

# Kapitel 5

## Betriebssystemgrundlagen

*In einem geschlossenen System nimmt das Chaos mit der Zeit zu.  
– Zweiter Hauptsatz der Thermodynamik*

Damit Anwendungsprogramme auf Computern laufen können, ist ein Betriebssystem erforderlich. Dieses Kapitel beschreibt die historischen und theoretischen Grundlagen dieser Systeme. Der praktische Einsatz verschiedener Systeme wird in Kapitel 6, »Windows«, Kapitel 7, »Linux«, und Kapitel 8, »macOS«, erläutert.

Das Betriebssystem ist das grundlegende Computerprogramm. Es steuert die Hardware, koordiniert die Ressourcenzugriffe der Anwendungsprogramme und stellt dem Benutzer Steuerungsmöglichkeiten zur Verfügung. Im Einzelnen erfüllen Betriebssysteme vor allem die folgenden Aufgaben:

- ▶ *Prozessmanagement.* Die Ressourcen des Computersystems müssen zwischen den verschiedenen laufenden Programmen und Systemaufgaben verteilt werden. Zu diesem Zweck werden die einzelnen Aufgaben als sogenannte *Prozesse* ausgeführt, die vom Betriebssystem als übergeordnetem Steuerprozess verwaltet werden.
- ▶ *Speichermanagement.* Obwohl Computersysteme heutzutage über einen vergleichsweise großen Arbeitsspeicher verfügen, finden dennoch oftmals nicht alle Programme und Daten auf einmal Platz darin. Das Speichermanagement sorgt dafür, dass immer die gerade benötigten Speicherinhalte zur Verfügung stehen, ohne dass die Programmierer der Anwendungssoftware sich sonderlich darum kümmern müssten.
- ▶ *Steuerung und Abstraktion der Hardware.* Computersysteme sind hochgradig modular aufgebaut; jede Aufgabe kann durch viele verschiedene Geräte unterschiedlicher Hersteller erledigt werden. Betriebssysteme lösen dieses Problem durch den Einsatz der sogenannten *Gerätetreiber*, die die Steuerlogik für bestimmte Hardware enthalten. Vor dem Programmierer werden die konkreten Einzelheiten bestimmter Geräte verborgen, weil es vollkommen unzumutbar wäre, beim Schreiben eines Anwendungsprogramms auf die Besonderheiten Hunderte möglicher Geräte einzugehen.
- ▶ *Ein- und Ausgabesteuerung.* Computerprogramme sind auf die Eingabe von Daten angewiesen, ihre Benutzer erwarten die Ausgabe von Ergebnissen. Betriebssysteme steuern die Zusammenarbeit mit vielen verschiedenen Ein- und Ausgabekanälen wie Tastatur und Bildschirm, Datenträgern oder Netzwerken.

- ▶ **Dateiverwaltung.** Programme und Daten müssen auf einem Computer dauerhaft gespeichert werden, weil der Arbeitsspeicher zu klein ist und vor allem weil sein Inhalt beim Ausschalten verloren geht. Aus diesem Grund werden Daten in Form von *Dateien* auf Datenträgern wie Festplatten, CDs, DVDs oder USB-Sticks gespeichert; auch Netzwerkordner oder Cloud-Speicher im Internet werden auf ähnliche Weise angesteuert. Die Logik der Dateiverwaltung wird in Form sogenannter *Dateisysteme* vom Betriebssystem zur Verfügung gestellt, damit alle Programme auf dieselbe Art und Weise darauf zugreifen.
- ▶ **Bereitstellen der Benutzeroberfläche.** Ein Spezialfall der Ein- und Ausgabesteuerung besteht im Bereitstellen der Benutzeroberfläche. Das Betriebssystem selbst und alle Programme müssen mit dem Benutzer kommunizieren, um nach dessen Wünschen Aufgaben zu erledigen. Es existieren zwei Arten von Benutzeroberflächen: Die *Konsole* ermöglicht das dialogbasierte Arbeiten; Benutzer geben per Tastatur Befehle ein und erhalten Antworten. Die *grafische Benutzeroberfläche* ermöglicht dagegen die intuitive Erledigung von Aufgaben über das Aktivieren von Schaltflächen, Menüs und Symbolen mit der Maus. So gut wie alle modernen Betriebssysteme besitzen unterschiedlich leistungsfähige Varianten beider Arten von Oberflächen.
- ▶ **Netzwerkzugriff.** In modernen Betriebssystemen ist der Zugriff auf Netzwerke und das Internet ebenfalls ein integraler Bestandteil. Auch hier ist die Hardwaresteuerung einer der Aspekte, während der Datenaustausch über das Netzwerk zur Ein- und Ausgabe gehört. Dieser Themenkomplex wurde bereits im vorigen Kapitel ausführlich behandelt.

## 5.1 Entwicklung der Betriebssysteme

In diesem Abschnitt wird kurz die Entstehung und Weiterentwicklung der Betriebssysteme nachgezeichnet. Betriebssysteme im heutigen Sinn wurden erst erforderlich, als Menschen begannen, direkt mit Computern zu kommunizieren.

### 5.1.1 Die Vorgeschichte

Die allerersten Computersysteme der 40er- und 50er-Jahre des letzten Jahrhunderts wurden über Schalter und Steckverbindungen programmiert. Dem einzigen Programm, das zu einer bestimmten Zeit auf dem Computer lief, standen sämtliche Ressourcen ungeteilt zur Verfügung. Diese Ressourcen waren den Programmierern unmittelbar bekannt, es musste keine Abstraktion der Hardwareressourcen stattfinden. Solche Rechner besaßen überhaupt kein Betriebssystem.

Als die Lochkarten Einzug in die Rechenzentren hielten, war es üblich, dass ein Programmierer dem Operator den mithilfe eines mechanischen Geräts gestanzten Kartenstapel übergab. Die Programmierer selbst bekamen den eigentlichen Computer möglicherweise nicht einmal zu Gesicht, weil nur die mit weißen Kitteln bekleideten Operatoren das »Allerheiligste«

betreten durften. Der Operator war dafür verantwortlich, den Inhalt des Lochkartenstapels in den Computer einzulesen, und händigte dem Programmierer einen Stapel Endlospapier mit den Ergebnissen aus, falls das Programm keine Fehler enthielt.

Erst in den 60er-Jahren wurden bestimmte häufig verwendete Programmteile oder Hilfsmittel wie Interpreter für höhere Programmiersprachen auf Magnetbändern statt auf Lochkarten abgespeichert. In der ersten Zeit waren wieder die Operatoren dafür zuständig, das richtige Band einzulesen, um die Programmlochkarten eines bestimmten Programmierers korrekt zu verarbeiten.

Allmählich wurden übergeordnete Steuerprogramme eingeführt, die in der Lage waren, auf Lochkarten mit speziellen Metabefehlen zu reagieren, die nicht zum Programm selbst gehörten, sondern organisatorische Informationen enthielten. Damit konnten einige der Aufgaben von Operatoren automatisiert werden, sodass die Steuerprogramme als *Operating Systems* bezeichnet wurden – noch heute das englische Wort für Betriebssystem. Diese allerersten Systeme ermöglichen die automatisierte Abarbeitung mehrerer Lochkartenstapel; diese Form der Verarbeitung wird deshalb als *Stapelverarbeitung (Batch Processing)* bezeichnet. Das Verfahren kommt auch heute noch zum Einsatz, obwohl es sich längst nicht mehr um einen Stapel von Lochkarten, sondern um einen Stapel von Befehlen handelt.<sup>1</sup>

»Richtige« Betriebssysteme, wie sie bis heute verwendet werden, wurden allerdings erst eingeführt, als die *Dialogverarbeitung* entwickelt wurde. Seit Mitte der 60er-Jahre wurden immer mehr Rechenzentren mit *Terminals* ausgestattet. Ein Terminal (wegen seiner fehlenden eigenen Rechenfähigkeiten auch »dummes Terminal« – *dumb terminal* – genannt) ist eine Ein- und Ausgabeeinheit, die direkt am Arbeitsplatz eines Programmierers steht und mit dem eigentlichen Computer verbunden ist. Wie bereits in Kapitel 1, »Einführung«, erwähnt, waren die ersten Terminals umgebauten Fernschreiber mit Schreibmaschinentastatur und Ausgabe auf Endlospapier. Später wurden sie durch Einheiten aus ähnlichen Tastaturen und monochromen Bildschirmen ersetzt beziehungsweise ergänzt. Heute sind Terminals, oder genauer gesagt *Terminal-Emulationen*, Softwareprogramme für die verschiedenen Unix-Systeme.

Das Betriebssystem muss die Eingabe des Benutzers in den Computer transportieren und die Antwort des Computers an das Terminal zurückmelden. Die besondere Herausforderung für die Entwickler der frühen Betriebssysteme bestand darin, die Benutzer mehrerer Terminals zeitgleich zu bedienen. Die damaligen Computersysteme waren nämlich erheblich zu selten und zu teuer, um einem einzelnen Benutzer sämtliche Ressourcen zu überlassen.

Aus dieser Anforderung entwickelte sich das sogenannte *Timesharing*-Verfahren, das die Anfragen mehrerer Benutzer scheinbar gleichzeitig verarbeitet, indem es die Rechenzeit in kleine Einheiten (Zeitscheiben, *Time Slices*) unterteilt und den Anforderungen der einzelnen Benutzer der Reihe nach zuweist. Aus diesem grundsätzlichen Prinzip entwickelte sich bei-

<sup>1</sup> In der klassischen Windows-Konsole, einem Erbstück aus Microsofts erstem Betriebssystem MS-DOS, haben Skripte deshalb die Dateiendung .bat (kurz für batch).

spielsweise das moderne Multitasking, das die Ausführung mehrerer Programme auf demselben Rechner ermöglicht. Außerdem wurden allmählich Mechanismen eingeführt, um die Arbeit verschiedener Benutzer im Speicher des Rechners voneinander zu trennen. Die Grundlagen der Zugriffsrechte, der persönlichen Anmeldung und des Speicher- und Resourcenmanagements wurden damit entwickelt.

Die ersten Timesharing-Betriebssysteme wurden für einzelne Computer und ihre Anwender in Universitäten und anderen großen Institutionen hergestellt. Eines der wenigen frühen Beispiele, die relativ bekannt geworden sind, ist das am MIT (Massachusetts Institute of Technology) entwickelte *ITS (Incompatible Timesharing System)* – es wurde von Richard M. Stallman mitentwickelt, dem Begründer der Idee der freien Software.<sup>2</sup> Einige der Fähigkeiten von ITS wurden später in Unix aufgenommen.

Der erste Versuch, ein kommerzielles Timesharing-System zu schreiben, erfolgte Ende der 60er-Jahre: Die Bell Laboratories, das Entwicklungszentrum der US-Telefongesellschaft AT&T, arbeiteten mit einigen anderen Firmen an einem System namens *MULTICS*. Die Idee war, viele Hundert Terminals an einem für damalige Verhältnisse recht mächtigen Rechner anzuschließen. Leider wurde MULTICS erst viel später fertiggestellt, als es kaum noch jemanden interessierte.

### 5.1.2 Die Geschichte von Unix

Einige Ideen von MULTICS inspirierten jedoch *Ken Thompson* von den Bell Labs dazu, mit der Arbeit an einem eigenen Betriebssystem zu beginnen. Anfangs war es als Einzelplatzsystem geplant und wurde auf einem PDP-7-Kleincomputer von Digital Equipment (DEC) entwickelt, der, verglichen mit der für MULTICS verwendeten Maschine, nicht besonders leistungsfähig war. Thompson und sein späterer Mitstreiter *Dennis Ritchie* nannten ihr Betriebssystem in einer Anspielung auf MULTICS *Unics*; nach einiger Zeit wurde daraus *Unix*. Die ursprünglich nicht geplante Mehrbenutzerfähigkeit wurde übrigens sehr schnell nachgerüstet, nachdem die ersten Versionen von Unix gut funktionierten.

Eines der wichtigsten Designmerkmale des Betriebssystems Unix ist seine Modularität: Jeder (als eigenständiges Programm realisierte) Befehl und jeder Bestandteil des Systems kann einzeln ausgetauscht werden, ohne das Gesamtsystem ändern zu müssen. Unix wurde zunächst in der Maschinensprache der PDP-7 programmiert. Ab 1971 entwickelten Dennis Ritchie und *Brian Kernighan* dann die Programmiersprache C, in der Unix schließlich neu geschrieben wurde. Da C-Compiler bald für verschiedene Computersysteme verfügbar waren, fand Unix schnell Verbreitung.

Eine kommerzielle Verbreitung von Unix war anfangs nicht möglich. Da AT&T in den 70er-Jahren noch das US-Telekommunikationsmonopol innehatte, durfte das Unternehmen

<sup>2</sup> Zuvor hatte es ein – inzwischen längst in Vergessenheit geratenes – System namens *Compatible Time-Sharing System (CTSS)* gegeben, auf das sich die Namenswahl bezog.

keine Geschäfte in anderen Branchen wie etwa dem Computerbereich tätigen. Deshalb wurden Lizenzen für den Unix-Quellcode unter sehr lockeren Bedingungen an Universitäten vergeben. Das ist der Grund dafür, dass die Grundlagen des Betriebssystemaufbaus bis heute vornehmlich am Beispiel von Unix und seinen Nachfolgesystemen gelehrt werden.

Die aktivste aller Universitäten, die den Unix-Quellcode erhielten, war die University of California in Berkeley. Die dortigen Programmierer erweiterten das ursprüngliche Unix und brachten schließlich eine eigene Version heraus, die *Berkeley System Distribution (BSD)*. Als AT&T schließlich in den 80er-Jahren die Lizenz erhielt, Unix kommerziell zu vertreiben, hatten sich die AT&T-Version, genannt *System V*, und die BSD bereits erheblich auseinanderentwickelt. Zudem wurde die Berkeley University verpflichtet, sämtliche Bestandteile des AT&T-Unix aus der BSD zu entfernen.

Seitdem ist jede kommerzielle und jede freie Unix-Version eine mehr oder weniger starke Mischung aus BSD- und System-V-Features. Allerdings haben System V und BSD wieder zahlreiche Fähigkeiten voneinander übernommen, sodass es nicht immer ganz leicht ist, sie auseinanderzuhalten.

Heute existiert eine Vielzahl unterschiedlicher Unix-kompatibler Betriebssysteme. Dazu gehören oder gehörten kommerzielle Systeme wie *Sun Solaris*<sup>3</sup>, *IBM AIX*, *HP UX* und freie Varianten wie etwa *Linux* oder *FreeBSD*. Eine Sonderstellung nimmt Apples Betriebssystem *macOS* (vormals Mac OS X, zwischenzeitlich nur OS X) ein: Es basiert auf einer Version der BSD, der Betriebssystemkern *Darwin* ist Open-Source-Software und läuft auf verschiedenen Plattformen. Die grafische Benutzeroberfläche *Aqua* ist dagegen eine kommerzielle Eigenproduktion von Apple und funktioniert nur auf Macs mit Intel-Prozessoren<sup>4</sup>.

Die Mindestanforderung dessen, was ein Unix-System leisten muss, ist in einem Standard namens *POSIX* (*Portable Operating System Interface*) festgeschrieben. Allerdings werden von einem »richtigen« Unix heute auch einige weitere Quasistandards verlangt, die nicht im POSIX-Standard festgelegt sind. Am wichtigsten ist dieser Standard für Programmierer. Wenn Sie ein Programm POSIX-konform schreiben, können Sie davon ausgehen, dass es sich unter jeder beliebigen Unix-Version kompilieren lässt.

Anfang der 90er-Jahre sah es übrigens bereits fast so aus, als würde Unix nicht mehr lange überleben: Die verschiedenen Varianten entwickelten sich zunehmend auseinander. Auf dem Desktop dominierte Microsoft Windows, während Server für die immer häufiger eingesetzten PC-Netzwerke vor allem unter Novell NetWare betrieben wurden.

Zwei wichtige Umstände haben Unix gerettet und machen es heute, über 40 Jahre nach seiner Entwicklung, zu einem der gefragtesten Betriebssystemkonzepte: Der eine Grund ist die immense Ausbreitung des Internets, dessen wichtigste Konzepte unter Unix entwickelt wur-

<sup>3</sup> Später wurde auch eine Open-Source-Variante namens *OpenSolaris* entwickelt, die heute jedoch keine große Rolle mehr spielt (das letzte Update fand 2009 statt).

<sup>4</sup> Der Support für PowerPC-Prozessoren, auf denen Mac-Rechner bis 2005 basierten, wurde bereits vor etlichen Versionen des damals noch Mac OS X genannten Systems aufgegeben.

den. Der zweite Anlass für die Verbreitung von Unix ist die Erfolgsgeschichte des freien Betriebssystems Linux, das 1991 von dem finnischen Informatikstudenten *Linus Torvalds* entwickelt wurde und inzwischen das Betriebssystem mit dem größten jährlichen Wachstum ist. Mehr über die Entwicklung von Linux erfahren Sie im weiteren Verlauf dieses Kapitels.

### 5.1.3 PC-Betriebssysteme

Eine völlig andere Entwicklungslinie im Bereich der Betriebssysteme begann mit der Entwicklung der Personal Computer ab Mitte der 70er-Jahre. Auf das Timesharing, den Mehrbenutzerbetrieb und selbst Multitasking wurde bei den ersten Versionen von PC-Betriebssystemen zugunsten der Performancesteigerung zunächst völlig verzichtet.

Der erste weitverbreitete PC war der von *Steve Wozniak* und *Steve Jobs* 1977 entwickelte Apple II. Er erlaubte es, an einer wenig komfortablen Konsole BASIC- und Systembefehle einzutippen. Stellte man einem Befehl eine Zeilenummer voran, wurde er Teil eines im Arbeitsspeicher abgelegten Programms, das beispielsweise mithilfe von RUN gestartet und über LIST auf dem Bildschirm angezeigt werden konnte. Befehle ohne Zeilenummer wurden dagegen unmittelbar ausgeführt.

Dieses Konzept wurde in den 80er-Jahren von unzähligen Homecomputern wie dem Commodore C64, dem Atari 800XL oder dem Sinclair ZX81 und ZX Spectrum nachgeahmt. Die in das ROM eingebauten BASIC-Editoren und -Interpreter lassen sich allerdings nicht als vollwertige Betriebssysteme bezeichnen.

Das erste »richtige« Betriebssystem für Personal Computer wurde von *Gary Kildall* entwickelt. Es hieß *CP/M* (*Control Program for Microcomputers*). Im Vergleich zu den bereits weit fortgeschrittenen Betriebssystemen für Großrechner und Kleincomputer war es ziemlich primitiv, hatte aber mehrere Vorteile: Es lief auf zwei verschiedenen, zu jener Zeit weitverbreiteten Prozessorplattformen, dem Intel 8080 und dem Zilog Z80. Außerdem besaß es eine Dateiverwaltungsfunktion für Diskettenlaufwerke und konnte über verschiedene Konsolebefehle gesteuert werden.

### DOS und Windows

1981 kam dann der IBM-PC auf den Markt, der bald zum erfolgreichsten Personal Computer wurde. IBM wollte ein CP/M-ähnliches Betriebssystem für diesen Rechner haben. Gary Kildall war jedoch nicht bereit, IBMs Verschwiegenheitserklärungen zu unterschreiben. Deshalb wandte sich IBM an die junge Firma Microsoft, die in Seattle arbeitete und Programmiersprachenpakete für verschiedene PCs produzierte. *Bill Gates*, der die Firma 1975 mit seinem Schulfreund *Paul Allen* gegründet hatte, fand bei einem Unternehmen namens *Seattle Computer Products* einen halb fertigen CP/M-Nachbau namens *QDOS – Quick and Dirty Operating System*. Angeblich kaufte Bill Gates alle Rechte an diesem System für 50.000 US\$.

Microsoft verbesserte das System und lieferte es als *MS-DOS* (*Microsoft Disk Operating System*) an IBM. Eine leicht modifizierte Fassung wurde unter dem Namen *IBM PC DOS* mit jedem IBM-PC ausgeliefert. Die meisten später erschienenen kompatiblen Geräte enthielten dann ab Werk das eigentliche MS-DOS. Auf diese Weise entstand das bis heute andauernde Kerngeschäft von Microsoft, denn weltweit wird noch immer fast jeder neue PC mit einem Microsoft-Betriebssystem ausgeliefert.

CP/M, MS-DOS und einige andere Versuche besaßen allesamt nur eine Konsole, keine grafische Benutzeroberfläche. Dabei war diese bereits ab 1963, mitsamt der Maus als Eingabegerät, als Designstudie von *Douglas Engelbart* entwickelt worden. Im Forschungszentrum *Xerox PARC* wurde sie etwa Mitte der 70er-Jahre fertiggestellt. Allerdings waren die Manager von Xerox nicht in der Lage, die Entwicklungen der PARC-Forscher zu verstehen und zur Marktreife zu bringen. Erst Anfang der 80er-Jahre besichtigte *Steve Jobs* von Apple das PARC und verliebte sich auf den ersten Blick in die grafische Oberfläche. Daraufhin erschien im Jahr 1983 der teure Flop *Apple Lisa*; 1984 kam schließlich der *Macintosh* auf den Markt. Beide waren mit Maus und grafischer Oberfläche ausgestattet.

Auf den IBM-PCs hielt die grafische Benutzeroberfläche erst Jahre später Einzug. Da der Macintosh einen gewissen, wenn auch anfangs noch keinen überwältigenden Erfolg hatte, beschloss Bill Gates, dass auch der PC ein *Graphical User Interface (GUI)* erhalten sollte. 1990 erschien schließlich die erste brauchbare Version von *Microsoft Windows*, zunächst als grafischer Aufsatz für MS-DOS.

Ebenfalls im Jahr 1990 wurde der Prozessor Intel 80386 eingeführt. Es handelte sich um den ersten 32-Bit-Prozessor der Intel-Baureihe. Der IBM PC/AT und kompatible Geräte anderer Hersteller wurden bald damit ausgestattet. Ärgerlicherweise war MS-DOS jedoch ein reines 16-Bit-Betriebssystem, das bestimmte eingebaute Fähigkeiten des 386er wie Speicherschutz, Multitasking und die Adressierung von mehr Arbeitsspeicher nicht nutzen konnte.

In dieser Situation wurde bei IBM ein neues Betriebssystem konzipiert. Das 32-Bit-System OS/2 sollte zusammen mit dem neuartigen PC-System PS/2 verkauft werden. Dieses Computersystem sollte durch patentierte, inkompatible Schnittstellen die Hersteller IBM-kompatibler PCs ausbooten. Die Programmierung von OS/2 wurde zunächst in Zusammenarbeit mit Microsoft in Angriff genommen. Schließlich gerieten die beiden Firmen jedoch in Streit, weil IBM Windows nicht unterstützen wollte und voll auf OS/2 setzte. Letztlich setzte sich Windows als wichtigstes PC-Betriebssystem durch. Dennoch besaß OS/2, dessen bisher letzte Version Warp 4 im Jahr 1996 erschien, einige hervorragende Konzepte.

Microsoft blieb dagegen noch jahrelang bei DOS als Betriebssystem und Windows als grafischer Erweiterung, wobei DOS jedoch allmählich zum Windows-Starter verkam und Windows – zumindest teilweise – mit 32-Bit-Fähigkeiten ausgestattet wurde. Parallel entwickelte Microsoft ein völlig neues Betriebssystem namens *Windows NT (New Technology)*. Als Chefentwickler konnte Microsoft *David Cutler* gewinnen, der das Betriebssystem VMS für die

legendären VAX-Rechner von DEC entwickelt hatte. 1993 erschien die erste Version, die aufgrund der Windows-3.0-Benutzeroberfläche als Windows NT 3.0 bezeichnet wurde.

1995 kam Windows 95 auf den Markt, die Privatkundenversion eines 32-Bit-Windows-Systems mit modernisierter, stärker von Mac OS beeinflusster grafischer Oberfläche. Unter der Oberfläche war es noch immer ein DOS-Betriebssystem, das die Windows-Umgebung startete. Allerdings waren die meisten 32-Bit-Errungenschaften nun auch in Windows 95 verfügbar, beispielsweise die sehnsüchtig erwarteten langen Dateinamen. 1996 wurde Windows NT 4.0 vorgestellt, das die neue Oberfläche von Windows 95 auch in der NT-Welt einführt.

Erst das 2001 eingeführte Windows XP brachte schließlich auch in der Privatkundenwelt den Umstieg auf die NT-Basis. Das System wurde in einer Home- und einer Professional-Version angeboten.

Zuvor waren noch zwei Nachfolger von Windows 95 namens *Windows 98* und *Windows Me* (Millennium Edition) erschienen; das ursprünglich Windows NT 5.0 genannte System kam 1999 als Windows 2000 auf den Markt.

Anfang 2007 stellte Microsoft mit Windows Vista endlich den seit Langem entwickelten XP-Nachfolger bereit. Das System wurde in sieben verschiedenen Versionen verkauft, vom minimalen Vista Home Basic über mehrere Home- und Business-Varianten bis hin zum voll ausgestatteten Vista Ultimate.

Im Oktober 2009 erschien Windows 7, und 2012 wurde Windows 8 vorgestellt – beide ebenfalls in verschiedenen Editionen. In Windows 8 wurde erstmals versucht, eine einheitliche Bedienoberfläche für PCs, Tablets und Smartphones zu etablieren; unter anderem wurde der Desktop durch ein erweitertes Startmenü mit konfigurierbaren rechteckigen Kacheln ersetzt. Viele Benutzer hielten diesen ersten Versuch für nicht sonderlich gelungen, sodass das bald darauf erschienene Windows 8.1 einige der Änderungen wieder rückgängig machte: Genau wie bei der aktuellen Version Windows 10 wird hier nach dem Start erst einmal wieder der klassische Desktop angezeigt, und eine durch die Windows-8-Kacheln erweiterte Version des Startmenüs erscheint erst nach Klick auf die START-Schaltfläche.

Die Merkmale der diversen Windows-Versionen werden im nächsten Kapitel genauer erläutert.

## Linux

Der 386er-PC brachte nicht nur der Windows-Familie den Durchbruch, sondern war auch der Anlass für Linus Torvalds, das Betriebssystem Linux zu entwickeln. Er war ein wenig enttäuscht von den eingeschränkten Möglichkeiten des zu Lehrzwecken entwickelten abgespeckten Unix-Systems Minix und wollte auf keinen Fall Windows benutzen. Aus diesem Grund begann er mit der Entwicklung seines eigenen Systems.

Das Betriebssystem Linux ist ein voll ausgestattetes POSIX-konformes Unix-Betriebssystem, dessen Kernel allerdings von seinem Erfinder Linus Torvalds vollkommen neu entwickelt

wurde. Seit seiner ursprünglichen Entwicklung im Jahr 1991 wurde der Linux-Kernel durch die Mitarbeit zahlloser Freiwilliger immer weiter ausgebaut. Heute unterstützt er beinahe jede erdenkliche Hardware und läuft nicht etwa nur auf der Intel-PC-Architektur, sondern wurde auf viele verschiedene Plattformen portiert, beispielsweise PowerPC, Alpha, Sun SPARC sowie auf diverse IBM-Großrechner. In Form von Google Android läuft eine Variante auch auf zahllosen Smartphones und Tablets.

Selbstverständlich besteht ein Unix-System wie Linux aber nicht nur aus dem Kernel. Um damit arbeiten zu können, wird eine Benutzeroberfläche in Form einer Shell oder eines grafischen Desktops benötigt (alle aktuellen Linux-Distributionen bieten beides). Ein weiterer wichtiger Bestandteil eines Unix-Systems sind die zahlreichen Systemprogramme. Die Linux-Versionen dieser Programme sind kompatibel mit den entsprechenden Befehlen kommerzieller Unix-Versionen, entstammen aber größtenteils dem *GNU-Projekt* (<http://www.gnu.org>). Diese Abkürzung steht für *GNU's Not Unix*; es handelt sich um ein *rekursives Akronym*<sup>5</sup>, in dem der erste Buchstabe immer wieder für den Namen des Ganzen steht – eine beliebte Form der Unterhaltung in der Unix- und Open-Source-Szene. Aus diesem Grund müsste ein vollständiges Linux-System aus Kernel und Dienstprogrammen korrekterweise als *GNU/Linux* bezeichnet werden, aber daran hält sich nur die Distribution Debian.

Das GNU-Projekt wurde 1984 von *Richard Stallman* ins Leben gerufen und setzte sich zum Ziel, freie Versionen sämtlicher Unix-Systemprogramme zu entwickeln. Deshalb gründete Stallman gleichzeitig die *Free Software Foundation* (FSF, <http://www.fsf.org>). Als Linus Torvalds mit der Arbeit an Linux begann, existierte bereits ein C-Compiler (der *GNU C Compiler* oder *GCC*); auch ein Großteil der Standardbefehle waren bereits verfügbar. Die meisten GNU-Varianten der Unix-Programme sind leistungsfähiger als die ursprünglichen Versionen. Lediglich der lange geplante GNU-Kernel Hurd ist bis heute nicht als finale Version erschienen, zumal er durch die Entwicklung von Linux im Grunde überflüssig wurde.

Freie Software hat nicht nur etwas damit zu tun, dass die Programme kostenlos erhältlich sind, sondern auch damit, dass Sie den Quellcode erhalten und damit fast alles machen dürfen, was Sie möchten. Deshalb lautet ein Motto der Free Software Foundation »Free software is a matter of liberty, not price« (»Freie Software ist eine Frage der Freiheit, nicht des Preises«). Kommerzielle Softwarelizenzen enthalten nämlich in der Regel eine Reihe von Einschränkungen und erlauben im Grunde nichts weiter, als die Software für ihren offiziellen Anwendungszweck einzusetzen. Die FSF hat dafür eine eigene Softwarelizenz ausgearbeitet, die vor allem verhindern soll, dass kommerzielle Softwareentwickler freie Softwareprojekte an sich binden und die ursprüngliche Freiheit beeinträchtigen. Unter dieser Lizenz, der *GNU General Public License (GPL)*, wird übrigens auch Linux selbst verbreitet.

<sup>5</sup> Rekursion ist ein beliebtes Verfahren in der Programmierung, das Sie in Kapitel 9, »Grundlagen der Programmierung«, kennenlernen werden. Es geht darum, dass eine Funktion sich selbst aufruft, um ineinander verschachtelte Probleme zu lösen.

Zu Beginn seiner Entstehung war Linux vor allem im Kreis der Entwickler verbreitet, die daran arbeiteten. Es war schwierig, den Linux-Kernel zu installieren und die GNU-Versionen aller erforderlichen Systemprogramme zu beschaffen und miteinander zu koordinieren. In den ersten Jahren wurde Linux deshalb vornehmlich von Informatikern und interessierten Studenten weitergereicht, da die Universitäten bereits über Internetanbindungen verfügten, als der Rest der Welt noch nichts damit zu tun hatte.

Einige der Studenten, die mit Linux arbeiteten, begannen allmählich, den Kernel und die Systemprogramme zusammenzustellen und Installationsprogramme für diese Betriebssystempakete zu schreiben. Aus diesen ersten Bemühungen entstanden allmählich verschiedene *Linux-Distributionen*, die anfangs auf Disketten, später auf CD-ROMs, noch später auf DVDs und in den letzten Jahren vorwiegend als Download verbreitet wurden, oft zusammen mit mehr oder weniger ausführlichen gedruckten Dokumentationen. Firmen wurden gegründet, die solche Distributionen erstellten und zu vergleichsweise günstigen Preisen verkauften.

Beachten Sie, dass der Kauf einer Distribution nichts mit dem Erwerb kommerzieller Software gemeinsam hat: Sie dürfen die Software, die Sie erhalten, auf beliebig vielen Rechnern installieren und an alle Ihre Bekannten weitergeben. Die Distributoren erhalten das Geld nicht für die Software selbst oder für ein Nutzungsrecht daran, sondern lediglich für ihre Arbeit an Installationsprogrammen und Dokumentationen. Eine Distribution kann allerdings einzelne kommerzielle Programme enthalten, für die andere Bedingungen gelten. Beachten Sie die Beschreibung, die der Distributor mitliefert.

Heute ist eine Reihe verschiedener Distributionen erhältlich, die sich bezüglich ihres Umfangs, ihres Anwendungsschwerpunkts und ihres Preises unterscheiden. In Kapitel 7, »Linux«, finden Sie eine Übersicht über die wichtigsten Distributionen und ihre Bezugsquellen im Web.

## 5.2 Aufgaben und Konzepte

Dieser Abschnitt geht näher auf die eingangs skizzierten Hauptaufgaben eines Betriebssystems ein. Hier lernen Sie nicht, wie Sie mit einem bestimmten Betriebssystem arbeiten können (das steht in den nächsten drei Kapiteln), stattdessen erfahren Sie, was »unter der Haube« vorgeht.

In den folgenden Abschnitten wird jedes wichtige Konzept zunächst allgemein und theoretisch vorgestellt. Anschließend wird skizziert, wie zwei verbreitete Betriebssysteme das jeweilige Problem lösen: Linux und Windows. Für einige der dargestellten Sachverhalte sind Programmierkenntnisse von Vorteil, auch wenn in diesem Kapitel kein Quellcode enthalten ist. Wenn Sie mit der Programmierung überhaupt noch nicht vertraut sind, sollten Sie zunächst Kapitel 9, »Grundlagen der Programmierung«, und Kapitel 10, »Konzepte der Programmierung«, lesen.

### 5.2.1 Allgemeiner Aufbau von Betriebssystemen

Die Überschrift dieses Abschnitts ist eine recht kühne Übertreibung. Sie enthält ein Versprechen, das niemand halten kann: Selbstverständlich gibt es gar keine allgemeine Art und Weise, wie Betriebssysteme aufgebaut sind. Die verschiedenen Systemfamilien unterscheiden sich gerade durch ihren recht andersartigen Aufbau.

Dennoch soll kurz skizziert werden, aus welchen Bestandteilen Betriebssysteme aufgebaut sind beziehungsweise sein können, bevor in den folgenden Abschnitten auf die einzelnen Aufgaben eingegangen wird. Gewisse Grundbestandteile besitzt tatsächlich jedes Betriebssystem, denn alle Systeme müssen Computer verwalten, die bestimmte Gemeinsamkeiten aufweisen.

Beinahe jedes neuere Betriebssystem besteht aus dem Kernel, den mehr oder weniger fest zu diesem gehörenden Gerätetreibern, den Systemprogrammen, einer Schnittstelle für Anwendungsprogramme und einer Benutzeroberfläche.

#### Der Kernel

Der *Kernel* (das englische Wort für Kern, beispielsweise den Obstkern) ist das grundlegende Computerprogramm, das unmittelbar auf dem Prozessor des Rechners ausgeführt wird. Er läuft bis zum Herunterfahren des Systems permanent im Hintergrund und steuert alle anderen Betriebssystemkomponenten sowie den Start und den Ablauf der Anwendungsprogramme. Der Kernel initialisiert die Zusammenarbeit mit der Hardware, indem er die Treiber lädt und koordiniert. Aus einer technischen Perspektive können Sie sich vorstellen, dass der Kernel das einzige echte Programm ist, das permanent ausgeführt wird, während alle anderen Programme, die später geladen werden, nur Unterprogramme sind, die vom Kernel aufgerufen werden und die Kontrolle durch einen Rücksprung wieder abgeben.

Es gibt unterschiedlich konzipierte Kernels. Das ältere Kernelkonzept ist der sogenannte *monolithische Kernel*, der so viele Funktionen wie möglich selbst ausführt. Moderner ist das Konzept des *Mikrokernels*, der so wenig wie möglich selbst tut und die meisten Aufgaben an Prozesse delegiert, die im Benutzermodus laufen wie gewöhnliche Anwendungsprogramme.

Da Mikrokernels kleine, elegante Programme sind und die einzelnen Teile des Betriebssystems nur bei Bedarf im Speicher halten, müssten Betriebssysteme auf Mikrokernel-Basis theoretisch schneller und effizienter laufen als Systeme mit monolithischen Kernels. Allerdings wird dabei oft vergessen, dass der ständige Wechsel zwischen Benutzer- und Kernelmodus Zeit und Ressourcen verbraucht. Außerdem können auch monolithische Kernel inzwischen häufig von einem der entscheidenden Vorteile des Mikrokernels profitieren: Die meisten Gerätetreiber sind modular, können also je nach Bedarf geladen und wieder aus dem Speicher entfernt werden. Dies ist besonders wichtig für hot-plugging-fähige Schnittstellen wie USB, FireWire oder Bluetooth.