bit)
 - * Single Thread Indirect Branch Predictors (STIBP)
 - * SPEC_CTRL MSR is available: YES
 - * CPU indicates STIBP capability: YES (Intel STIBP feature bit)

```
...
```

- * CPU vulnerability to the speculative execution attack variants

```
* Affected by CVE-2017-5753 (Spectre Variant 1): YES
```

```
* Affected by CVE-2017-5715 (Spectre Variant 2): YES
```

```
* Affected by CVE-2017-5754 (Variant 3, Meltdown): YES
```

```
...
```

```
> SUMMARY:
```

```
CVE-2017-5753:OK   CVE-2018-3640:OK   CVE-2018-3620:OK   ...
```

```
CVE-2017-5715:OK   CVE-2018-3639:OK   CVE-2018-3646:OK
```

```
CVE-2017-5754:OK   CVE-2018-3615:OK   CVE-2018-12126:OK
```

```
Need more details about mitigation options? Use --explain
```

```
A false sense of security is worse than no security at all,
```

```
see --disclaimer
```

Schutzmechanismen deaktivieren

Standardmäßig sind im Kernel fast alle gerade verfügbaren Schutzmaßnahmen aktiv. Die einzige Ausnahme betrifft Hyperthreading bzw. *Simultaneous Multithreading* (SMT). Das ist

ein Verfahren, mit dem eine CPU mehr Threads parallel ausführen kann, als echte Cores zur Verfügung stehen. SMT bringt zwar einen deutlichen Performance-Schub, es müsste aus Sicherheitsgründen aber eigentlich deaktiviert werden. Aktuell sind Angriffe, die darauf abzielen, aber nur schwierig umzusetzen.

Für jedes der im Kernel implementierten Schutzverfahren gibt es eigene Kernelparameter, um das jeweilige Verfahren zu deaktivieren: `nospectre_v1`, `nospectre_v2`, `nopti` usw. Darüber hinaus gibt es den Parameter `mitigations`, der drei Einstellungen vorsieht:

- ▶ `mitigations=auto`: In der Defaulteinstellung sind alle verfügbaren Schutzmaßnahmen aktiviert. SMT bleibt aber aktiv.
- ▶ `mitigations=auto,nosmt`: Diese Einstellung deaktiviert SMT und ist noch sicherer. Sofern Ihre CPU Multithreading unterstützt, sinkt die Performance von Anwendungen mit vielen Threads aber nochmals deutlich.
- ▶ `mitigations=off`: Diese Einstellung deaktiviert alle Schutzmaßnahmen und macht Ihren Rechner schneller. Empfehlenswert ist das aber nur, wenn Sie sich vor eventuellen Angriffen sicher fühlen. Das lässt sich in manchen Fällen durchaus befürworten: So wurden die Sicherheitslücken Spectre, Meltdown & Co. bisher zwar theoretisch nachgewiesen, es gibt aber zurzeit kaum bekannte Schadsoftware, die diese Sicherheitslücken tatsächlich ausnutzt.

Insofern ist die Versuchung groß, in bestimmten Anwendungsfällen auf den Schutz zu verzichten – z. B. auf Entwicklungs- oder Labor-Rechnern, auf denen häufig CPU-intensive Anwendungen laufen. Vollkommen tabu ist diese Option jedoch auf allen Servern, die virtuelle Maschinen unterschiedlicher Besitzer oder Kunden ausführen.

Um die `mitigations`-Einstellungen einmal auszuprobieren, öffnen Sie während des Boot-Vorgangs mit `[E]` den Editor für den betreffenden GRUB-Menüeintrag und fügen am Ende der `linux`-Zeile `mitigations=...` hinzu. Um die Option dauerhaft einzustellen, verändern Sie die Zeile `GRUB_CMDLINE_LINUX_DEFAULT` in `/etc/default/grub` und aktualisieren dann die GRUB-Konfiguration mit `update-grub` bzw. mit `grub2-mkconfig`.

Auf einen Blick

Teil I

Installation..... 25

Teil II

Linux anwenden..... 103

Teil III

Linux-Grundlagen..... 227

Teil IV

Text- und Code-Editoren..... 421

Teil V

Systemkonfiguration und Administration 469

Teil VI

Server-Konfiguration 865

Teil VII

Sicherheit..... 1123

Teil VIII

Virtualisierung, Container und Co. 1259



Inhalt

Vorwort	19
---------------	----

TEIL I Installation

1 Was ist Linux?	27
1.1 Einführung	27
1.2 Hardware-Unterstützung	28
1.3 Distributionen	29
1.4 Open-Source-Lizenzen (GPL & Co.)	34
1.5 Die Geschichte von Linux	37
 2 Installationsgrundlagen	41
2.1 Voraussetzungen	41
2.2 BIOS und EFI	42
2.3 Installationsvarianten	45
2.4 Überblick über den Installationsprozess	48
2.5 Grundlagen der Partitionierung	50
2.6 LVM und Verschlüsselung	53
2.7 Linux-Partitionen anlegen	56
2.8 Installationsumfang festlegen	60
2.9 Grundkonfiguration	61
2.10 Probleme beheben	63
2.11 Systemveränderungen, Erweiterungen, Updates	65
2.12 Linux wieder entfernen	67
 3 Installationsanleitungen	69
3.1 Die Qual der Wahl	69
3.2 Debian	74
3.3 Fedora	83

3.4	Ubuntu	89
3.5	CachyOS	96

TEIL II Linux anwenden

4	Gnome	105
4.1	Erste Schritte	107
4.2	Dateimanager	114
4.3	Systemkonfiguration	121
4.4	Gnome Tweaks	130
4.5	Gnome-Shell-Erweiterungen	131
4.6	Screenshots	136
4.7	Freigaben und Fernanmeldung	136
4.8	Gnome-Interna	141
5	KDE	147
5.1	Bedienung	149
5.2	Dateimanager	152
5.3	KDE-Konfiguration	155
5.4	Screenshots	159
6	Desktop-Apps	161
6.1	Firefox	162
6.2	Google Chrome und Chromium	163
6.3	Thunderbird	165
6.4	Multimedia-Grundlagen	169
6.5	Shotwell	173
6.6	digiKam	175
6.7	GIMP	177
6.8	RawTherapee und Darktable	180
6.9	draw.io	180
6.10	Audio-Player	182
6.11	VLC	184

6.12	Audio- und Video-Tools	185
6.13	Etcher	189
7	Raspberry Pi	191
7.1	Grundlagen	192
7.2	Raspberry Pi OS installieren und konfigurieren	195
7.3	Der PIXEL-Desktop	202
7.4	Hardware-Basteleien	207
7.5	Kamera	217
7.6	SSD	219
7.7	Interna	222

TEIL III Linux-Grundlagen

8	Arbeiten im Terminal	229
8.1	Textkonsolen	230
8.2	Terminal	231
8.3	Textdateien anzeigen	234
8.4	Texteditoren	235
8.5	Hilfetexte und Online-Dokumentation lesen	238
9	bash	241
9.1	Was ist eine Shell?	241
9.2	Konfiguration	243
9.3	Kommandoeingabe	246
9.4	Ein- und Ausgabeumleitung	251
9.5	Kommandos ausführen	254
9.6	Globber, Substitution und Expansion	256
9.7	Variablen	263
9.8	bash-Scripts	266
9.9	Grundregeln für bash-Scripts	272
9.10	Variablen in bash-Scripts	274
9.11	Verzweigungen, Schleifen und Funktionen	279
9.12	Referenz wichtiger bash-Sonderzeichen	286

10	zsh	289
10.1	Installation und Konfiguration	290
10.2	Anwendung	295
10.3	Oh my zsh!	298
11	fish	301
11.1	Installation und erste Schritte	301
11.2	Konfiguration	305
11.3	Interna und Programmierung	308
12	Dateien und Verzeichnisse	311
12.1	Umgang mit Dateien und Verzeichnissen	311
12.2	Links	322
12.3	Dateien suchen (find, grep, locate)	325
12.4	Mehr Komfort mit modernen Kommandos	331
12.5	Zugriffsrechte, Benutzer und Gruppenzugehörigkeit	334
12.6	Spezialbits und die umask-Einstellung	341
12.7	Access Control Lists und Extended Attributes	347
12.8	Die Linux-Verzeichnisstruktur	351
12.9	Device-Dateien	354
13	Prozessverwaltung	357
13.1	Prozesse starten, verwalten und stoppen	357
13.2	Prozesse unter einer anderen Identität ausführen (su)	364
13.3	sudo	366
13.4	Polkit	372
13.5	Systemprozesse (Dämonen)	375
13.6	Prozesse automatisch starten (Cron)	378
13.7	Prozesse automatisch starten (systemd-Timer)	382
14	Konverter für Grafik, Text und Multimedia	387
14.1	Grafik-Konverter	387
14.2	Audio- und Video-Konverter	389

14.3	Dokumentkonverter (PostScript, PDF, HTML, LaTeX)	390
14.4	Markdown und Pandoc	395
15	Netzwerk-Tools	399
15.1	Netzwerkstatus ermitteln	399
15.2	Auf anderen Rechnern arbeiten (SSH)	404
15.3	Dateien übertragen (wget, curl, ftp)	411
15.4	Lynx	416
15.5	Mutt	417
 TEIL IV Text- und Code-Editoren		
16	Visual Studio Code	423
16.1	Installation und erste Schritte	424
16.2	Konfiguration	427
16.3	Git-Funktionen	430
16.4	Remote-SSH-Erweiterung	431
17	Vim	435
17.1	Schnelleinstieg	437
17.2	Text bearbeiten	440
17.3	Suchen und Ersetzen	443
17.4	Mehrere Dateien gleichzeitig bearbeiten	445
17.5	Interna	447
17.6	Tipps und Tricks	448
18	Emacs	451
18.1	Schnelleinstieg	451
18.2	Text bearbeiten	454
18.3	Suchen und Ersetzen	458
18.4	Puffer und Fenster	462
18.5	Bearbeitungsmodi	463
18.6	Konfiguration	465

TEIL V Systemkonfiguration und Administration

19	Basiskonfiguration	471
19.1	Einführung	471
19.2	Konfiguration der Textkonsolen	475
19.3	Datum und Uhrzeit	477
19.4	Datum und Uhrzeit via NTP synchronisieren	479
19.5	Benutzer und Gruppen, Passwörter	481
19.6	PAM	493
19.7	Spracheinstellung, Internationalisierung, Unicode	497
19.8	Hardware-Referenz	502
19.9	CPU-Tuning	512
19.10	Notebook-Optimierung	517
19.11	Drucksystem (CUPS)	523
19.12	Syslog	531
19.13	Journal	539
19.14	Cockpit	543
20	Netzwerkkonfiguration	547
20.1	Der NetworkManager	547
20.2	Grundlagen	555
20.3	Manuelle Konfiguration	563
20.4	Konfigurationsdateien	573
20.5	Distributionsspezifische Konfiguration	576
20.6	Zeroconf und Avahi	585
21	Software- und Paketverwaltung	587
21.1	Einführung	587
21.2	dnf und rpm (Fedora, RHEL)	593
21.3	zypper (SUSE)	607
21.4	apt und dpkg (Debian, Ubuntu)	608
21.5	pacman (Arch Linux)	627
21.6	PackageKit	632
21.7	Firmware-, BIOS- und EFI-Updates	633
21.8	Verwaltung von Parallelinstallationen (alternatives)	637
21.9	Flatpak und Snap	639

22	Grafiksystem	647
22.1	Grundlagen	648
22.2	Grafiktreiber	653
22.3	Den Status des Grafiksystems feststellen	661
22.4	Start des Grafiksystems	666
22.5	Dynamische Konfigurationsänderungen mit RandR	670
23	Administration des Dateisystems	673
23.1	Wie alles zusammenhängt	675
23.2	USB-Datenträger formatieren und nutzen	676
23.3	Device-Namen	680
23.4	Partitionierung der Festplatte oder SSD	685
23.5	Das parted-Kommando	688
23.6	Partitionierungswerkzeuge mit grafischer Benutzeroberfläche	694
23.7	Dateisystemtypen	696
23.8	mount und /etc/fstab	701
23.9	systemd versus /etc/fstab	710
23.10	Das ext-Dateisystem (ext2, ext3, ext4)	713
23.11	Das btrfs-Dateisystem	718
23.12	Das xfs-Dateisystem	734
23.13	Windows-Dateisysteme (VFAT, exFAT und NTFS)	736
23.14	Swap-Partitionen und -Dateien	740
23.15	RAID	744
23.16	Logical Volume Manager (LVM)	755
23.17	SMART	760
23.18	SSD-TRIM	766
23.19	Verschlüsselung	767
24	GRUB	779
24.1	GRUB-Grundlagen	779
24.2	Initrd-Dateien	785
24.3	GRUB-Bedienung (Anwendersicht)	789
24.4	GRUB-Konfiguration	791
24.5	Manuelle GRUB-Installation und Erste Hilfe	795
24.6	systemd-boot	799
24.7	Limine	802

25	systemd	805
25.1	Grundlagen	805
25.2	Eigene systemd-Services	815
25.3	Distributionsspezifische Details beim Systemstart	819
25.4	shutdown, reboot und halt	821
25.5	Das traditionelle Init-V-System	823
26	Kernel und Module	827
26.1	Kernelmodule	828
26.2	Device Trees	834
26.3	Kernelmodule selbst kompilieren	837
26.4	Kernel selbst konfigurieren und kompilieren	840
26.5	Kernel-Live-Patches	851
26.6	Die Verzeichnisse /proc und /sys	854
26.7	Kernel-Boot-Optionen	856
26.8	Kernelparameter verändern	860
26.9	Spectre und Meltdown	860

TEIL VI Server-Konfiguration

27	Server-Installation	867
27.1	Grundlagen	867
27.2	Red Hat Enterprise Linux	875
27.3	Ubuntu Server	883
27.4	Debian-Server-Installation	886
27.5	Elastic Compute Cloud	888
27.6	Hetzner Cloud Hosting	900
28	Secure Shell (SSH)	905
28.1	Installation	905
28.2	Konfiguration und Absicherung	906
28.3	Fail2Ban	909
28.4	Authentifizierung mit Schlüsseln	911
28.5	Zwei-Faktor-Authentifizierung	916
28.6	Zusatzwerkzeuge	922

29	Apache	925
29.1	Apache	926
29.2	Apache-Konfiguration	928
29.3	Verschlüsselte Verbindungen (HTTPS)	932
29.4	Let's Encrypt	939
29.5	Webverzeichnisse einrichten und absichern	947
29.6	Virtuelle Hosts	955
29.7	Webzugriffsstatistiken	958
29.8	PHP	962
29.9	nginx	965
30	MySQL und MariaDB	969
30.1	Installation und Inbetriebnahme	970
30.2	Administrationswerkzeuge	979
30.3	Backups	983
30.4	WordPress installieren	986
31	Postfix und Dovecot	991
31.1	Einführung und Grundlagen	991
31.2	Postfix (MTA)	1003
31.3	Postfix-Verschlüsselung (TLS/STARTTLS)	1012
31.4	Postfix-Konten	1018
31.5	Dovecot (IMAP-Server)	1029
31.6	Client-Konfiguration	1037
31.7	SpamAssassin	1038
31.8	ClamAV (Virenabwehr)	1045
31.9	SPF, DKIM und DMARC	1048
31.10	Konfigurationstest und Fehlersuche	1058
32	Nextcloud	1061
32.1	Installation	1062
32.2	Konfiguration	1068
32.3	Wartung	1072
32.4	Betrieb	1074
32.5	Kontakte und Termine	1076
32.6	Videokonferenzen (Talk)	1079

33	Samba	1081
33.1	Grundlagen und Glossar	1083
33.2	Basiskonfiguration und Inbetriebnahme	1086
33.3	Passwortverwaltung	1093
33.4	Netzwerkverzeichnisse	1100
33.5	Beispiel – Home- und Medien-Server	1105
33.6	Beispiel – Firmen-Server	1109
33.7	Linux-Client-Konfiguration	1112
33.8	Windows-Client-Konfiguration	1120

TEIL VII Sicherheit

34	Backup und Synchronisation	1125
34.1	Déjà Dup	1126
34.2	Back In Time	1128
34.3	Grsync	1131
34.4	Syncthing	1133
34.5	Inkrementelle Backup-Tools (rdiff-backup, rsnapshot, Borg Backup)	1138
34.6	Dateien komprimieren und archivieren	1147
34.7	Verzeichnisse synchronisieren (rsync)	1150
34.8	Backup-Scripts	1154
34.9	Backups auf S3-Speicher	1157
35	Firewalls	1163
35.1	Netzwerkgrundlagen und -analyse	1163
35.2	Basisabsicherung von Netzwerkdiensten	1169
35.3	Firewall-Grundlagen	1173
35.4	Firewall-Konfigurationshilfen	1174
35.5	Firewall mit nft selbst gebaut	1181
35.6	Geo-Blocking	1193
36	SELinux und AppArmor	1199
36.1	SELinux	1199
36.2	AppArmor	1207

37	Monitoring mit Prometheus und Grafana	1215
37.1	Monitoring-Grundlagen	1216
37.2	Setup-Überblick	1219
37.3	Den Node Exporter auf dem zu überwachenden Server installieren	1222
37.4	Docker-Setup für Traefik, Grafana und Prometheus	1225
37.5	Prometheus-Weboberfläche	1233
37.6	Grafana-Weboberfläche	1236
37.7	Den Node Exporter absichern	1238
37.8	Den Monitoring-Host überwachen	1242
37.9	Automatische Benachrichtigungen (Alerts)	1243
37.10	Monitoring für Webserver (Blackbox Exporter)	1251
37.11	Monitoring für MariaDB/MySQL	1255

TEIL VIII Virtualisierung, Container und Co.

38	VirtualBox	1261
38.1	VirtualBox installieren	1262
38.2	VirtualBox-Maschinen einrichten	1267
38.3	Arbeitstechniken und Konfigurationstipps	1273
39	QEMU/KVM	1279
39.1	Grundlagen	1280
39.2	Der Virtual Machine Manager	1289
39.3	libvirt-Kommandos	1296
39.4	Integration in das lokale Netzwerk (Netzwerkbrücke)	1302
39.5	Manipulation von Image-Dateien	1306
40	Docker und Podman	1313
40.1	Grundlagen und Nomenklatur	1315
40.2	Installation	1319
40.3	Docker oder Podman kennenlernen	1325
40.4	Container-Administration	1339
40.5	Eigene Images erzeugen (Dockerfile)	1346
40.6	Container-Setups mit compose	1354

40.7	Docker-Interna	1357
40.8	Podman-Interna	1361
41	Windows Subsystem for Linux (WSL)	1365
41.1	WSL ausprobieren	1366
41.2	WSL-Netzwerkanbindung	1371
41.3	Das Kommando wsl und WSL-Konfiguration	1373
42	KI-Sprachmodelle ausführen	1377
42.1	Grundlagen von Sprachmodellen	1378
42.2	GPT4All	1379
42.3	Ollama	1381
42.4	llama.cpp	1392
	Index	1401

Index

"	bash-Zeichenketten	262	/apparmor	1209
'	bash-Zeichenketten	262	/apparmor.d	1208
(())	arithmetische Ausdrücke	260	/apt/apt.conf	612
*	bash-Globbing	258, 297, 318	/apt/sources.list	612
~	Heimatverzeichnis	312	/bashrc	244
&	Hintergrundprozesse	358	/boot/grub.cfg	791
< >	Ein- und Ausgabeumleitung	251	/chrony.conf	481
	Pipes	253	/cockpit	545
\$	bash-Variablen	263	/containers	1361
\$()	Kommandosubstitution	260	/cron.daily	380
			/cron.hourly	380
			/cron.monthly	380
			/cron.weekly	380
			/crontab	378
			/crypttab	771
			/cups	526
			/default/grub	793
			/default/language	497
			/deluser.conf	483
			/dnf/dnf.conf	595
			/dovecot	1030
			/dracut.conf	787
			/file	321
			/firewall.d	1175
			/fstab	705, 706, 711, 767, 1118
			/group	485
			/gshadow	491
			/hostname	575
			/hosts	573
			/hosts.allow	1170
			/hosts.deny	1170
			/inittab	823
			/inputrc	244
			/libvirt	1285
			/lightdm	669
			/locale.gen	501
			/localtime	478
			/login.defs	346, 489
			/logrotate.conf	536
			/machine-id	1300
			/manpath.config	239
			/mdadm/mdadm.conf	747
			/mkinitcpio.conf	788
			/modprobe.conf	564, 831
			/modprobe.d	831
			/modules	832
			/modules-load.d	831
1-Wire-Thermometer	215			
2FA (Zwei-Faktor-Authentifizierung)	916			
389-Directory-Server	472			
7zr	1148			
.cache-Verzeichnis	143			
.config-Verzeichnis	143			
.emacs-Datei	466			
.htaccess-Datei	955			
.local-Verzeichnis	143			
/bin	351			
/boot	351			
/efi	43, 780			
/grub	790			
/initrd	781, 785			
/vmlinuz	781, 850			
/dev	351			
/disk	684			
/mapper	758			
/md	747			
/mmcblk	681			
/nvme	681			
/pts	699			
/sda, /sdb	681			
/vda, /vdb	681			
/zram	742			
Interna	354			
/etc	351, 474			
/Mutttrc	419			
/NetworkManager/system-connections	576			
/PackageKit	632			
/X11/xorg.conf	670			
/adduser.conf	483			
/aliases	995, 1006, 1021			
/alsa	510			
/alternatives	637			

/mtab	702	/opt	351
/my.cnf	972	/proc	351, 699, 854
/network/interfaces	579	/asound	510
/nsswitch.conf	492	/config.gz	847
/pam.conf	494	/crypto	770
/pam.d	494	/filesystems	699
/passwd	484	/mounts	702
/php.ini	963	/pci	505
/polkit-1	373	/sys	860
/postfix	1006	/registries.conf	1361
/printcap	526	/root	351
/profile	244, 264, 265	/run	699
/rc.d	824	/log/journal	539
/rc.d/rc.local	820	/sbin	351
/resolv.conf	574, 575	/init	823
/rsyslog.conf	531	/srv	
/samba/smb.conf	1087	/www	927
/sdboot-manage.conf	801	/sys	351, 855
/selinux	1205	/kernel/security	1208
/shadow	487	/tmp	
/shells	243	im Arbeitsspeicher	712
/skel	485	/usr	351
/smartd.conf	765	/bin	351
/ssh	906	/lib	351
/ssl/certs	549	/local	351
/subgid	1362	/sbin	351
/subuid	1362	/var	351, 699
/sudoers	366	/lib/docker	1358
sysconfig		/lib/dpkg/alternatives	638
network-scripts	576	/lib/rpm/alternatives	638
/sysconfig/i18n	497	/log/Xorg.O.log	665
/sysconfig/language	497	/log/journal	539
/sysconfig/network	575	/run	361
/sysctl.conf	860	/spool/cron/tabs	378
/systemd/journald.conf	541	/www	927
/systemd/network	584		
/timezone	479		
/udev	355		
/updatedb.conf	326		
/vconsole.conf	476		
/wpa_supplicant	571		
/wsl.conf	1369, 1374		
/xdg/user-dirs.conf	143		
/yum.conf	595		
/zsh	291		
/home	351		
/lib	351		
/firmware	569		
/modules	828, 829, 850		
/modules.dep	832		
/lostfound	351, 715		
/media	351		
/mnt	351		

A

A-Eintrag (DNS)	998
A/B-Bootprozess	592
a2disconf	930
a2dismod	929
a2dissite	957
a2enconf	930
a2enmod	929
a2ensite	930, 957
aa-complain	1210
aa-enforce	1210
aa-status	1209
AAAA-Eintrag (DNS)	998
abbr (fish-Kommando)	307
Abkürzungen	250
fish	307

Access Control Lists	347	SELinux	928
Access Time	717	Unicode	931
Access-Point	549	Verzeichnis absichern	953
ACL	347	virtuelle Hosts	955
acme.sh	940	apfs-Dateisystem (Apple)	698
ACPI	504	APIC	859
<i>Kerneloptionen</i>	859	aplay	510
Active Directories	1085	App-Zentrum (Ubuntu)	589
Adaptable Linux Platform (ALP)	39	AppArmor	1199, 1207
addgroup	483	apparmor-utils	1210
addr		AppImage	189, 592
<i>addr</i>	564	AppIndicator (Gnome Extension)	134
<i>link</i>	564	Apple	
adduser	483	CUPS	523
Administration	471	Dateisystem	698
Administrator-Account	61	zsh	289
Akku-Lebensdauer optimieren	521	Applets	
akms	840	KDE	149
Aktion (Syslog)	533	AppStream	592, 598, 876
Aktivitäten (Gnome)	107	apt	608
Alert Manager (Prometheus)	1244	<i>apt-cache</i>	623
Alerts (Prometheus)	1243	<i>apt-config</i>	610
alias	250	<i>apt-daily-upgrade</i>	620
<i>alias_database</i>	1022	<i>apt-get</i>	611
<i>alias_maps</i>	1022	<i>apt-key</i>	613
<i>aliases-Datei (Postfix)</i>	1006	<i>apt-mark</i>	624
Apache	949	<i>automatische Updates</i>	620
E-Mail	1021	aptitude	611
fish	307	Arbeitsfläche (Desktop-System)	111, 152
<i>modprobe.conf</i>	833	Arch Linux	31, 72
Allow	951	<i>Paketverwaltung</i>	627
allow-hotplug	579	<i>systemd-boot</i>	800
Allowed-Origins	620	arch-chroot	797
AlmaLinux	31, 878	Archivieren von Dateien	1147
Alpine Linux	1331	Gnome	120
alsactl	510	arecord	510
alsamixer	510	arithmetische Ausdrücke (bash)	260
alternatives	637	ARM-CPU's	28
Amazon Linux	891, 899	arptables	1183
Amazon Web Services (AWS)	888, 1157	Asahi (Fedora-Variante)	83
amdgpu (Grafiktreiber)	655	Asahi Linux	28
<i>amdgpu-pro</i>	655	ASCII	497
<i>amdgpu_top</i>	664	Asymmetrische Verschlüsselung	933
Ollama	1383	atime (mount-Option)	717
amdtm (Kerneloption)	1385, 1396	Atomic Distribution	39
anacron	382	Atomic Linux	592
Android	27, 31	Audacious (Audio-Player)	183
Apache	925	Audio	
Authentifizierung	953	ALSA	510
FPM/FastCGI	963	Dateien recodieren	187
HTTPS	932	Konverter	389
Monitoring	1251	audiofile	389
Passwort	953	audit.log (SELinux)	1206

- aufs-Dateisystem 700
- AUR-Pakete 630
- Ausgabeumleitung 251
 - sudo* 370
- Auslagerungsdatei 58, 740
- authconfig 496
- AuthConfig 950
- Authentifizierung
 - Apache* 953
 - authselect* 493, 496
 - IMAP* 1035
 - SMTP* 1036
- auto (fstab-Option) 699
- Auto-Login 669
- autocd 293
- autofs 700
- automatic.conf 600
- Automatische Ausführung von Jobs 378, 382
- automount 700
- Autostart 144, 145, 156, 1135
- Avahi 585
 - avahi-browse* 586
 - avahi-daemon* 586
 - avahi-discover* 586
 - avahi-dnsmconfd* 586
 - Gnome-Freigaben* 137
- avconv 170
- AWStats 958
- B**

- Backbox Exporter 1251
- Background-Prozesse 358
- backintime 1128
- Backup Domain Controller (BDC) 1084
- Backups 1125
 - Emacs* 452
 - inkrementelle* 1138
 - KVM* 1156
 - LVM-Snapshots* 1155
 - MySQL* 983
 - Script* 270
- baobab 120
- Base Images (Docker) 1326
- bash 241, 385
 - bash_history* 246
 - bashrc* 87, 244
 - completion* 248
 - Konfiguration* 243
 - Programmierung* 266
 - Shell-Arten* 245
 - Tastenkürzel* 249
 - Variablen* 275
- Basic Authentication 953
- bat 331
- Batterie (Notebooks) 504
- Bedingungen (bash) 280
- Benutzer 483
 - einrichten* 482
 - Gruppen* 335
 - verwalten* 481
- Berkeley Database 1006
- Besitzer
 - neue Dateien* 345
 - von Dateien* 334
- Betriebssystem 27
- Betterbird 168
- bg 359
- Bilder-Verzeichnis 142
- Bildschirmfreigabe 139
- bind interfaces only 1091
- bind-address 973
- binwalk 788
- Binärpaket 605
- BIOS 42
 - BIOS-GRUB-Partition* 784
 - BIOS-RAID* 744
 - Systemstart* 783
 - Updates* 633
- BitLocker 736
- Blacklist (E-Mail) 997
- blacklist (modprobe.conf) 833
- Blender 188
- blkid 706, 711
- Bluetooth 506
 - bluetoothctl* 506
 - bluetoothd* 506
- BMP-PS-Konverter 388
- bmp2eps 388
- boltctl 509
- Bonjour 585
- Bookworm 75, 615
- Boot-Optionen 856
- Boot-Partition 57
- Boot-Probleme 64
- Boot-Prozess
 - Bootloader* 783
 - System-V-Init* 823
- bootctl 801
- Bootloader 791
 - systemd-boot* 799
- Borg Backup 1143
- Boxes 1279
- Bridge 1302
- bridge-utils 1302
- Bridged Networking 1274

browseable	1101
Browsing (Samba)	1083
BSD-Lizenz	35
btop	664
btrfs-Dateisystem	697, 718, 719
<i>Btrfs Assistant</i>	734
RAID	726
<i>Snapper</i>	733
<i>Swap-Dateien</i>	742
Buckets (S3)	1158
Budgie	90
build-Kommando (Docker)	1350
Bullseye	75, 615
bunzip2	1147
Buster	75, 615
bzip2	1147, 1148

C

CachyOS	31, 72
<i>Firewall</i>	1179
<i>Installation</i>	96
<i>sdboot-manage</i>	801
<i>systemd-boot</i>	800
Calamares Installer	79, 96
CalDAV	1077
Camera Serial Interface (CSI)	217
Cannot change locale (Fehlermeldung)	501
Canonical	33, 89
canonical-livepatch	853
canonical_maps	1025
Capabilities	351
CardDAV	1077
Carriage Return	391
case	282
cat	234
cat-config (systemd-analyze)	542
ccat	300
cd (zsh)	293
cdh (fish-Kommando)	304
Celeste	1153
CentOS	31, 878
<i>Stream</i>	878
certbot	940
cgroups	699
<i>Docker</i>	1359
chage	488
character set	497
chattr	723
chcon	1202
checkarray	748
chpasswd	490
Chrome	163
Chrome OS	32
chrome-gnome-shell	133
Chromium	164
Chrony	481
chroot	796, 1172
chsh	243, 290
cifs	699, 1116
cifs-utils	1116
Cinnamon	648
ClamAV	1045
clamav-milter	1046
classes.conf	526
Clear Linux (/etc/fstab)	712
cless	300
cloneconfig	847
Cloud-Server-Installation	868
cloud-guest-utils	896
cloud-utils-growpart	896
Cluster-Dateisysteme	674, 700
CMD (Dockerfile)	1348
cmdline-Datei	855
cmus (Audio-Player)	183
Cockpit	472, 543
<i>cockpit-machines</i>	1289
<i>cockpit.conf</i>	545
Code (Editor)	423
<i>Ollama und Continue</i>	1391
Codec	170
CodeReady Linux Builder	602
colored-man-pages	300
colorize	300
commit (btrfs-Option)	731
complete	248
Completion (Shell)	248, 302
compose.yaml (Datei)	1354
<i>Prometheus/Grafana-Beispiel</i>	1225
Compositor (Wayland)	649
compress (btrfs-Option)	731
compsize	725
compsys	292
config.fish (Datei)	306
config.txt (Device Trees)	835
console-setup	475, 477
Container	1313
Container Layer (Docker)	1357
Continue (VSCode-Plug-in)	1391
Contrib-Pakete	614
Control Groups	699, 805
<i>Docker</i>	1359
Copr (DNF)	603
Copy on Write (COW)	718
<i>deaktivieren</i>	723
CoreOS (Fedora)	83

- COSMIC (Desktop) 72, 648
- count (fish-Kommando) 308
- cp 313
 - Namen beim Kopieren ändern* 320
- cPanel 472
- cpio 789
- CPU
 - aktuelle Taktfrequenz ermitteln* 513
 - cpu-checker* 1281
 - CPU-Stresstest* 1247
 - cpupower* 513
 - Governor* 514
 - Temperatur* 515
 - Tuning* 512
 - Turbo Boost* 514
- cracklib 489
- cramfs-Dateisystem 700
- CRB-Repository 602
- create mask 1102
- Cron 378
 - crontab* 378
 - durch systemd ersetzen* 382
 - WSL* 1369
- CrowdSec 909
- Crypto-Dateisystem 674
- cryptsetup 769
- crypttab (Datei) 771
- csh 242
- ctrl-alt-del.target 810
- CUA-Modus (Emacs) 455
- CUDA-Bibliothek 651, 1381
- CUPS 523
 - Interna* 526
 - cupsd.conf* 526
- curl 412, 1157
- custom.conf 669
- D**

- Dämonen 375
- daemon.conf 669
- dash (Shell) 243, 267
- Dash (Gnome) 109
 - Dash to Dock (Gnome Extension)* 135
- Dateien 311
 - Dateinamen* 311
 - drucken* 530
 - Grundlagen* 311
 - Jokerzeichen* 258
 - komprimieren* 1147
 - kopieren mit sed* 320
 - suchen* 325, 326
 - versteckte Dateien* 312
- Dateimanager 114
- Dateisystem
 - ext2/3/4* 707, 713
 - Konfiguration* 705
 - Loopback-Device* 700
 - Typen* 696
 - vergrößern (ext3)* 716
 - vergrößern (xfs)* 735
 - verschlüsseln* 674, 767
 - verwalten* 673
 - virtuelles* 699
 - WSL* 1368
- Dateityp
 - im ls-Kommando* 314
 - Magic-Datei* 321
- Dateiverwaltung
 - Grundlagen* 311
- Datenbank-Server 969
- Datenpartition 57
- Datenträger formatieren 676
- DAV (Gnome) 137
- DaVinci Resolve 188
- dbus-broker 509
- dbus-daemon 509
- dcfldd 225
- dconf-Datenbank 141
- dcraw 389
- dctrl-Format 624
- dctrl-tools 624
- dead keys 475
- deb822-Format 613
- Debian 32
 - DKMS* 838
 - Firmware* 76
 - initrd* 786
 - Installation* 74
 - Server-Installation* 886
 - statische Netzwerkkonfiguration* 579
 - Tastatur* 475
 - VirtualBox* 1271
- declare 265
- Decoder 169
- degraded (btrfs-Option) 727
- Déjà Dup 1126
- delgroup 483
- deluser 483
- Deny 951
- Desktop-System 73, 105, 147
- desktop-Datei (Syncthing) 1135
- Device Trees 216, 834
 - device-tree-Parameter* 835
- DeviceKit 508
- Devices 337, 354, 680

<i>Interna</i>	354	do-release-upgrade	622
<i>Kernelmodule</i>	833	Dock (Gnome)	109
<i>udev</i>	355	Docker	1313
devtmpfs	699	<i>docker compose</i>	1354
df	701	<i>Dockerfile-Syntax</i>	1347
DGX Spark	1379	<i>Desktop</i>	1318, 1322
dhclient	567	<i>Engine</i>	1320
DHCP	558	<i>Hub</i>	1316
<i>dhcpd</i>	567	<i>Installation</i>	1320
<i>Client-Konfiguration</i>	558	<i>Prometheus/Grafana-Beispiel</i>	1225
digiKam	175	<i>Rootless</i>	1321
directory mask	1102	<i>Universal Base Images (UBI)</i>	1331
Directory Server	472	<i>Volumes</i>	1333
dirh (fish-Kommando)	304	DocumentRoot (Apache)	927
dis-icode-ld (Kerndoption)	858	Dokument-Konverter	390
dis-ucode-ldr (Boot-Parameter)	636	Dokumente-Verzeichnis	142
disable_vrfy_command (Postfix)	1029	Dolphin	152
discard	708, 767	Domain-Level-Sicherheit	1084
<i>LUKS-Verschlüsselung</i>	771	DomainKeys Identified Mail	1050
discard (btrfs-Option)	731	Domainname	556
Discover	159, 632	Doppellizenzen	35
<i>Firmware-Updates</i>	635	DOS-Dateien konvertieren	391
Discoverable Partitions Specification	711	dotglob	258
Disk-Images (libvirt/KVM)	1287	Dovecot	
Disk-Quotas	674	<i>dovecot.conf</i>	1030
dislocker	736	<i>IMAP-Authentifizierung</i>	1035
Dispatcher (NetworkManager)	555	<i>IPv6</i>	1034
Display-Manager	667	<i>SMTP-Authentifizierung</i>	1036
Distributionen		Downloads-Verzeichnis	142
<i>Linux</i>	29	dpkg	624
<i>Updates</i>	65	<i>dpkg-reconfigure</i>	479, 626
<i>Überblick</i>	31	<i>Statuscode</i>	625
DKIM	1050	dracut	787
DKMS	838	Dragon	185
<i>VirtualBox</i>	1263	draw.io	180
DLNA (Gnome-Freigabe)	138	DRI	652
dm_crypt	767	Dritte Maustaste	113
DMARC	1057	DRM	170, 649
dmesg	535	Drucken	523
dnf	593	<i>automatische Datenkonversion</i>	525
<i>automatische Updates</i>	600	<i>Drucker-Server</i>	523
<i>dnf-automatic</i>	600	<i>Druckjobs verwalten</i>	530
<i>dnf-makecache</i>		<i>Dämon (CUPS)</i>	526
<i>dnf-plugin-system-upgrade</i>	601	<i>Filter</i>	525
<i>dnf-utils</i>	600	<i>Gnome</i>	123
<i>dnf.conf</i>	595	<i>KDE</i>	158
<i>systemd-Timer-Interna</i>	383	<i>Konfiguration</i>	523
DNS		<i>MIME (CUPS)</i>	527
<i>Client-Konfiguration</i>	558, 574	<i>per Kommando</i>	530
<i>DNS-Resolver</i>	575, 1281	<i>PostScript</i>	524
<i>dnsmasq</i>	1281	<i>Spooling-System</i>	524
<i>Mail-Server</i>	998	<i>Warteschlange</i>	524
<i>Reverse DNS</i>	1001	DS1820	215

DTB-Dateien 835
dtoverlay (Schlüsselwort) 835
dtparam (Schlüsselwort) 835
duf 331
Duke 615
duplicity 1146
DVD-Ripper 187

E

E-Mail

Alias 1021
 Blacklist 997
 DNS 998
 Grundlagen 991
 mutt 417
 Relaying 997
 Server 991
 Thunderbird 165
 Viren 1045
e2label 716
ebtables 1183, 1306
ECDSA/ED25519 (SSH-Schlüssel) 912
EDITOR 238
Editoren 235
 Emacs 451
 Nano 235
 Vim 435
 VSCode 423, 424
edk2-ovmf 1294
EEPROM-Update 203
EFI 42
 efibootmgr 797, 874
 EFI-Partition 692, 738
 GPT 780
 GRUB-Reparatur 795
 in KVM/QEMU 1294
 Partition 43, 780
 Secure Boot 44, 64, 782
 Server-Installation 872, 873
 System Partition 43, 799
 systemd-boot 799
 Systemstart 780
 Updates 633
 Verfügbarkeit testen 872
EGL 649, 652
Eingabeumleitung 251
 sudo 370
Elastic-Cloud-Dienst (EC2) 888
 Elastic Block Store (EBS) 892
 Elastic IP (AWS EC2) 898
Electronic Frontier Foundation 940
Elementary OS 90

Elevate (RHEL-Klon-Updates) 601
ELinks 416
Elisa (Audio-Player) 182
Elvis 237
Emacs 451
 .emacs-Datei 465, 466
 automatische Sicherheitskopie 452
 Bearbeitungsmodi 463
 Cursorbewegung 453
 dynamische Abkürzungen 458
 Erweiterungen 468
 Fenster 462
 Fließtext 457
 Konfiguration 465
 MELPA 468
 Puffer 462
 reguläre Ausdrücke 460
 Schnelleinstieg 236
 Schriftart einstellen 465
 suchen 458
 suchen und ersetzen 461
 Tabulatoren 456
emergency (Kerneloption) 857
Emergency-Target 810
EnableInsecureGuestLogin (Samba) 1122
Encoder 169
Encryption (Dateisystem) 674
EndeavourOS 31
Energiesparfunktionen 504
ENI-Konfiguration 579
ENTRYPOINT (Dockerfile) 1348
Environment-Variablen 264
EPEL-Paketquelle 602
Epiphany 164
epsffit 392
erd 331
Erweiterte Partition 52
ESP (EFI System Partition) 43, 799
 esp-Flag (parted) 738
ESR-Version (Firefox) 162
Etcher 46, 189
etckeeper 474
Ethernet-Controller
 IP-Adresse 559
 konfigurieren 563
 MAC-Adresse 557
ethtool 565
Evolution 168
EWS (Exchange Server) 994
exa/eza 332
Exchange-Server 994
Exec Shield 1200
ExecCGI 950

ExecStart-Schlüsselwort (systemd)	816	Filter	
exFAT-Dateisystem	736	CUPS	525
exfat-utils	677, 737	IP-Paketfilter	1174
EXIF-Informationen	389	find	326
exiftool	389	findmnt	702
exit (bash)	273	Firewall	1163
Expansionsmechanismen (bash)	256	AWS EC2	899
Expansion von Dateinamen	247	Beispiel	1191
expect	490	Docker	1321
Experimental-Pakete	615	firewall-cmd	1005, 1177
export	265	firewalld	900, 1175, 1241
Exporter für Prometheus	1216	Geo-Blocking	1193
ext4-Dateisystem	697, 713	libvirt	1282
Extended Attributes	347	Mail-Server	1005
extendedglob	293	Paketfilter	1174
Extension Pack (VirtualBox)	1264	Prometheus Node Exporter	1241
extractres	392	Samba	1090
		Web-Server	926
F		Firmware	569
faac/faad	389	Debian	76
FAI (Fully Automatic Installation)	472	Updates	633
Fake-RAID	744	fish (KDE)	155
Fallout	860	fish (Shell)	301
Farbprofile	125	fish-config	305
FastCGI Process Manager	963	fish_add_path	307
FAT-Dateisystem	698, 736	fish_greeting	306
fc-list	129	fish_update_completions	302
fd	331	flac	390
Fedora	32	Flat Printed Circuit (FPC)	217
Distributions-Update	601	Flatpak	641
DKMS	838	FollowSymLinks	950
dracut	787	Fonts	128
Firewall	1175	for	282
initrd	787	force group (Samba)	1102
Installation	83	Foreshadow	860
Silverblue	39	Fork-Typ (systemd)	816
statische Netzwerkkonfiguration	576	Forky	615
Tastatur	476	FORMAT (Windows)	50
Fensterbuttons (Gnome)	130	Formatieren	
Fernanmeldung	140	btrfs-Dateisystem	719
Fernwartung	139	exfat-Dateisystem	676
Feste Links	322	ext3/ext4-Dateisystem	714
Festplatte		FAT-Dateisystem	676
formatieren	50	ntfs-Dateisystem	676
partitionieren, Linux	56	Windows-Dateisystem	737
überwachen (SMART)	760	xfs-Dateisystem	735
FFmpeg	170, 390	Forward Proxy	1220
fglrx (Grafiktreiber)	655	FPM/FastCGI-Verfahren	963
FIFO	253	Fraktionale Skalierung	125
file (Kommando)	321, 789	free	504
FileInfo	950	Free Software Foundation	34
Filesystem Hierarchy Standard (FHS)	351	Freigaben	
		Medien (Gnome)	138

WebDAV (Gnome)	137
Samba	1100
freshclam	1046
fsck.ext4	715
fsck.xfs	735
FSF	34
fstab (Datei)	705
CIFS	1118
fstrim	767
LUKS-Verschlüsselung	771
FTP	413
Client	414
ftp-Kommando	414
Secure FTP Server	906
Server (sftp)	906
function (bash)	285
FUSE	700
fuser	361
Fusion-Paketquellen (Fedora)	87
Fuzzy Finder	332
fwupd/fwupdmgr	634
fx	331
fzf	331, 332

G

GART-Speicher (AMD-GPU)	1384, 1396
Gastlogin (Windows)	1121
Gateway	557
Client-Konfiguration	566, 574
GB10-Chip	1379
gdisk	711, 754
gdm	667
Geary	168
Geo-Blocking	1193
getafm	392
getcap	351
getfacl	348
getfattr	350
getsebool	1204
getsll	940
gfs-Dateisystem	674, 700
GGUF-Dateien	1395
Ghostsript	392
GID	485
GIMP	177
GL (Open GL)	652
glibc (Zeitzone)	478
Global Filesystem	674
Global Unicast (IPv6)	560
Globbering	
bash	256
fish	303
globstar	259
zsh	296
glow	331
GLX	652
glx-utils	663
glxinfo	663
GMT (Greenwich Mean Time)	477
Gnome	105
Display-Manager	667
Extensions	132
gnome-browser-connector	133
gnome-disk-utils	695
gnome-disks	695
gnome-extensions	133
gnome-extensions-app	132
gnome-font-viewer	129
gnome-keyring-daemon	914
gnome-language-selector	497, 619
gnome-music (Audio-Player)	183
gnome-nettool	403
gnome-remote-desktop	140
gnome-software	632, 635
gnome-system-monitor	360
gnome-terminal	231
gnome-tweaks	130
gnome-user-share	137
Proxy-Einstellungen	550
Shell Extensions	132
systemd	144
Thunderbolt	509
Tweak Tool	130
Wayland	650
GNU	37
Emacs	451
General Public License	34
GRUB	791
GoAccess	959
Google	
Google Analytics	958
Google Authenticator	916
Google Chrome	163
Nameserver	558
Governor (CPU)	514
GParted	694
gpsswd	491
gpg	1161
GPGPU	653
GPL	34
Apple	289
GPT	51, 687
BIOS-GRUB-Installation	784
EFI	780
GPT-Generated Unified Format	1395

<i>gpt-oss-Modell</i>	1397	Hardware Enablement Stack	95
<i>Partitionsnummern</i>	683	Hardware-RAID	744
GPT4All (LLMs)	1379	Haruna	185
GPU-Auslastung darstellen	664	Hashbang	267
Grafana	1215	hcloud	903
<i>Weboberfläche</i>	1236	HEIC/HEIF-Dateien	172, 389
Grafik-Konverter	387	Heimatverzeichnis	312, 484, 485
Grafiksystem	647	help	240
graphical-Target	666	Heredoc-Syntax	286
greetd	669	<i>SSH</i>	407
grep	328	Hetzner Cloud Hosting	900
<i>Beispiele</i>	266	Hibernate (Kerneloptionen)	859
<i>grep-dctrl</i>	624	hidden_host	419
<i>grepall (Script-Beispiel)</i>	266	HiDPI-Bildschirm	125, 1270
groupadd	483	Hintergrundprozesse	358
growpart	896	History (bash)	246
Grsync	1131	History (Paketinstallation)	
GRUB	791	<i>APT</i>	624
<i>Bedienung</i>	789	<i>DNF</i>	597
<i>grub.cfg</i>	791	<i>ZYpp</i>	608
<i>grub2-mkconfig</i>	793	hold-Status (dpkg-Kommando)	627
<i>grubby</i>	791, 794	Hollama	1390
<i>Kernel-Updates</i>	786	Home-Server	1105
<i>Konfiguration</i>	791	Home-Verzeichnis	312
<i>Reparatur (EFI)</i>	795	HOOKS (Arch Linux)	788
<i>Secure Boot</i>	782	host	1001
<i>Server-Installation</i>	872	Host-RAID	744
Gruppen	485	Hostname	556
<i>neue Dateien</i>	345	<i>einstellen</i>	575
<i>von Dateien</i>	334	<i>HOSTNAME-Variable</i>	265
gs	392	<i>hostnamectl</i>	814
gsettings	141	<i>Server-Installation</i>	869
gsox	390	hosts-Datei	573
GStreamer	171, 511	hosts allow	1091, 1170
GTT-Speicher (AMD-GPU)	1384, 1396	hosts deny	1091, 1170
GTUBE-Testnachricht	1042	Hotplug-System	508
gucharmap	129	Hotspot einrichten (WLAN)	549
guest account (Samba)	1104	HP-Druckertreiber	527
guest ok (Samba)	1088, 1100, 1104	HPLIP	527
guest only (Samba)	1104	hplip-gui	528
gummiboot-Projekt	799	hplip-toolbox	528
gunzip	1147, 1148	htop	360
Gutenprint	525	htpasswd	953, 1229
GVFS	119	HTTP (Webserver)	925
gvim	435	<i>HTTP-Proxy</i>	550
gzip	1147, 1148	<i>httpd</i>	925
		<i>httpd.conf</i>	928
		<i>HTTPS</i>	932
H		hwclock	477
Hacker-Kernel	842	HWE-Pakete	95
halt	821	hydra	909
Hardware	503	Hyper-Threading	859, 862
<i>Devices</i>	354		

Hyper-V 1266
Hyprland 648

I

i18n (Internationalisierung) 497
i915 (Grafiktreiber) 656
ICMP 555, 1165
Icons (Gnome) 107, 135
IdentityFile 915
idn 575
if 279
ifcfg-rh (NetworkManager-Plug-in) 554, 576
ifconfig 565
ifdown 581
IFS 269
ifupdown (NetworkManager-Plug-in) 554
Image Magick 387
Image Mode for RHEL 39, 592
Images
 Docker 1315
 Docker, Interna 1357
 Format umwandeln 1311
 Formate 1287
 lesen/manipulieren 1306
 libvirt/KVM 1285
 vergrößern (QCOW2) 1311
 VirtualBox 1268
IMAP 992, 993
 Authentifizierung 1035
Immunix 1207
Immutable Distribution 39, 592
includeres 392
Includes 950
Indexes 950
inet6 580
inet_interfaces (Postfix) 1009
info 240
init (Kerneloption) 857
Init-System 805, 823
 Init-Scripts 824
 Kernelparameter 860
Initramfs-Disk 785, 788, 858
initrd-Datei 781
 Kerneloption 858
 selbst erzeugen 785
inittab 823
Inkrementelle Backups 1138
Inkscape 182
InnoTek 1261
Inode 323
inotify-Limit (Syncthing) 1137
inputrc 244
Insecure Guest Login (Samba) 1122
install (modprobe.conf) 833
Installation 48
 Anleitungen 69
 Benutzerverwaltung 61
 externe Datenträger 47
 Grundkonfiguration 61
 Grundlagen 41
 Linux deinstallieren 67
 Netzwerkinstallation 46
 Netzwerkkonfiguration 62
 Probleme 63
 root-Passwort 61
 Software-Installations-Script 590
 Tastaturprobleme 63
 Updates 65
 Varianten 45
Intel
 CPUs 512
 intel_gpu_top 664
 Spectre und Meltdown 860
Interaktive Shell 245
Interface 556
interfaces 579, 1091
Internal Field Separator 269
Internationalisierter Domainname 575
Internationalisierung 497
Internet Printing Protocol (IPP) 528
inxi 503
ionice 364, 1156
iotop 360
ip-Kommando 400
 addr show 568
 route 566
IP-Adresse 555, 558
IP-Filter 1174
IP-Forwarding 1305
IP-Nummer 555, 558
IP-Ports 555
 Liste 1164
IPng 559
IPP 528
iptables 1174
IPv6
 AWS EC2 898
 deaktivieren 858
 Debian 580
 Dovecot 1034
 Grundlagen 559
 machine-id (Datei) 1300
 Mail-Server 998
 manuelle Konfiguration 567
 MySQL und MariaDB 973

Postfix	1010
Samba	1092
SSH-Server	908
TCP-Wrapper	1171
ISO-8859-Zeichensätze	497
ISO-Image	45
iso9660	698, 707
iw	569

J

J8-Header	208
jed	236, 451
jmacs	236, 451
Jobs regelmäßig ausführen	378, 382
joe	236, 451
Jokerzeichen (Globbing)	258, 317
<i>Komplikationen</i>	319
Journal (systemd)	531
<i>journalctl</i>	540
<i>journal.conf</i>	541
<i>WSL</i>	1369
Journaling-Dateisysteme	
<i>btrfs</i>	718
<i>ext4</i>	713
<i>xfs</i>	734
jove	236, 451

K

k10temp (Kernelmodul)	516
kacpid	376
Kalender	
<i>Lightning (Thunderbird)</i>	167
Kalender (Gnome)	123
Kali Linux	32
Kamera (Raspberry Pi)	217
kbd	476
kblockd	376
kdbus	509
KDE	147
<i>KDE Linux (Distribution)</i>	148
<i>KDE Neon (Distribution)</i>	33, 148
Kdenlive-Programm	188
kdevtmpfs	376
Kernel	27, 842
<i>Boot-Optionen</i>	790, 856
<i>Dokumentation</i>	843
<i>Device Trees</i>	834
<i>Einstellungen ändern</i>	860
<i>Hotplug-Funktion</i>	508
<i>installieren</i>	850
<i>IP-Filter</i>	1174

Kernel Mode Setting (KMS)	652
kernel.img-Datei	203
Kernelmodule	834
<i>kompilieren</i>	840, 849
<i>Konfiguration feststellen</i>	846
<i>konfigurieren</i>	846
<i>Live-Patches</i>	851
<i>Logging</i>	535
<i>Module</i>	828
<i>neueste Version</i>	844
<i>Optionen</i>	790, 833
<i>Parameter</i>	860
<i>Prozesse</i>	376
<i>tainted</i>	654
<i>Unified Kernel Image</i>	785
<i>Update (GRUB)</i>	786
keyboard-setup	476
keyfile (NetworkManager-Plug-in)	554, 576
kGraft	852
kgx	231
khelperd	376
KI-Sprachmodelle	1377
kill	363
killall	363
KMail	169
kmod	828, 831, 832
KMS	649, 652
<i>video</i>	858
Kodi	185
Kommandos	357
<i>ausführen</i>	254
<i>bedingt ausführen</i>	256
<i>Eingabe</i>	246
<i>im Hintergrund ausführen</i>	255
<i>Kommandointerpreter</i>	241
<i>regelmäßig ausführen</i>	378, 382
<i>siehe auch Prozesse</i>	357
<i>starten</i>	358
<i>Substitution (bash)</i>	260
Konfiguration	471
<i>Benutzereinrichten</i>	482
<i>Dateisystem</i>	705
<i>Kernel</i>	840, 846
<i>Netzwerk</i>	547
<i>Passwort</i>	489
<i>Prompt</i>	245
<i>Schriftart</i>	477
<i>Tastatur (Textkonsole)</i>	475
<i>Textkonsole</i>	475
<i>Zeitzone</i>	477
Konsole	231
<i>Schriftart</i>	477
<i>Tastatur</i>	475

- wechseln* 230
- Kontakte (Gnome) 123
- Konverter 387
- kpartx 1307
- kPatch 852
- Krita 179
- ksh 242
- ksoftirqd 376
- Ksplice 852
- KStatusNotifierItem (Gnome Extension) 134
- kswapd 376
- ksysguard 360
- kthread 376
- Kubuntu 33, 90
- KVM 1279, 1280, 1283
 - Backup* 1156
 - EFI* 1294
 - kvm-ok* 1281
 - SELinux* 1300
- kworker 376
- L**

- l10n (Localisation) 497
- Label
 - /etc/fstab* 706
 - root* 857
- labwc (Wayland) 202
- lame 390
- LAMP/LEMP-System 966
- Landscape (Ubuntu) 472, 589
- LANG 498, 499
- LANGUAGE 499
- language-selector-gnome 619
- lapic (Kerneloption) 859
- Large Language Model (LLM) 1377
- LaTeX 394
- Latin-Zeichensätze 497
- Laufwerke (gnome-disks) 695
- Laufwerksbuchstaben (C:, D:) 676
- LC_ALL 499
- LC_COLLATE 498
- LC_CTYPE 498
- LC_MESSAGES 498
- LC_MONETARY 498
- LC_NUMERIC 498
- LC_PAPER 498
- LC_TIME 498
- LC_TYPE 497
- ldd 493
- Leapp (RHEL-Updates) 601
- less 234
 - /proc-Dateien* 854
- let 265
- Let's Encrypt (Zertifikate) 939
 - Cockpit* 545
- lfs 331
- lftp 415
- LGPL 35
- libdbus 509
- libdrm 654
- libgudev 508
- libguestfs 1307, 1308
- libheif (Bibliothek) 172, 389
- libinput 649
- libpam-google-authenticator 918
- LibreELEC 185
- libsvg2 389
- libudisk2 508
- libvirt 1279, 1283
 - Kommandos* 1296
 - SELinux* 1300
 - SSH* 1291
- libwrap 1171
- lightdm 669, 670
- Lightning 1077
- Limine 802
- Limit 950
- Line Feed 391
- Link-Local-Adressen (IPv6) 560
- Links 322
- Linus Torvalds 38
- Linux 27
 - deinstallieren* 67
 - Distribution* 29
 - Entstehung* 34
 - Installation* 41, 48
 - Kernel kompilieren* 840
 - Kernelmodule* 828
 - Konfiguration* 471
 - Linux Mint* 71, 90
 - Linux Standard Base* 34
 - Linux Time Machine* 1153
 - Linux Unified Key Setup (LUKS)* 767
 - Linux Vendor Firmware Service* 633
 - Startprobleme* 64
 - Systemveränderungen* 65
 - Updates* 65
 - Voraussetzungen* 41
- Live-System 30, 47
- Lizenzen 34
- llama.cpp 1392
- llvmpipe 663
- lm-sensors 515
- LMDB-Dateien (Postfix) 1006
- LMTDP 1043

-
- ln 323
 - local_recipient_maps 1023
 - Locales/Internationalisierung 497
 - locale 500
 - locale-gen 501
 - localectl 476, 497, 814
 - localhost 556, 573
 - localmodconfig 848
 - locate 326
 - log file 1093
 - logger 534
 - Logging-Dateien 531
 - Apache 932
 - Docker 1335
 - Logrotate 535
 - Logwatch 537
 - MySQL 985
 - Postfix 1011
 - Samba 1093
 - X 665
 - Logical Volume Manager 54, 755
 - Login-Name 484
 - Login-Shell 245
 - login.defs 489
 - loginctl 813
 - Logische Partition 52
 - LOGNAME-Variable 265
 - logrotate 535
 - Apache 932
 - logwatch 537
 - Lokale Netze 547
 - Sicherheit 1163
 - Lokalisierung 497
 - Loopback-Device 700
 - Loopback-Interface 557
 - lpadmin 530
 - lpc 530
 - lpd 526
 - lpinfo 530
 - lpoptions 526, 530
 - lpq 530
 - lpr 530
 - lprm 530
 - lpstat 530
 - ls 314
 - lsattr 724
 - LSB 34
 - lsblk 502, 683
 - lscpu 512
 - lsd 331
 - lsmem 504
 - lsmod 830
 - lsuf 1167
 - lspci 502, 505, 662, 830
 - lsscsi 505, 684
 - lsusb 502, 505
 - LTS-Version (Ubuntu) 89
 - LTS Enablement Stack 95
 - LTS-Support-Status (Ubuntu) 617
 - Lubuntu 90
 - Lüftersteuerung 522
 - LUKS 767
 - lvcreate 758, 1155
 - lvextend 758
 - LVM 755
 - Backup mit Snapshots 1155
 - Grundlagen 54
 - RAID 756
 - Server-Konfiguration 870
 - Snapshots 759
 - lvremove 1156
 - LXDE 202
 - Lynx 416
 - lzop 1147, 1148, 1156
- ## M
-
- m-a (Kommando) 840
 - m23 472, 589
 - MAC-Adresse 400, 557, 1170, 1200
 - feststellen 568
 - ändern (Server-Virtualisierung) 1305
 - mac80211-Framework 569
 - Machine Owner Key (MOK) 850
 - NVIDIA 655
 - Machine Owner Keys (Secure Boot) 782
 - machine-id (Datei) 1300
 - macOS-Dateisystem 698
 - MacVTap-Device 1288
 - madplay 389
 - Magic-Dateien 321
 - magick 271, 387
 - Mail-Server 991
 - Fehlersuche 1059
 - IPv4 998
 - Mailbox 994, 1004
 - Dovecot 1034
 - maildir-Format 994
 - Dovecot 1034
 - Mutt 418
 - Postfix 1020
 - mailq 1012
 - Main-Pakete 614
 - main.cf-Datei (Postfix) 1005, 1007
 - Major Device Number 354
 - make-ssl-cert 938

- makepasswd 490
- makethumbs 271
- man 239
- Managed Server 869
- Mandatory Access Control 1200
- Manjaro 31
- Manuelle Netzwerkkonfiguration 576
- map to guest (Samba) 1088, 1104
- mapfile 277
- MAPI (Exchange Server) 994
- MariaDB 969
 - Docker-Container* 1333
 - Exporter (Prometheus)* 1255
 - Monitoring* 1255
- Markdown 395
 - Emacs-Erweiterung* 468
- Master Boot Record 783
- master.cf-Datei (Postfix) 1011
- math (fish-Kommando) 308
- Matomo 959
- max log size 1093
- maxcpus (Kerneloption) 859
- mb 1160
- mbox-Format 994, 1004
- MBR 687, 783
 - Partitonsnummern* 682
- md_mod (Kernel-Treiber) 747, 755
- MDA 992
- mdadm 746, 749
- mdadm.conf 747
- mdcheck 753
- mdnsd 375
- Medien-Server 1105
- Medienfreigabe 138
- medusa 909
- MELPA (Emacs-Erweiterungen) 468
- Meltdown 633, 860
- Memtest86 504
- mencoder 390
- menu.lst 790
- Mesa (Bibliothek) 649, 652
 - mesa-utils* 663
- mg 236
- MicroOS 592
- Microsoft
 - Exchange Server* 994
 - KVM-Installation* 1295
 - SMB-Protokoll* 1083
 - Subsystem for Linux* 1365
 - TrueType-Fonts* 129
 - VSCode* 423, 424
 - Windows-Partitionen* 736
- Midori 164
- Mikrocode-Update 633
 - verhindern* 858
- Milter 1046, 1056
- MIME (CUPS) 527
- Minor Device Number 354
- Mint 33
- Mirrored Mode Networking (WSL) 1372
- Mirroring 745
- MIT-Lizenz 35
- mitigations (Kerneloption) 858, 863
- mkconf 747
- mkfs.btrfs 719
- mkfs.exfat 737
- mkfs.ext4 714
- mkfs.ntfs 737
- mkfs.vfat 737
 - EFI-Dateisystem* 738
- mkfs.xfs 735
- mkinitcpio 788
- mkinitramfs 786
- mklabel 687, 690
- mkosi 787
- mkpart 690
- mkpasswd 490
- mkswap 741, 742
- Mobile Shell 923
- modinfo 830
- modprobe 829
- modprobe.conf 564, 831
- Module (Kernel)
 - Abhängigkeiten* 832
 - automatisch laden* 832
 - Device Trees* 834
 - Device-Dateien* 833
 - kompilieren* 837, 849
 - module-assistant* 840
 - modules.alias* 832
 - modules.dep* 832
 - Optionen* 833
 - Parameter* 830
 - Stripping* 850
 - Versioning* 828
 - verwenden* 829
- Module (DNF) 598, 876
- mogrify 388
- MOKs (Secure Boot) 782
 - mokutil* 850
 - NVIDIA* 655
- Monitoring 1215
 - Alerts* 1243
- Monolithischer Kernel 847
- more 234
- moreutils 252

-
- Mosh 923
 - mount 701, 702
 - Beispiele* 703
 - Optionen* 707
 - remount für Systempartition* 705
 - mp3ogg 389
 - mpgl23 389
 - MTA 991
 - mtab 702
 - MUA 992
 - Multicast-Adressen (IPv6) 560
 - multiuser-Target 666
 - MultiViews 950
 - Musik (Audio-Player) 183
 - Musik-Verzeichnis 142
 - Musique (Audio-Player) 183
 - mutt 417
 - Mutter (Gnome Shell) 650
 - mv 320
 - Dateien umbenennen* 320
 - Sicherheitsabfragen* 87
 - MX-Eintrag (DNS) 998
 - mydestination 1008, 1025
 - myhostname 1008
 - mynetworks 1008
 - myorigin 1008
 - MySQL 969
 - Administration* 979
 - Backups* 983
 - Exporter (Prometheus)* 1255
 - IPv6* 973
 - Monitoring* 1255
 - mysql (Kommando)* 979
 - mysqladmin (Kommando)* 980
 - mysqldump (Kommando)* 983
 - Workbench* 980
- N**
-
- Nachtmodus 125
 - Name Service Switch (NSS) 492
 - Nameserver
 - aktuelle Adresse herausfinden* 553
 - Client-Konfiguration* 558, 574
 - frei verfügbare* 558
 - Ubuntu* 553
 - Namespace (PCIe) 681
 - nano 235
 - Nautilus (Gnome) 118
 - nautilus-image-converter* 120
 - nautilus-nextcloud* 1075
 - nc (Kommando) 410, 1029
 - SMTP-Fehlersuche* 1059
 - ncdu 317, 331
 - ncrack 909
 - ncspot 184
 - needrestart (DPKG) 621
 - needs-restarting (DNF) 600
 - negativo-Paketquelle (Fedora) 658
 - net-tools 1166
 - NetBIOS 1083
 - netcat 410, 1029
 - SMTP-Fehlersuche* 1059
 - Netfilter (Firewall-System) 1174
 - Netpbm 388
 - netplan 582
 - Netzwerkbrücke* 1303
 - netstat 1166
 - Network File System 699
 - Network Time Protocol (NTP) 479
 - Network-Maske 557
 - networkd 584
 - NetworkManager 547
 - Netzwerkbrücke* 1303
 - Plug-ins* 554
 - Netzwerk 555
 - Aktivität überwachen* 1166
 - Konfiguration* 547, 563, 576
 - Schnittstelle* 556
 - Sicherheit* 1163
 - Netzwerkbrücke 1302
 - VirtualBox* 1274
 - Neustart
 - erzwingen* 822
 - KVM-Hostsystem* 1286
 - nach Update* 600, 621
 - newaliases 995, 1021
 - newgidmap 1362
 - newgrp 344
 - newuidmap 1362
 - Nextcloud 1061
 - Backups* 1072
 - Dateien synchronisieren* 1074
 - Interna* 1072
 - nextcloud-client* 1074
 - Updates* 1073
 - nextd (fish-Kommando) 304
 - nfs 699
 - nft (Firewall) 1174, 1181
 - Beispiel* 1191
 - Geo-Blocking* 1193
 - Prometheus Node Exporter* 1241
 - nginx 925, 965
 - NIC 556
 - nice 363
 - NixOS 39, 592

nl80211-Schnittstelle 569
nm-connection-editor 1303
nmap 1168
nmbd (Samba) 1086
nmblookup 1113
nmcli 552
 Netzwerkbrücke 1303
noacpi/noapic (Kerneloption) 859
noatime (ext4-Option) 717
noauto 708
nodatacow (btrfs-Option) 731
Node Exporter (Prometheus) 1216, 1222
 absichern 1238
 Firewall 1241
nodeadkeys 475
nodev-Dateisysteme 699
nodev (mount-Option) 708
noexec 708
nofail 708
 für EFI-Partition 874
noht (Kerneloption) 859
nolapic (Kerneloption) 859
nomatch 293
Nomic (GPT4All) 1379
nomodeset (Kerneloption) 858
Non-Free-Pakete 614
none-Dateisystem 700
noresume (Kerneloption) 859
nosmp (Kerneloption) 859
nosuid 708
Notebook-Optimierung 517
 Akku-Ladeverhalten 521
 Batterie 504
 Lüftersteuerung 522
nouveau (Grafiktreiber) 656
NOVA (Grafiktreiber) 656
nproc 512, 849
ntfs 698, 736
ntfsprogs 677, 737
ntpd 480
ntpddate 480
nullok (PAM-Konfiguration) 920
NVIDIA 656
 Debian 83
 DGX Spark 1379
 Fedora 88
 nvidia-prime-select 660
 nvidia-settings 660
 nvidia_smi 664
 PRIME 660
 Secure Boot 45
 Treiberinstallation 657
 Ubuntu 95

nvme (Kommando) 505
nvme-cli 505, 684
NVMe-Protokoll 680
nvtop 664

O

occ (Nextcloud) 1073
ocfs-Dateisystem 674, 700
Offener DNS-Resolver 1281
Öffentlich-Verzeichnis 142
oggdec/oggenc 389
Oh my zsh 298
Ollama 1381
ondemand (CPU Governor) 514
Oneshot-Typ (systemd) 816
Online-Konten (Gnome) 123
open (Kommando) 143, 358
Open Container Specification (OCI) 1361
Open GL 652
Open Source 34
 Open Source Initiative (OSI) 36
Open WebUI 1388
OpenAI 1397
OpenCL 652
OpenDKIM 1050
OpenMediaVault (OMV) 1086
openssh 905
openssl 934
openSUSE 33, 71
 opensuse-migration-tool 608
 zypper 607
Optimus-Hybrid-Grafik 660
Optionen
 Apache 950
 Kernel 846
 Module (modprobe.conf) 833
Oracle
 Cluster Filesystem 674
 Linux 32, 879
 MySQL 969
 VirtualBox 1261
Order 951
Origins-Patterns 621
os-prober 795
Overlay-Dateisystem 700, 1357
 Docker 1357
Overlays (Device Trees) 835
override.conf (systemd-Konfiguration) 812
ovmf 1294
ownCloud 1061
owner 708

P

-
- p7zip 1148
 - paccache 631
 - PackageKit 632
 - packagekitd 632
 - Pacman 627
 - pages_limit (Kerneloption) 1385, 1396
 - pages_pool_size (Kerneloption) 1385, 1396
 - Pakete (Software)
 - Abhängigkeiten 606
 - Debian 614
 - Ubuntu 616
 - Verwaltung 587
 - Paketfilter 1174
 - PAM 493
 - Google Authenticator 918
 - pam-auth-update 493
 - pam_cracklib 489
 - pam_pwquality 489
 - pam_unix 489
 - systemd 812
 - Pamac 632
 - pandoc 397
 - Panel (Gnome) 107
 - Panel (KDE) 150
 - Papierkorb (Gnome) 116
 - Papierkorb (Samba) 1104
 - Parallel SSH 922
 - Parametersubstitution 277
 - Paravirtualisierung 1286
 - Parity Striping 745
 - parted 688
 - EFI-Partition 738, 780
 - Parted Magic 33
 - Partition
 - ändern, Linux 56
 - Bezeichnung unter Linux 680
 - Dateisystem 59
 - EFI 780
 - Grundlagen 50
 - ideale Partitionierung 57
 - remount 705
 - klonen 754
 - Partitions Grenzen 688
 - Typen 52
 - vergrößern 896
 - verwalten 688
 - Verzeichnisbaum 676
 - paru 631
 - passdb backend 1089, 1095
 - Passwort 487
 - Ablaufdatum (*chage*) 488
 - Apache 953
 - aging 488
 - ändern 489
 - für Gruppen 491
 - PAM 493
 - Qualität 489
 - root 489
 - Samba 1094
 - vergessen 490
 - path (fish) 308
 - path (Samba) 1100
 - PATH (Variable) 265
 - pavucontrol 512
 - PCI-Bus 505, 680
 - Kerneloption 858
 - pdbedit 1095
 - PDC 1084
 - PDF-Tools 391, 393
 - pdksh 242
 - performance (CPU Governor) 514
 - Periodische Ausführung von Jobs 378, 382
 - pesign 844
 - PGP 997
 - Phonon 511
 - PHP 962
 - Emacs-Erweiterung 468
 - FastCGI Process Manager 963
 - memory-limit 1066
 - output-buffering 1066
 - phpMyAdmin 981
 - Unicode 931
 - Physical Extent 755
 - Physical Volume 54, 755
 - Pico (Raspberry-Pi-Modell) 193
 - PID 360
 - PID-Datei 361
 - pidof (Kommando) 361
 - pinctrl 214
 - ping 401
 - Podman 1363
 - Windows 1120
 - Pipes 253
 - PipeWire 512
 - Piwik 959
 - PIXEL-Desktop 202
 - pkcon 632
 - pkexec 366, 373
 - pkmon 632
 - Plasma 148
 - Plasmoids 149
 - Plesk Panel 472

plocate	326	printcap (CUPS)	526
Plug-ins (DNF)	596	printenv	265
Pluggable Authentication Modules	493	printers.conf	526
pnuke	922	pro (Kommando)	618
Podman	1313	Procmail	992
<i>Desktop</i>	1318, 1319	profile	244, 264, 265
<i>Installation</i>	1319	Programm (Prozessverwaltung)	357
<i>Interna</i>	1361	Prometheus	1215
<i>Pods</i>	1364	<i>Alert Manager</i>	1244
<i>Rootful</i>	1363	<i>Apache Exporter</i>	1255
Policy-Dateien (X)	374	<i>Konfiguration</i>	1229
policycoreutils-gui	1204	<i>MySQL Exporter</i>	1255
PolicyKit	372	<i>nginx Exporter</i>	1251, 1255
Polkit	372	<i>Node Exporter</i>	1216, 1222
POP-Server	993, 1029	<i>prometheus.yml</i>	1230
Pop_OS	33, 72	<i>Query Language (PromQL)</i>	1236
<i>systemd-boot</i>	800	<i>Retentionszeit</i>	1229
Poppler	394	<i>Weboberfläche</i>	1233
Ports (TCP/IP)	555	Prompt	
<i>FTP (20, 21)</i>	1164	<i>bash</i>	245
<i>HTTP (80)</i>	1164	<i>PROMPT_COMMAND</i>	246, 265
<i>IMAP (25, 587)</i>	994	<i>zsh</i>	299
<i>Liste</i>	1164	Protokoll-Dateien (Logging)	531
<i>Referenz</i>	1164	Proxmox	1289
<i>Scan</i>	1168	Proxy-Konfiguration	550
<i>SMTP (25, 587)</i>	994	Prozesse	357
Portable Bitmap Utilities	388	<i>gewaltsam beenden</i>	363
postalias	1021	<i>Größe begrenzen</i>	364
Postfach (Mail-Server)	1004, 1027	<i>Hierarchie</i>	362
Postfix	1003	<i>Hintergrundprozesse</i>	358
<i>Alias</i>	1021	<i>Priorität</i>	363
<i>IPv6</i>	1010	<i>Rechenzeit</i>	363
<i>Logging</i>	1011	<i>regelmäßig ausführen</i>	378, 382
<i>virtuelle Domänen</i>	1025	<i>unter anderer Identität ausführen</i>	364
postmap	1006, 1024	<i>unterbrechen</i>	359
postqueue	1012	<i>verwalten</i>	359
PostScript	524	<i>Vordergrundprozesse</i>	358
<i>PDF-Konverter</i>	391	ps	359
<i>Printer Definition (PPD)</i>	527	PS1	245, 265
<i>Utilities</i>	392	ps2pdf	391
power-profiles-daemon	514	psbook	392
powerfilesctl	514	psnup	392
powersave (CPU Governor)	514	psresize	392
powertop	517	psselect	392
PPAs (Ubuntu)	617	pssh	922
PPD-Dateien	527	pstops	392
PPP	555	pstree	362
Präfix-Notation (Netzwerkadressen)	557	psutils	392
prevd (fish-Kommando)	304	ptyxis	231
Primary Domain Controller	1084	Pull Limit (Docker)	1316
prime-run	660	PulseAudio	512
prime-select	660	Punycode	575
Primäre Partition	52	pw-top	512

Q

QCOW2-Format	1287
<i>Image vergrößern</i>	1311
QEMU	1279, 1280
<i>qemu-img</i>	1307, 1311, 1312
<i>qemu-kvm</i>	1283
<i>qemu-system-x86</i>	1283
Qt (Bibliothek)	148
Quantisierung (LLMs)	1395
Quellpaket	605
queue (Druckerwarteschlange)	524
quiet (Kerneloption)	858
Quotas	674

R

radeon (Grafiktreiber)	655
<i>radeontop</i>	664
RAID	744
LVM	756
RAID-0	745
RAID-1	745
RAID-10	745
raid-check	753
Scrubbing	753
Server-Konfiguration	870
Überwachung	748
RANDOM-Variable	265
RANDOM_DELAY	382
RandR	652, 670
Raspberry Pi	191
Connect	205
Device Trees	834
NAS	1086
Raspberry Pi OS	32, 195, 820
RAW-Bildateien	180, 1287
rb	1160
rc-Dateien	824
rc.local	820
rclone	1153
rdiff-backup	1138
RDP (Wayland)	651
read	279
readline	244
reboot	821
erzwingen	822
KVM-Hostsystem	1286
reboot-required-Datei	621
Rechnerstart	779
Probleme	64
recode	391
recycle	1105
Red Hat	32, 875

<i>Image Mode</i>	592
<i>initrd</i>	787
<i>Satellite</i>	589
<i>statische Netzwerkkonfiguration</i>	576
<i>UBI</i>	1331
redshift-Programm	125
ReFS-Dateisystem	698, 737
Regelmäßige Ausführung von Jobs	378, 382
Reguläre Ausdrücke	329
Emacs	460
relatime (ext4-Option)	717
Relaying	997
relayhost (Postfix)	1009
reload	824
RemainAfterExit-Schlüsselwort (systemd)	816
REMI-Paketquelle	602
remount	705
remove (modprobe.conf)	833
Rendezvous	585
renice	363
Require	951
reserve (Kerneloption)	858
reset	234
resize2fs	716
resolv.conf	574
Ubuntu	575
resolvectl	553, 584
restart	824
restic	1146
restorecon	1202, 1206
Retentionszeit (Prometheus)	1229
Retina-Bildschirme	125, 1270
Reverse DNS	1001
AWS EC2	898
Reverse Proxy	1220
RHEL	875
Firewall	1175
Image Mode	39, 592
Paketverwaltung	593
RHSM	875
Tastatur	476
Rhythmbox (Audio-Player)	183
Richard Stallman	37
RIDL	860
rlogin	404
rm	316
Sicherheitsabfragen	87
rmmod	831
ro (Kerneloption)	857
Rocky Linux	879
ROCm (Bibliothek)	653, 1381
Ollama	1383

Rolling Release	30, 592	Schlafmodus	504
<i>Ubuntu-Kernel</i>	95	Schleifen	282
root	61, 489	Schlüssel	
<i>Kerneloption</i>	857	<i>HTTPS (Apache)</i>	933
<i>MySQL</i>	974	<i>IMAP/SMTP (Dovecot)</i>	1035
Root-Partition	57	<i>SSH</i>	911
root-Passwort vergessen	490	Schnittstelle	556
Root-Server	868	Schriftarten (Fonts)	128
Rootful Podman	1315, 1363	<i>Emacs</i>	465
Rootless Docker	1315, 1321	<i>Textkonsolen</i>	477
route	565, 566	scp	408
Routing-Tabelle	557	Scraping (Prometheus)	1230
rpi-eeprom	203	screen	923
rpi-update	203	Screenshots	136, 159
rpica-m-xxx (Raspberry-Pi-Kommandos)	217	<i>Wayland</i>	651
rpm	604	ScriptAlias	949
<i>cannot open packages database</i>	606	Scripts	266, 274, 407
<i>Datenbank reparieren</i>	606	Scrubbing (RAID)	753
<i>RPM Fusion (Fedora)</i>	87	SCSI	680
<i>rpmkeys</i>	606	SD-Karte	189, 676
<i>RPMS</i>	605	sdboot-manage	801
RSA (SSH-Schlüssel)	912	seahorse	914
rsnapshot	1140	seahorse-nautilus	120
rsvg-convert	389, 389	sealert	1206
rsync	1150	Secure Boot	44, 64, 782
rsyslogd	531	<i>eigener Kernel</i>	850
Ruhezustand	504	<i>NVIDIA</i>	45
<i>Neustart erzwingen</i>	822	Secure Shell	905
RUN (Dockerfile)	1348	Secure Sockets Layer	932
run-Installationsprogramme	591	security (Samba)	1089
run-Kommando (Docker)	1340	securityfs	1208
run-parts	381	sed	320
runO	371	Selektor (Syslog)	532
Runlevel	823	SELinux	1199, 1200
rygel	138	<i>Apache</i>	928
		<i>Docker</i>	1333
S		<i>eigener systemd-Service</i>	1400
S3-Cloud-Dienst	1157	<i>Google Authenticator</i>	920
Samba	1081, 1083	<i>libvirt</i>	1300
<i>/etc/fstab</i>	1118	<i>opendkim</i>	1052
<i>Firewall</i>	1090	<i>Samba</i>	1092, 1101, 1102
<i>Gäste</i>	1103	<i>selinux-policy-mls</i>	1204
<i>IPv6</i>	1092	<i>SSH</i>	915
<i>Nautilus</i>	118	<i>SSH-Port</i>	908
<i>Netzwerkverzeichnisse einrichten</i>	1100	Sender Policy Framework	1048
<i>Papierkorb</i>	1104	sensors	516
<i>Passwörter</i>	1094	sensors-detect	515
<i>SELinux</i>	1092	Server-Installation	867
<i>Sicherheitsmechanismen</i>	1083	<i>Datenbank (MySQL)</i>	969
Sandbox (Flatpak/Snap)	639	<i>Samba</i>	1083
SATA	680	<i>SSH</i>	905
schedutil (CPU Governor)	514	<i>Webserver (Apache)</i>	925
		<i>X</i>	649

Server Message Block	1083	Smack	1201
server role (Samba)	1089	SMART	760
server string (Samba)	1089	smartctl	761
ServerAlias	956	smartd	765
ServerName	930, 956	SMB-Protokoll	1083
service (Kommando)	814, 825	<i>smb.conf</i>	1087
Service (systemd)	811	<i>smbclient</i>	1114
Services	375	<i>smbd (Samba)</i>	1086
sestatus	1205	<i>smbfs</i>	699, 1116
set	258, 265, 307	<i>smbpasswd</i>	1095
setcap	351	<i>smbstatus</i>	1087, 1090
setenforce	1207	<i>smbtree</i>	1113
setfacl	348	<i>Versionen</i>	1083
setfattr	350	SMT	862
Setgid-Bit	342	SMTP	993
setopt	294	<i>Authentifizierung</i>	1036
setroubleshoot-server	1206	<i>Fehlersuche</i>	1059
setsebool	1204	<i>smtp_tls-Parameter (Postfix)</i>	1014
Setuid-Bit	341	<i>smtpd_tls-Parameter (Postfix)</i>	1014
sfdisk	754	Snakeoil-Zertifikat und -Schlüssel	
sftp	416, 906	<i>Apache</i>	938
sgdisk	754	<i>Postfix</i>	1012
SGI-Dateisystem	697	Snap	642
sh-Datei (Software-Installation)	590	snap-store	589
shadow	487	snapd	644
shadowutils	1362	Snapdragon-CPU	28
Share-Level-Sicherheit	1083	Snapper	733
Shared Folder (VirtualBox)	1276	Snapshots	721, 759
Shares	1083	socat	1299
Shebang	267	Socket-Dateien	322
Shell		soft_bounce (Postfix)	1010
<i>aktive Shell herausfinden</i>	290	Software-Installation	587
<i>bash</i>	241	software-properties	619
<i>Programmierung</i>	266, 385	Software-RAID	744
<i>Standard-Shell ändern</i>	243	Solid (KDE-Framework)	508
<i>Variablen</i>	263, 274	Sonderzeichen (bash)	286
<i>zsh</i>	289	Sound Converter	187
Shim	44, 782	Sound-System (ALSA)	510
shopt	258	source (bash)	275
Shotwell	173	sources.list	612
Showtime	185	sox	390
shutdown	821	SpamAssassin	1038
Shuttleworth Mark (Ubuntu)	89	speaker-test	510
Sicherheit	1163	special bits (Zugriffsrechte)	341
Sicherheitskontext	1201	Spectacle	159
Sieve	1043	Spectre	633, 860
Silverblue (Distribution)	39, 83, 592	<i>spectre-meltdown-checker.sh</i>	861
Simple Storage Service (S3)	1157	SPF-Eintrag (Mail-Server)	1048
single (Kerneloption)	857	SPICE	1288
Single-User-Modus (systemd)	810	<i>spice-vdagent</i>	1295
Site-Local-Adressen (IPv6)	560	Spin (Fedora)	83
skip-networking	973	sponge	252
Slax	33	Spooling-System (CUPS)	524

Spotify	184	su	365
Spracheinstellung	497	<i>Wayland</i>	651
squashfs-Dateisystem	700	Subiquity	884
<i>Snap</i>	644	submission (Postfix)	1011
Squid	1220	Substitutionsmechanismen (bash)	256
SRPMS	605	subuid/subgid-Datei	1362
SSD-TRIM	766	Subvolumes (btrfs)	720
SSD (Raspberry Pi)	219	<i>subvol (Option)</i>	731
ssh	404, 905	sudo	366
<i>absichern</i>	907	<i>Ein-/Ausgabeumleitung</i>	370
<i>Dateisystem</i>	410	<i>ohne Passwort</i>	368
<i>Google Authenticator</i>	916	<i>Raspberry Pi OS</i>	201
<i>Konqueror</i>	155	<i>sudo-rs</i>	371
<i>IPv6</i>	908	<i>Wayland</i>	370, 651
<i>libvirt</i>	1291	suid	341
<i>Login vermeiden</i>	911	SUSE	33, 71
<i>Mobile Shell</i>	923	<i>Adaptable Linux Platform (ALP)</i>	39
<i>Port ändern</i>	908	<i>Kernelkonfiguration</i>	847
<i>Portumleitung</i>	528	<i>zypper</i>	607
<i>SELinux</i>	915	Suspend to Disk	504
<i>Server</i>	905	<i>Kerneloptionen</i>	859
<i>ssh-agent</i>	914	SVG-Konverter	389
<i>ssh-copy-id</i>	913	swaks	1042
<i>ssh-keygen</i>	911	Swap on ZRAM	742
<i>sshd</i>	905	Swap-Datei	742
<i>sshfs</i>	410, 699	Swap-Partition	58, 740
<i>SSHGuard</i>	909	<i>EC2-Instanz</i>	893
<i>Tunnel</i>	409	swapon	741, 742
<i>VSCode</i>	431	swappiness (Kernel-Parameter)	743
SSL (Apache)	932	Sway	648
<i>ssl-cert-snakeoil.pem</i>	938	switcheroo-control	660
<i>SSLCipherSuite</i>	945	Symbolische Links	323
<i>SSLEngine</i>	938	<i>versus Bind-Mounts</i>	710
<i>SSLProtocol</i>	945	Synaptic	611
<i>SSLStrictSNIVHostCheck</i>	956	sync	1160
<i>SSLxxxFile</i>	938	Synchronisations-Tools	1125, 1131, 1133, 1150
Stable-Pakete	614	Syncthing	1133
Stallman, Richard	34	sysctl	860
Standardeingabe und -ausgabe	251	sysfs	699
star (tar mit ACL)	350	syslog	1093
start.elf-Datei	203	System Monitor (Gnome)	135
Startprobleme	64	system-config-printer	123
STARTTLS	1012, 1035	system-config-selinux	1204
stat	337	System-V-Init-Prozess	823
Statische Netzwerkkonfiguration	576	Systemadministration	471
Sticky-Bit	343, 345	systemctl	382, 807
Strawberry (Audio-Player)	184	<i>edit</i>	713, 812
stress-ng	1247	systemd	805
string (fish-Kommando)	308	<i>/etc/fstab</i>	710
stripcomments	268	<i>Cron-Ersatz</i>	382
Striping	745	<i>daemon-reload</i>	710
Stromsparfunktionen	504	<i>Discoverable Partitions Specification</i>	711
Stubble (Device Trees)	834	<i>eigene Service-Datei</i>	815

<i>Gnome</i>	144	telnet	404, 410, 1029
<i>Grafiksystem starten</i>	666	<i>SMTP-Fehlersuche</i>	1059
<i>llama-server</i>	1400	Temperatur (CPU)	515
<i>Netzwerk-Device-Namen</i>	565	Temperaturmessung (Raspberry Pi)	215
<i>Netzwerkkonfiguration</i>	584	Temporäres Verzeichnis	712
<i>Netzwerkschnittstellen</i>	557	Terminal	231
<i>Prozesse periodisch ausführen</i>	382	test (bash)	280
<i>Raspberry Pi OS</i>	820	Tethering	549
<i>SELinux</i>	1400	Text-Konverter	391
<i>Service-Datei für den Node Exporter</i>	1224	Textdatei durchsuchen	328
<i>systemd-analyze</i>	542	Texteditoren	235, 423, 451
<i>systemd-boot</i>	799	Textkonsole	230
<i>systemd-gpt-auto-generator</i>	711	<i>Konfiguration</i>	475
<i>systemd-journald</i>	539	<i>Schriftart</i>	477
<i>systemd-networkd</i>	584	<i>Tastatur</i>	475
<i>systemd-resolved</i>	553, 584	Thumbnails	271, 387
<i>systemd-stub</i>	834	Thunderbird	165
<i>systemd-timedated</i>	477	<i>CalDAV/CardDAV</i>	1077
<i>systemd-timesyncd</i>	480	Thunderbolt	509
<i>systemd-udev</i>	355	Tiling Assistant (Gnome)	135
<i>systemd-udev</i>	355	Time Machine	1153
<i>systemd-vconsole-setup</i>	477	time-sync	480
<i>temporäres Verzeichnis</i>	712	timedatectl	477, 479, 814
<i>Timers</i>	382	Timers (systemd)	382
<i>WSL</i>	1369	timestamp_timeout	367
<i>ZRAM-Swap</i>	742	TinyCore	33
Systempartition	57	tldr	238, 331
<i>remount</i>	705	TLP	519
Systemstart	819	tlp-stat	520
<i>GRUB</i>	791	TLS	997
<i>Init-V</i>	823	<i>Dovecot</i>	1035
		<i>Postfix</i>	1012
T		tmpfs	699
		<i>/tmp-Verzeichnis</i>	712
Tabulatoren (Emacs)	456	tmux	924
tail	234	top	360
Tainted Kernel	654	<i>für GPUs</i>	664
Talk (Nextcloud)	1079	Torvalds, Linus	38
tar	1147, 1149	Totem	185
Target (systemd)	808, 810	touch	344
targeted (SELinux)	1203	Touchpad deaktivieren	122
Tartarus	1154	traceroute	403
tasksel	622	Traefik	1220, 1228
Tastatur		Transkodieren (HandBrake)	187
<i>Gnome</i>	122	Transport Layer Security	997
<i>KDE</i>	158	Treiberinstallation (Ubuntu)	619
<i>Konfiguration</i>	475	TRIM (SSDs)	766
<i>US-Tastaturliste</i>	63	<i>LUKS-Verschlüsselung</i>	771
TCP-Wrapper-Bibliothek	1170	Trixie	615
TCP/IP	555	Troll Tech	148
tcsh	242	TrueNAS	1086
TDB	1095	Trusted Platform Modul (TPM)	1295
tee	254	Trusted TLS Connection (Postfix)	1015

ttf-mscorefonts-installer	129
tune2fs	715
tuned-adm	515
tuned-ppd	515
Tunnel (SSH)	409
Turbo Boost	514
type (Kommando)	247
Type-Schlüsselwort (systemd)	816

U

Ubuntu	33
DKMS	838
Docker-Image	1326
initrd	786
Installation	89
Paketverwaltung	616
Pro	618, 883
statische Netzwerkkonfiguration	582
Tastatur	475
Ubuntu Pro	852
Ubuntu Server	90
Ubuntu Studio	90
ubuntu-drivers	95, 619
ubuntu-restricted-extras	95
VirtualBox	1271
udev-System	355, 508
Netzwerk-Device-Namen	565
udisks2	508
UDP	555, 1165
UEFI	42
in KVM/QEMU	1294
Partition	43
Secure Boot	44, 64, 782, 850
ufw	1179
Uhrzeit	477
UID	485
uidmap	1362
ukify	834
umask	345
Umgebungsvariablen	264
umount	703
Unicode	497
Apache	931
Dateisystem	311
Konsole	477
PHP	931
Unified Kernel Image (UKI)	785
unionfs-Dateisystem	700
Unit (systemd)	808, 810
Univention Corporate Server	472
Universal Base Images (Red Hat)	1331
Unix	27

Unix Pseudo TTYs	699
unmkinitramfs	788
unset	265
Unstable-Pakete	615
Untrusted TLS Connection (Postfix)	1015
unxz	1147
unzip	1147
update-alternatives	637
update-grub	793
update-initramfs	786
update-ms-fonts	129
updatedb	326
Updates	65
BIOS/EFI/Firmware	633
Kernel	851
upower	508
US-Tastaturtabelle	63
USB	505
USB-Stick	189, 676
usbreset	505
User einrichten	482
User-Level-Sicherheit	1084
user_xattr	347
useradd	483
usermod	368
username map	1097
UTC (Universal Time, Coordinated)	477
UTF-8	497
UUID	
ermitteln	706
in /etc/fstab	706
in /dev/disk	685
utils	333

V

valid users (Samba)	1100
Vanilla Kernel	845
Vanilla OS	39, 592
Variablen (Shell)	263, 274, 279, 307
varlock	699
varrun	699
vboxdrv	1263
vboxguest	1271
vboxmanage	1277, 1278
vboxnetadp	1263
vboxnetflt	1263
vboxsf-Dateisystem	1276
vcgencmd	224
VDPAU	653
Vergleiche (bash)	280
Verschlüsselung	55
Dateisysteme	674, 767

Mail-Server	997	Virtueller Swap-Speicher	742
Versteckte Dateien	312	VISUAL	238
Verzeichnis	312	Visual Studio Code	424
Grundlagen	312	visudo	367
Partitionen	676	vmlinuz	781, 850
synchronisieren	1133	VNC	1288
Verzeichnisbaum	351	Wayland	651
Verzweigungen (bash)	279	vol_id	706
vfat	698, 736	VOLUME (Dockerfile)	1349
vgcreate	757	Volume Group	54, 755
vgscan	757	Volumes	
Vi	237, 435	Docker	1333, 1344
video (Kerneloption)	858	LVM	54
Video-Dateien transcodieren	187	Vordergrundprozesse	358
Videokonferenz (Nextcloud Talk)	1079	Vorlagen-Verzeichnis	142
Videos-Verzeichnis	142	Vorta	1146
Vim	237, 435	VRAM	1378
Cursorbewegung	438	versus GTT	1384, 1396
Konfiguration	447	VRFY-Kommando (Postfix)	1029
Maus	449	VSCoDe	423, 424
Optionen	447	Git	430
suchen und ersetzen	443	mit Ollama	1391
Swap-Datei	448	Remote-SSH-Erweiterung	431
Tabulatoren	449	VSCodium	430
vimrc	447	Vulkan	653
Virenschutz	1045	llama.cpp	1393
virsh	1284, 1296	vulkaninfo	1393
virt-cat	1310	vulnerabilities-Dateien	861
virt-clone	1299		
virt-df	1308	W	
virt-edit	1310		
virt-filesystems	1309	w3m	416
virt-inspector	1309	WannaCry-Schad-Software	1092
virt-make-fs	1310	Warteschlange	524
virt-manager	1284, 1289	watchdog	376
virt-resize	1311	Wayback	650
virt-sparsify	1312	Wayland	647, 649
virt-sysprep	1300	Einschränkungen	651
virt-tar-in/-out	1310	Protokoll	665
virt-top	1302	sudo	370
virt-viewer	1301	WSL	1365
virtio-Treiber	681, 1286	Zwischenablage bei Virtualisierung	1295
Virtual Private Cloud (VPC)	899	Web	164
Virtual Private Networks (VPN)	550	Webalizer	958
virtual_mailbox_domains	1027	Webbrowser	162
virtual_alias_domains	1026	im Textmodus	416
VirtualBox	1261	WebDAV	1075
Image in QCOW2-Format umwandeln	1311	Gnome-Freigabe	137
Virtuelle Dateisysteme	699	Webhosting	869
Virtuelle Domänen (Postfix)	1025	Webmin	472
Virtuelle Hosts (Apache)	955	Webproxy	1220
mit HTTPS	956	Webserver	925
Virtuelle Postfächer	1027	Monitoring	1251

W

w3m	416
WannaCry-Schad-Software	1092
Warteschlange	524
watchdog	376
Wayback	650
Wayland	647, 649
<i>Einschränkungen</i>	651
<i>Protokoll</i>	665
<i>sudo</i>	370
<i>WSL</i>	1365
<i>Zwischenablage bei Virtualisierung</i>	1295
Web	164
Webalizer	958
Webbrowser	162
<i>im Textmodus</i>	416
WebDAV	1075
<i>Gnome-Freigabe</i>	137
Webhosting	869
Webmin	472
Webproxy	1220
Webserver	925
<i>Monitoring</i>	1251

website 1160
WebUI (Ollama) 1388
Webverzeichnis
 absichern 953
well-known-DAV-Umleitungen 1067
wget 411
whereis 247, 325
which 247, 325
while 284
Whitelist (SpamAssassin) 1042
Window Manager 649
Windows
 Dateisystem 698, 736
 KVM-Installation 1295
 Netzwerkverzeichnisse 1083, 1116
 Samba-Client-Konfiguration 1120
 Subsystem for Linux (WSL) 1365
WINS 1083
wl-clipboard 1295
WLAN 549
wlr-randr 671
WoeUSB 189
WordPress 986
workgroup (Samba) 1089
Workgroup-Sicherheit 1084
WPA 571
 wpa_passphrase 572
 wpasupplicant 571, 581
writeable 1100
WS-Discovery (WSD) 1083, 1087
wsdd 1087
WSL 1365
 wsl (Kommando) 1373
 wsl.conf 1369, 1374
 wslconfig 1374

X

X Window System 647
 Konfiguration 670
 Logging 665
 Protokoll 665
 Server 649
 Window Manager 649
x265 (Codec) 172
xargs 261
XDG 142
xdg-desktop-icon 143
xdg-desktop-menu 143
Xdg-desktop-portal 651
xdg-email 143
xdg-icon-resource 143
xdg-mime 143

xdg-open 143, 358
xdg-screensaver 143
xdg-user-dirs 142
XDG_SESSION_TYPE (Umgebungsvariable) 661
xdm 669
xe (Grafiktreiber) 656
Xfce 648
xfs 697, 734
 xfs_admin 736
 xfs_check 735
 xfs_growfs 735
 xfs_repair 735
Xorg.O.log 665
xorg.conf 670
xrandr 670
XRender 653
xsensors 516
Xubuntu 33, 90
xvda-Devices 892
Xwayland 650
xz 1147, 1148

Y

YaST 71
yay 631
yum 593
 automatische Updates 600
 yum.conf 595

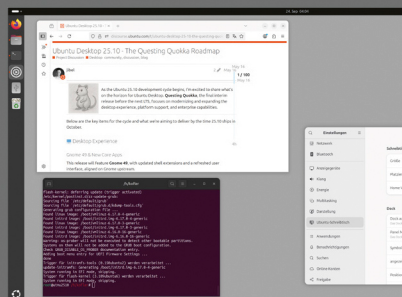
Z

z (Kommando) 331
Zahlenvergleiche (bash) 280
Zeichenketten
 bash 262
 Parametersubstitution (bash) 277
Zeichensatz 497
 Apache 931
 PHP 931
 ändern 391
Zeitgesteuerte Ausführung von Jobs 378, 382
Zeitzone 477
Zentyal 472
ZENworks 472
Zero (Raspberry-Pi-Modell) 193
Zero Install 592
Zeroconf 585
Zertifikat
 erstellen und signieren 934
 HTTPS (Apache) 933
 IMAP/SMTP (Dovecot) 1035

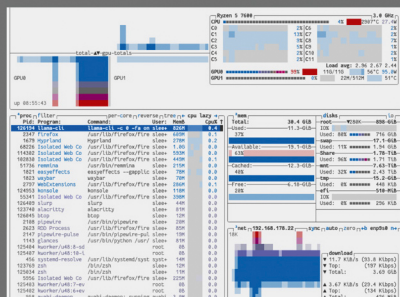
<i>Postfix</i>	1012
<i>Snakeoil-Zertifikat</i>	938
ZFS-Dateisystem	697
zile	236, 451
zip	1147
zipinfo	1147
ZombieLoad	860
Zorin OS	90
ZRAM-Device	742
zramctl	743
zsh	289
<i>zsh-newuser-install</i>	292
<i>zsh-syntax-highlighting</i>	300
<i>zshrc</i>	291
Zugriffsbits	334
<i>bei neuen Dateien</i>	345
<i>setuid, setgid</i>	341
<i>sticky</i>	343
Zugriffsrechte	334
Zugriffssteuerung (Benutzerverwaltung)	481
Zwei-Faktor-Authentifizierung	916
Zwischenablage	113
<i>KVM/libvirt</i>	1295
<i>VirtualBox</i>	1275
zypper	607

»Der Kofler« ist das Linux-Standardwerk für Einsteiger und Profis

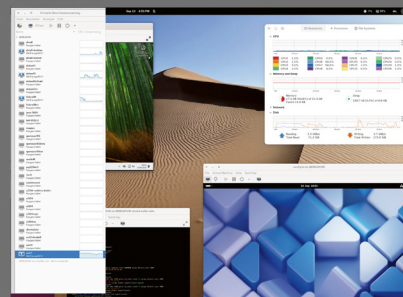
Das freie und offene Betriebssystem Linux hat vor 35 Jahren die IT-Welt revolutioniert. Fast genauso lange begleitet das umfassende Handbuch von Michael Kofler den Pinguin. Seit nunmehr 30 Jahren finden Sie dort alles, was Sie zu Linux wissen müssen – ein echtes Standardwerk eben.



Den richtigen Einstieg finden



KI-Modelle lokal ausführen



Cloud und Virtualisierung

Grundlagen verstehen und direkt loslegen

Sie lernen gängige Distributionen wie Ubuntu, Debian, CachyOS, Fedora oder RHEL kennen und richten Ihr System so ein, wie Sie es brauchen. Wählen Sie den passenden Desktop aus, machen Sie es sich auf der Shell gemütlich und fühlen Sie sich unter Linux wie zu Hause.

Ihren Desktop individuell einrichten

Wollen Sie Linux auf Ihrer Workstation nutzen oder suchen Sie ein schlankes System für den Servereinsatz? Sie arbeiten mit modernen Shells, greifen mit SSH und 2FA auf Cloud-Systeme zu oder bauen Entwicklungssetups mit VSCode und Ollama auf.

Linux professionell nutzen und administrieren

System- und Netzwerkkonfiguration, Monitoring und Dashboards mit Grafana, sichere Konfiguration von Samba und Postfix, virtualisierte Umgebungen, Firewalls oder automatisierte Backups: mit den geprüften Setups kein Problem!



Michael Kofler nutzt Linux seit über 30 Jahren und vermittelt sein Fachwissen in erfolgreichen IT-Fachbüchern. Zu seinen Themengebieten zählen auch IT-Sicherheit, Python, Docker, Git und der Raspberry Pi. Er ist Entwickler, berät Firmen und arbeitet als Lehrbeauftragter.

Aus dem Inhalt

Grundlagen

Installation und Konfiguration
Arbeiten mit Desktops und Terminal
Linux auf dem Raspberry Pi
Prozesse, Rechte und User

Fortgeschrittene Techniken

Linux-Shells: Bash, Fish und ZSH
Software- und Paketverwaltung
Kernel und Module, systemd
Firewalls mit nftables

Netzwerk, Server, Sicherheit

Sichere Datei- und Webserver
Backups, SSH, Fail2Ban, SELinux und AppArmor
Monitoring: Prometheus & Grafana
Virtualisierung: Docker und Podman, KVM, WSL2, VirtualBox
Die eigene KI: Ollama und llama.cpp

