4. Auflage

Eduard Glatz

Betriebssysteme

Grundlagen, Konzepte, Systemprogrammierung

dpunkt.verlag

Eduard Glatz, Prof. Dr. sc., studierte an der ETH in Zürich Elektrotechnik und Betriebswissenschaften. Nach 16 Jahren Berufspraxis in der Industrie und in Ingenieurunternehmungen wurde er 1996 an die Hochschule für Technik (FH Ostschweiz) in Rapperswil berufen, wo er Betriebssystemtheorie unterrichtete. Seit 2017 ist er privatwirtschaftlich u.a. im Kurswesen tätig.

Eduard Glatz

Betriebssysteme

Grundlagen, Konzepte, Systemprogrammierung

4., überarbeitete und aktualisierte Auflage

dpunkt.verlag

Eduard Glatz
eduard@glatz.name

Lektorat: Christa Preisendanz
Copy-Editing: Ursula Zimpfer, Herrenberg
Herstellung: Stefanie Weidner, Frank Heidt
Umschlaggestaltung: Helmut Kraus, *www.exclam.de*
Druck und Bindung: mediaprint solutions GmbH, 33100 Paderborn

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über *http://dnb.d-nb.de* abrufbar.

ISBN:
Print 978-3-86490-705-0
PDF 978-3-96088-839-0
ePub 978-3-96088-840-6
mobi 978-3-96088-841-3

4., überarbeitete und aktualisierte Auflage 2019

Wieblinger Weg 17
69123 Heidelberg

Hinweis:
Dieses Buch wurde auf PEFC-zertifiziertem Papier aus nachhaltiger Waldwirtschaft gedruckt. Der Umwelt zuliebe verzichten wir zusätzlich auf die Einschweißfolie.

PEFC zertifiziert
Das Papier für dieses Buch stammt aus nachhaltig bewirtschafteten Wäldern und kontrollierten Quellen.
PEFC™
PEFC/04-31-0810
www.pefc.de

Schreiben Sie uns:
Falls Sie Anregungen, Wünsche und Kommentare haben, lassen Sie es uns wissen: *hallo@dpunkt.de*.

5 4 3 2 1 0

Vorwort zur 4. Auflage

Betriebssysteme unterliegen einer stetigen Weiterentwicklung, die der Produktoptimierung hinsichtlich angebotener Dienste, Stabilität und Sicherheit dient. Darüber hinaus werden neue vielversprechende Technologien dienstbar gemacht, sofern sie eine gewisse Reife erreicht haben. Dies wurde bei der vierten Auflage angemessen berücksichtigt. Neu beschrieben sind beispielsweise NAS- (Network Attached Storage) und SAS-Systeme (Server Attached Storage), die dazu zählenden RAID-Laufwerksverbunde, Container-Systeme (z.B. Docker) und Unikernels. Ein näher beim Systemkern liegender Dienst betrifft den unter Linux zentralen Completely Fair Scheduler (CFS), der in seiner Funktionsweise erklärt wird. Nicht zu vernachlässigen sind thematische Ergänzungen bei der nebenläufigen Verarbeitung, wie etwa die Korrektheitsbedingungen der Parallelität, das Thread-Pool-Konzept und die Windows Services. Die bei Betriebssystemen nötige Konfigurationsdatenhaltung, realisiert mithilfe von Textdateien in der Unix-Welt, wird der Lösung mit einer Registrierungsdatenbank (Windows Registry) gegenübergestellt. Buchinhalte früherer Auflagen, die an Bedeutung verloren haben bzw. überholt sind, wurden entfernt oder auf die Buch-Webseite *(http://buch.novavia.ch)* ausgelagert. Abschließend möchte ich mich bei allen bedanken, die zu den bestehenden und neuen Themen beigetragen oder in anderer Form dieses Buch unterstützt haben.

Eduard Glatz
Urdorf, im Juli 2019

Vorwort zur 3. Auflage

Seit der Erstauflage dieses Lehrbuches, die Ende 2005 entstand, hat das Thema der Betriebssysteme nichts an Aktualität verloren. Vielmehr sind Betriebssysteme als Basissoftware von Smartphones und Embedded Systems fortlaufend in neue Anwendungsbereiche vorgestoßen. Entsprechend stellen Betriebssysteme einen obligatorischen Bestandteil einer Informatikausbildung auf Hochschulstufe dar. So gelten eine Mehrheit der in diesem Buch erfassten Themen in den von ACM und IEEE im Dezember 2013 aktualisierten *Computer Science Curricula* als *Core-Tier-1- und -Tier-2*-Inhalte, d.h., als Kernthemen eines Informatik-Bachelorstudiums.

Dieses Buch nutzt zwei Betrachtungswinkel: Einerseits ist dies die Sicht auf die Programmierschnittstelle eines Betriebssystems, also die Blackbox-Betrachtung des Softwareentwicklers. Andererseits werden die dahinter steckenden Prinzipien, Algorithmen und Mechanismen beschrieben, was der Whitebox-Betrachtung des Ingenieurs entspricht. Ergänzend werden ein paar Grundlagen der Prozessortechnik vermittelt. Diese helfen die Schnittstelle zwischen Betriebssystem und Hardware besser zu verstehen. Als Beispiele dienen die Betriebssysteme Windows und Unix, deren kombinierte Kenntnis heute eine wichtige Anforderung der Praxis darstellt. Als Hilfe für Dozierende stehen auf der Buch-Website `http://unix.hsr.ch` Übungsaufgaben mit Lösungen, alle Abbildungen des Buches und etliche Vorlesungfolien in elektronischer Form zur Verfügung.

Für die aktuelle Auflage wurde das Buch gesamthaft überarbeitet, teilweise neu gegliedert und zu Beginn jedes Kapitels mit kompetenzorientierten Lernzielen versehen. Die gestiegene Bedeutung der Mobilbetriebssysteme und der Rechnervirtualisierung wurde gebührend berücksichtigt, indem diese Themen nun in eigenen Kapiteln behandelt werden. Einen Einblick in mögliche zukünftige Systemarchitekturen gibt die Darstellung der aktuellen Forschung bzw. deren Erkenntnisse. Abschließend bedanke ich mich bei allen Personen, die mir bei der Realisierung dieses Buchprojektes geholfen haben.

Eduard Glatz
Urdorf, im Dezember 2014

Vorwort zur 1. und 2. Auflage

Obwohl heute Middleware-Systeme zur Verfügung stehen, die in vielen Fällen unabhängig von einem darunter laufenden Betriebssystem sind, ist die Betriebssystemthematik aus der Informatik-Grundausbildung aus verschiedenen Gründen nicht wegzudenken. Ohne Kenntnisse der hauptsächlichen Strukturen, Mechanismen und der Programmierschnittstelle eines Betriebssystems ist es nicht möglich, spezielle Systemdienste zu benutzen oder die Effizienz eines Systems zu optimieren. Es ist ein altes Ideal des Software Engineering, dass die hinter einer Schnittstelle stehende Implementierung als solche unwichtig ist und daher ignoriert werden soll. Auf Betriebssysteme bezogen würde dies heißen, nur die Programmierschnittstelle zu betrachten. Leider sind die dazugehörigen Beschreibungen praktisch immer minimal und befassen sich kaum mit den Konzepten der dahinter stehenden Implementierungen. Dies erschwert nicht nur ein tieferes Verständnis der Systemdienste, sondern kann auch zu einer inadäquaten Nutzung in komplexen Applikationen führen. Dieses Buch will Studierenden der Informatik und weiteren interessierten Personen die Grundlagen der Betriebssystemtheorie aus einer praktischen Perspektive näher bringen. Damit ist gemeint, dass nicht nur die Prinzipien von Betriebssystemen, sondern auch deren Nutzung bei der systemnahen Programmierung aufgezeigt werden. Dieser Ansatz entspricht der Idee des Bachelor-Studiums, das eine Berufsbefähigung nach drei Studienjahren anstrebt und die forschungsorientierte Ausbildung in die Master- und Doktoratstufe verschiebt.

Methodisch wird ein Weg zwischen der Betrachtung anfallender Probleme und ihren Lösungen auf einer theoretischen und einer praktischen Basis beschritten. Der Praxisbezug orientiert sich an den zwei am meisten verbreiteten Systemwelten, nämlich Unix und Windows. Kenntnisse der Prozessortechnik werden keine vorausgesetzt. Wo nötig werden die wichtigsten Prozessorgrundlagen erklärt, soweit sie für das Verständnis des Betriebssystems und der systemnahen Programmierung hilfreich sind. Die zahlreichen Beschreibungen von Systemfunktionen dienen dazu, die Programmbeispiele genau zu verstehen. Dies ist im Zeitalter der Java-Programmierung umso wichtiger, da eine gründliche Ausbildung in

der Programmiersprache C nicht mehr vorausgesetzt werden kann, wenn auch C-Grundkenntnisse für dieses Buch notwendig sind. Im Weiteren verschaffen diese Beschreibungen einen Einstieg in die Systemdokumentationen, wie sie in Form der Unix-Handbuchseiten oder der MSDN-Beschreibungen (Microsoft Developer Network) für Windows vorliegen und erfahrungsgemäß für Neulinge nur schwer verständlich sind. Auf die Betrachtung von Computernetzen wird weitgehend verzichtet, da dieses Thema in einem Betriebssystembuch nur oberflächlich gestreift werden kann und ausgezeichnete Standardwerke zur Verfügung stehen. Dafür werden als Ergänzung die aktuellen Themen Multiprozessorsysteme, Handheld-Betriebssysteme und Virtualisierungstechnologien vorgestellt.

Als Hilfe für Dozierende stehen auf der Buch-Website `http://unix.hsr.ch` Übungsaufgaben mit Lösungen, alle Abbildungen des Buches und etliche Vorlesungsfolien in elektronischer Form zur Verfügung. Abschließend bedanke ich mich bei allen Personen, die mir bei der Realisierung dieses Buchprojektes geholfen haben.

Eduard Glatz
Urdorf, im August 2005

Neuheiten in der zweiten Auflage

Eine neue Auflage erlaubt nicht nur das Aktualisieren schnell veralteter Informationen, sondern auch das Einbringen von Erfahrungen und zusätzlichen Themen. Aktualisiert haben wir die Informationen zu Windows 7, Windows CE und Symbian OS. Die ausführlichen tabellarischen Beschreibungen zu den Systemfunktionen sind nun nicht mehr im Buch, sondern in einer PDF-Datei zusammengefasst auf der Buch-Website (http://unix.hsr.ch) verfügbar. Auf vielfältigen Wunsch wurde das Literaturverzeichnis ausgedehnt. Neue Themen betreffen u.a. die Linux-basierten Smartphone-Betriebssysteme Android, WebOS und Maemo, die Systemprogrammierung aus C++, Java und .NET-Sprachen, die Bauweise von SSD (Solid State Disks), spezielle Dateisystemtechnologien (Schattenkopie, Disk Scheduling) und die Vermeidung von Synchronisationsengpässen.

Eduard Glatz
Urdorf, im Februar 2010

Inhaltsverzeichnis

1 Einführung

Lernziele

- Sie erklären den Zweck und die Rolle eines modernen Betriebssystems.
- Sie erkennen wie Rechnerressourcen durch Applikationen genutzt werden, wenn sie das Betriebssystem verwaltet.
- Sie beschreiben die Funktionen eines aktuellen Betriebssystems in Bezug auf Benutzbarkeit, Effizienz und Entwicklungsfähigkeit.
- Sie erklären die Vorteile abstrakter Schichten und ihrer Schnittstellen in hierarchisch gestalteten Architekturen.
- Sie analysieren die Kompromisse beim Entwurf eines Betriebssystems.
- Sie erläutern die Architektureigenschaften monolithischer, geschichteter, modularer und Mikrokernsysteme.
- Sie stellen netzwerkfähige, Client/Server- und verteilte Betriebssysteme einander gegenüber und vergleichen diese.

Als Einstieg in das Thema legen wir fest, welchen Zwecken ein Betriebssystem dient, wie es sich als Begriff definieren lässt und wo es in einem Rechner einzuordnen ist. Danach diskutieren wir die Anforderungen an den Betriebssystementwurf, mögliche Architekturen und weiterführende Ideen aus der Forschung.

1.1 Zweck

Der Begriff »Betriebssystem« kann unterschiedlich aufgefasst werden. Beispielsweise über die Frage: Was leistet ein Betriebssystem? Zwei Grundfunktionen sind:

- *Erweiterte Maschine*: Das Betriebssystem realisiert von vielen Applikationen geichartig genutzte Teilfunktionen als standardisierte Dienste. Damit wird die Applikationsentwicklung einfacher als beim direkten Zugriff auf die blanke

Rechnerhardware. Die erweiterte Maschine ist eine Abstraktion der Hardware auf hohem Niveau und entspringt einer Top-down-Sicht.

- *Betriebsmittelverwalter*: Das Betriebssystem verwaltet die zeitliche und räumliche Zuteilung von Rechnerressourcen. Im Mehrprogrammbetrieb wird im Zeitmultiplex der Prozessor zwischen verschiedenen ablauffähigen Programmen hin und her geschaltet. Im Raummultiplex wird der verfügbare Speicher auf geladene Programme aufgeteilt. Ausgehend von den Ressourcen entspricht dies einer Bottom-up-Sicht.

Detaillierter betrachtet erfüllt ein Betriebssystem sehr viele Zwecke. Es kann mehrere oder sogar alle der folgenden Funktionalitäten realisieren:

- *Hardwareunabhängige Programmierschnittstelle*: Programme können unverändert auf verschiedenen Computersystemen ablaufen (auf Quellcodeebene gilt dies sogar für unterschiedliche Prozessorfamilien mit differierenden Instruktionssätzen).
- *Geräteunabhängige Ein-/Ausgabefunktionen*: Programme können ohne Änderung unterschiedliche Modelle einer Peripheriegeräteart ansprechen.
- *Ressourcenverwaltung*: Mehrere Benutzer bzw. Prozesse können gemeinsame Betriebsmittel ohne Konflikte nutzen. Die Ressourcen werden jedem Benutzer so verfügbar gemacht, wie wenn er exklusiven Zugriff darauf hätte.
- *Speicherverwaltung*: Mehrere Prozesse/Applikationen können nebeneinander im Speicher platziert werden, ohne dass sie aufeinander Rücksicht nehmen müssen (jeder Prozess hat den Speicher scheinbar für sich allein). Zudem wird bei knappem Speicher dieser optimal auf alle Nutzer aufgeteilt.
- *Massenspeicherverwaltung (Dateisystem)*: Daten können persistent gespeichert und später wieder gefunden werden.
- *Parallelbetrieb (Multitasking)*: Mehrere Prozesse können quasiparallel ablaufen. Konzeptionell stehen mehr Prozessoren zur Verfügung als in der Hardware vorhanden, indem versteckt vor den Anwendungen parallele Abläufe, soweit nötig, sequenzialisiert werden.
- *Interprozesskommunikation*: Prozesse können mit anderen Prozessen Informationen austauschen. Die Prozesse können dabei entweder auf dem gleichen Rechner ablaufen (lokal) oder auf verschiedenen Systemen (verteilt) ausgeführt werden.
- *Sicherheitsmechanismen*: Es können sowohl Funktionen für die Datensicherung, d.h. die fehlerfreie Datenverarbeitung, als auch Datenschutzkonzepte implementiert sein. Der Datenschutz kann zum Beispiel durch das explizite Löschen freigegebener Bereiche im Hauptspeicher und auf Plattenspeichern sicherstellen, dass empfindliche Informationen nicht in falsche Hände fallen. Die Zugangskontrolle zum Rechner (Anmeldedialoge, Benutzerverwaltung) dient ebenfalls dem Datenschutz.

- *Bedienoberflächen*: Moderne Betriebssysteme realisieren grafische Bedienoberflächen mit ausgeklügelten Bedienkonzepten, die Dialoge mit dem System und Anwendungen komfortabel gestalten. Ergänzend existieren Eingabemöglichkeiten für Kommandozeilenbefehle, die geübten Benutzern sehr effiziente Dialogmöglichkeiten, z.B. zur Systemadministration, anbieten.

Die geräteunabhängige Ein-/Ausgabe war eine der wichtigsten Errungenschaften bei der erstmaligen Einführung von Betriebssystemen. Früher war es notwendig, dass Applikationen die Eigenheiten der angeschlossenen Peripheriegeräte im Detail kennen mussten. Mit einem Betriebssystem stehen hingegen logische Kanäle zur Verfügung, die Ein-/Ausgaben über standardisierte Funktionen bereitstellen (siehe Abb. 1–1). Die logischen Kanäle werden häufig mittels sprechender Textnamen identifiziert.

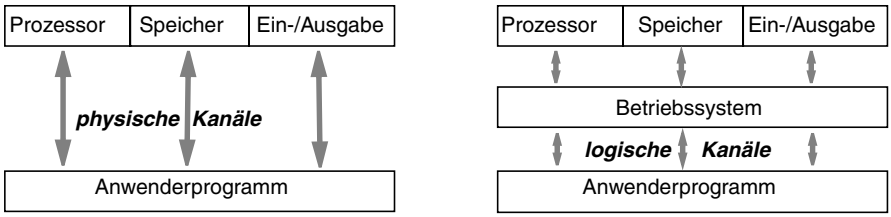

Abb. 1–1 *Ein-/Ausgabe ohne und mit Betriebssystem*

1.2 Definitionen

Leider existiert keine allgemein verbindliche Definition eines Betriebssystems. Welche Komponenten zu einem Betriebssystem gehören und welche nicht, lässt sich daher nicht endgültig festlegen. Nachfolgend sind drei unterschiedliche Definitionen stellvertretend vorgestellt, die dabei helfen, ein Betriebssystem zu charakterisieren. Eine erste, etwas schwer lesbare Definition nach DIN 44 300 beschreibt ein Betriebssystem wie folgt (Ausschnitt):

> *... die Programme eines digitalen Rechnersystems, die zusammen mit den Eigenschaften dieser Rechenanlage die Basis der möglichen Betriebsarten des Rechnersystems bilden und insbesondere die Abwicklung von Programmen steuern und überwachen.*

Eine zweite, der Literatur entnommene Definition lautet:

> *Ein Betriebssystem ist eine Menge von Programmen, welche die Ausführung von Benutzerprogrammen auf einem Rechner und den Gebrauch der vorhandenen Betriebsmittel steuern.*

Eine dritte Definition betrachtet das Betriebssystem als *Ressourcenverwalter*, wobei die Ressource hauptsächlich die darunter liegende Hardware des Rechners ist. Ein Computersystem lässt sich hierbei als eine strukturierte Sammlung von Ressourcenklassen betrachten, wobei jede Klasse durch eigene Systemprogramme kontrolliert wird (siehe Tab. 1–1).

	Zentrale Ressourcen	**Periphere Ressourcen**
Aktive Ressourcen	Prozessor(en)	Kommunikationseinheiten 1. Endgeräte (Tastaturen, Drucker, Anzeigen, Zeigegeräte etc.) 2. Netzwerk (entfernt, lokal) etc.
Passive Ressourcen	Hauptspeicher	Speichereinheiten 1. Platten 2. Bänder 3. CD-ROM/DVD etc.

Tab. 1–1 *Ressourcenklassen*

Ein Betriebssystem lässt sich auch mit einer Regierung (*government*) vergleichen. Wie diese realisiert das Betriebssystem keine nützliche Funktion für sich alleine, sondern stellt eine Umgebung zur Verfügung, in welcher andere Beteiligte nützliche Funktionen vollbringen können. Einige Autoren (z.B. K. Bauknecht, C. A. Zehnder) ziehen die Begriffe *Systemsoftware* bzw. *Systemprogramme* der Bezeichnung *Betriebssystem* vor. In diesem Sinne ist folgende Beschreibung dieser Autoren abgefasst:

»Die Systemprogramme, oft unter dem Begriff *Betriebssystem* zusammengefasst, lassen sich gemäß Abbildung 1–2 gruppieren.

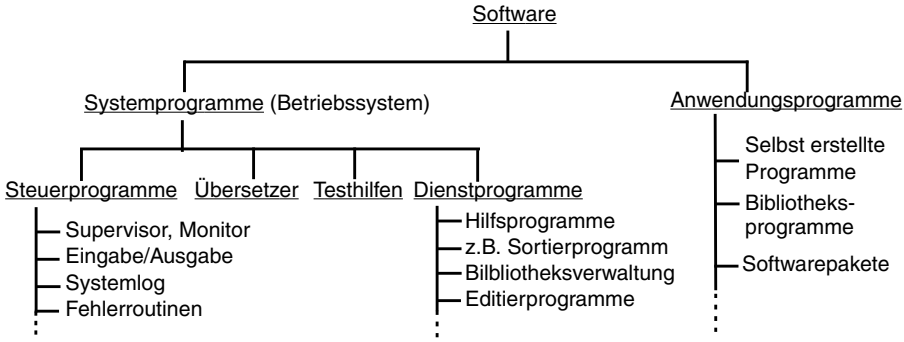

Abb. 1–2 *Softwaregliederung*

Die eigentlichen *Steuerprogramme* sind für folgende Funktionen zuständig:

- *Steuerung aller Computerfunktionen* und Koordination der verschiedenen zu aktivierenden Programme.

- *Steuerung der Ein-/Ausgabeoperationen* für die Anwendungsprogramme.
- *Überwachung und Registrierung* der auf dem Computersystem ablaufenden Aktivitäten.
- *Ermittlung und Korrektur* von Systemfehlern.«

Auffallend bei dieser Definition ist der Einbezug von *Übersetzern* (Compiler, Binder), *Testhilfen* und *Dienstprogrammen*. Für klassische Betriebssysteme (z.B. Unix und GNU-Tools) trifft dies vollumfänglich zu, während moderne Betriebssysteme oft die Bereitstellung von Übersetzungstools irgendwelchen Drittherstellern überlassen bzw. diese als separate Applikation ausliefern (z.B. Windows und Visual Studio).

1.3 Einordnung im Computersystem

In einem Rechner stellt das Betriebssystem eine Softwareschicht dar, die zwischen den Benutzerapplikationen einerseits und der Rechnerhardware andererseits liegt (siehe Abb. 1–3). Das Betriebssystem selbst besteht aus einem *Betriebssystemkern* und einer Sammlung von Programmen, die *Betriebssystemdienste* bereitstellen. Je nach Betrachtungsweise zählen dazu auch Programme zur Softwareentwicklung, wie Editoren und Compiler. Häufig wird nur der Betriebssystemkern als Betriebssystem bezeichnet, während der Begriff *Systemprogramme* für das Gesamtpaket inklusive der Programmentwicklungswerkzeuge benutzt wird.

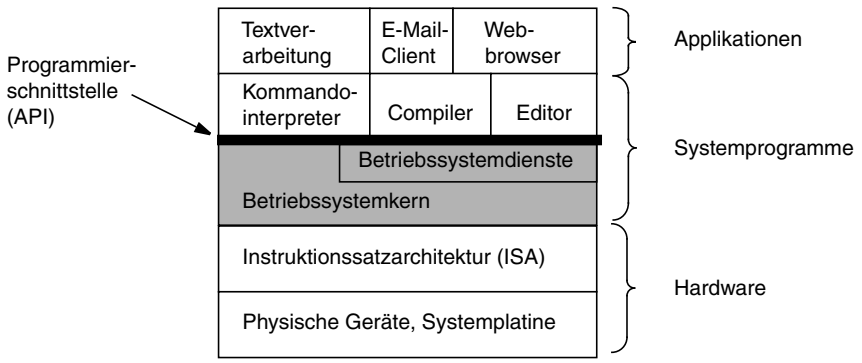

Abb. 1–3 *Schichtenmodell eines Rechners*

Das Betriebssystem setzt auf der Prozessorarchitektur auf, die durch einen Satz von Maschinenbefehlen und den Registeraufbau charakterisiert wird (sog. Instruktionssatzarchitektur, ISA). Die Systemplatine mit all ihren Bausteinen und den angeschlossenen Peripheriegeräten stellt die Arbeitsumgebung des Prozessors dar. Diese muss ebenfalls dem Betriebssystem in all ihren Details bekannt sein. Von zentraler Bedeutung für den Softwareentwickler ist die Programmierschnittstelle des Betriebssystems (*Application Programming Interface, API*). Die dort

zur Verfügung gestellte Funktionalität kann in Benutzerapplikationen eingesetzt werden. Aus Anwendungssicht unterscheiden sich Betriebssysteme in der *Programmierschnittstelle*, in den unterstützten *Dateiformaten für ausführbare Dateien,* im *Funktionsumfang*, in der *Bedienoberfläche* und der *Maschinensprache*, in die ihr Code übersetzt wurde. Zudem kann oft der Funktionsumfang, d.h. die installierten Systemteile, während des Installationsvorgangs unterschiedlich gewählt werden.

Wie bereits erwähnt, setzt das Betriebssystem direkt auf der Rechnerhardware auf und muss diese daher genau kennen. Denn es verwaltet folgende Hardwareelemente:

- Prozessor
- Arbeitsspeicher (*main memory*)
- Massenspeicher (*mass storage*), z.B. Festplatten, CD-ROM, DVD
- Benutzerschnittstelle (*user interface*)
- Kommunikations- und andere Peripheriegeräte (LAN, WLAN usw.)

Die Betriebssystemtheorie beruht damit auf den Prinzipien der Computertechnik. Computertechnik befasst sich mit:

1. Rechner-Grundmodellen (Von-Neumann-, Harvard-Architektur)
2. Funktionsweise des Prozessors (Instruktionssatz, Registeraufbau)
3. Speichern und ihren Realisierungen (Primär- und Sekundärspeicher)
4. Peripheriegeräten (Tastatur, Bildschirm, Schnittstellenbausteine usw.)

Um die hardwarenahen Teile des Betriebssystems oder nur schon den exakten Ablauf der Programmausführung zu verstehen, ist es daher unerlässlich, sich mit ein paar Details der Computertechnik zu befassen. Einige computertechnische Funktionsweisen, soweit sie für das Verständnis des Betriebssystems nötig sind, werden an passenden Stellen im Buch erklärt. Für weiter gehende Realisierungsdetails der Hardwareelemente sei auf entsprechende Spezialliteratur verwiesen.

1.4 Betriebssystemarten

Ein Betriebssystem stellt eine Umgebung zur Verfügung, in der Anwendungsprogramme ablaufen können. Eine Ablaufumgebung kann recht unterschiedlich realisiert sein:

- Als Laufzeitsystem (*Run-Time System*) einer Programmiersprache (ADA, Modula-2)
- Als virtuelle Maschine zur Ausführung eines Zwischencodes (z.B. Java Virtual Machine, .NET Common Language Runtime)
- Als Basisprogramm eines Rechners (z.B. Unix, Windows)

- Als (sprachunabhängige) Programmbibliothek (z.B. Mikrokontroller-Betriebssysteme)

Häufig findet man Kombinationen dieser vier Varianten. Beispielsweise können Sprach-Laufzeitsysteme Fähigkeiten zur Verfügung stellen, die ansonsten nur Bestandteil von Betriebssystemen sind. Dies beinhaltet Multitasking-Funktionen (z.B. in Java, Ada, Modula-2) und die Speicherverwaltung (verschiedene Sprachen).

1.4.1 Klassische Einteilungen

Eine elementare Klassifizierung von Betriebssystemen basiert auf folgenden Anwendungsarten:

- *Stapelverarbeitung (batch processing)*: Typisches Merkmal ist, dass Programme angestoßen werden, aber ansonsten keine nennenswerte Benutzerinteraktion stattfindet. Die auszuführenden Befehle sind stattdessen in einer Stapeldatei abgelegt, deren Inhalt fortlaufend interpretiert wird. Klassische Großrechnerbetriebssysteme werden auf diese Art und Weise genutzt, z.B. zur Ausführung von Buchhaltungsprogrammen über Nacht.
- *Time-Sharing-Betrieb*: Die zur Verfügung stehende Rechenleistung wird in Form von Zeitscheiben (*time slices, time shares*) auf die einzelnen Benutzer aufgeteilt mit dem Ziel, dass jeder Benutzer scheinbar den Rechner für sich alleine zur Verfügung hat. Historisch gesehen sind Time-Sharing-Systeme die Nachfolger bzw. Ergänzung der Batch-Systeme mit der Neuerung, dass sie Benutzer interaktiv arbeiten lassen (Dialogbetrieb).
- *Echtzeitbetrieb*: Die Rechenleistung wird auf mehrere Benutzer oder zumindest Prozesse aufgeteilt, wobei zeitliche Randbedingungen beachtet werden. Oft sind Echtzeitsysteme reaktive Systeme, indem sie auf gewisse Signale aus der Umgebung (Interrupts, Meldungen) möglichst rasch reagieren.

1.4.2 Moderne Einteilungen

Moderne Betriebssysteme fallen mehr oder weniger in die Gruppe der Echtzeitsysteme, weswegen letztere für uns im Vordergrund stehen. Eine ergänzende Klassifizierung unterteilt Betriebssysteme nach unterstützter Rechnerstruktur:

- Einprozessorsysteme
- Multiprozessorsysteme
- Verteiltes System

Je nach Auslegung kann ein Betriebssystem eine oder mehrere dieser drei Rechnerstrukturen unterstützen. Ergänzend sei noch bemerkt, dass populäre Betriebssysteme netzwerkfähig (*networked operating system*) sind, auch wenn sie nicht

verteilt ablaufen. Beispielsweise unterstützen sie verbreitete Netzwerkprotokolle, die Anbindung entfernter Laufwerke und – teilweise konfigurierbar – eine zentralisierte Benutzerverwaltung.

Beispiele:
Windows unterstützt Einprozessorsysteme und Multiprozessorsysteme. Das Betriebssystem Amoeba ermöglicht transparentes Arbeiten auf einem verteilten System. Für den Benutzer präsentiert es sich wie ein Einzelrechner, besteht in der Tat aber aus mehreren über ein Netzwerk verbundenen Computern.

1.4.3 Geschichte

Abbildung 1–4 zeigt eine kleine Auswahl an Entwicklungslinien gängiger Betriebssysteme, deren Geschichte wir kurz charakterisieren.

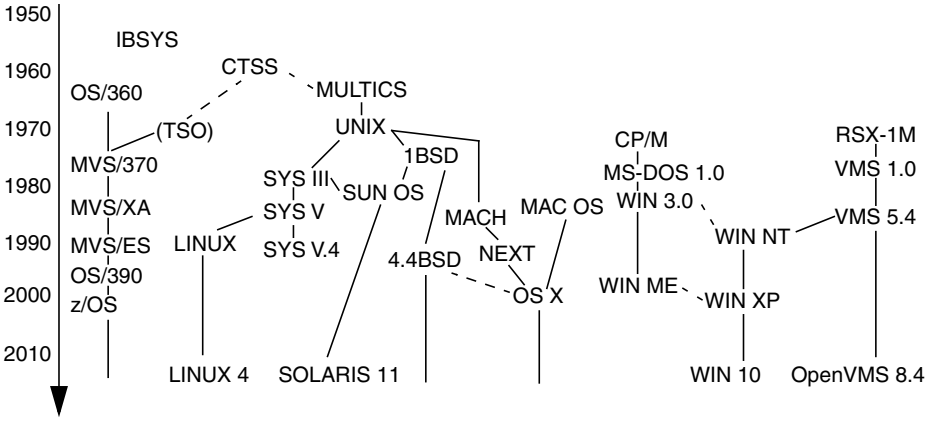

Abb. 1–4 *Entwicklungslinien einiger gängiger Betriebssysteme*

Ein erstes Betriebssystem für Großrechner war das rudimentäre *IBSYS*, das Stapelverarbeitung ermöglichte. Umfangreicher war bereits das *OS/360* von IBM, das in weiterentwickelter Form als *z/OS* auf heutigen Mainframe-Systemen läuft. Anfänglich hat es nur die Stapelverarbeitung unterstützt, wurde aber bald durch die *TSO (Time Sharing Option)* für den Dialogbetrieb ergänzt. Unabhängig davon entstand das *CTSS (Compatible Time Sharing System)*, das den Dialogbetrieb auf Großrechnern bereits sehr früh erlaubte. Sein Nachfolger war *MULTICS (Multiplexed Information and Computing Service)*, ein Konsortiumsprojekt, das letztlich nicht sehr erfolgreich war, jedoch viele neue Konzepte realisierte. Darin war es ein Vorbild für das ursprüngliche *Unix*, das jedoch ein wesentlich kompakterer Entwurf war, der die Komplexität des MULTICS vermied. Unix hat über viele Zwischenschritte die heutigen Systeme *Linux*, *Oracle Solaris* und *Apple OS X*

geprägt. Das *BSD (Berkeley Software Distribution) Unix* existiert heute als *FreeBSD*, *NetBSD* und *OpenBSD* in geringer Verbreitung weiter. Separate Entwicklungslinien gelten für das *Microsoft Windows*. Ursprünglich hat es als grafische Oberfläche für *MS-DOS* begonnen, wurde aber immer unabhängiger davon. Separat zu dieser originären Windows-Linie entstand das *Windows NT*, das von den *DEC VMS (Virtual Memory System)* Minicomputer-Betriebssystemen abgeleitet wurde, jedoch die API des Microsoft Windows realisierte. Mit dem *Windows XP* wurde die originäre Windows-Linie beendet, womit der schwache Unterbau des MS-DOS verschwand. Das VMS existiert als *OpenVMS* noch heute, ist aber nur minimal verbreitet.

1.5 Betriebssystemarchitekturen

Beim Entwurf eines Betriebssystems sind viele Anforderungen in Einklang zu bringen, die nicht widerspruchsfrei sind, weswegen Kompromisse nötig sind. Neben der Realisierung der in Abschnitt 1.1 beschriebenen Kernfunktionalitäten sind folgende exemplarische Entwurfsziele zu berücksichtigen:

- Fehlerfreiheit des Codes: z.B. durch minimale Komplexität des Quellcodes
- Einfache Operationen (auf API und für alle Schnittstellen)
- Erweiterbarkeit (*extensibility*)
- Skalierbarkeit (*scalability*)
- Orthogonalität: Operationen wirken gleich auf verschiedenartigen Objekten
- Robuste Betriebsumgebung (»crash-proof«, »reliable«)
- Einhaltung der Sicherheitsziele (mehrere Anforderungsstufen denkbar)
- Portabilität (Unterstützung verschiedenartiger Plattformen)
- Echtzeitfähigkeit (z.B. für Multimedia-Anwendungen)
- Effizienz (schnelle Diensterbringung, minimaler Ressourcenbedarf)
- Weiterentwickelbarkeit: Trennung von Strategie (*policy*) und Mechanismus (*mechanism*)

Auf der Suche nach einem optimalen Entwurf sind verschiedenartige Architekturideen entwickelt worden. Diese werden nachfolgend kurz beschrieben und diskutiert. Als Blick in die mögliche Zukunft des Betriebssystembaus wird eine kurze Zusammenfassung einiger interessanter Forschungsarbeiten zum Thema vorgestellt.

1.5.1 Architekturformen

Solange es lediglich um die Systemprogrammierung geht, ist eine Blackbox-Betrachtung des Betriebssystems ausreichend. Nach außen ist damit nur die Programmierschnittstelle sichtbar, jedoch nicht das Systeminnere (siehe A in Abb. 1–5). Dies entspricht einem klassischen Ideal des Software Engineering, das aussagt,

dass die Schnittstelle das Maß aller Dinge ist und die Implementierung dahinter beliebig austauschbar sein soll. Dennoch kann es hilfreich sein, die Innereien eines Betriebssystems zu kennen, damit man nicht Gefahr läuft, gegen die Implementierung zu programmieren. Dies könnte zum Beispiel in einem überhöhten Ressourcenverbrauch oder einer unbefriedigenden Ausführungsgeschwindigkeit resultieren. Daneben ist es stets interessant, unter die »Motorraumhaube« eines Betriebssystems zu gucken. Mit anderen Worten, es geht um eine Whitebox-Betrachtung (siehe B in Abb. 1–5).

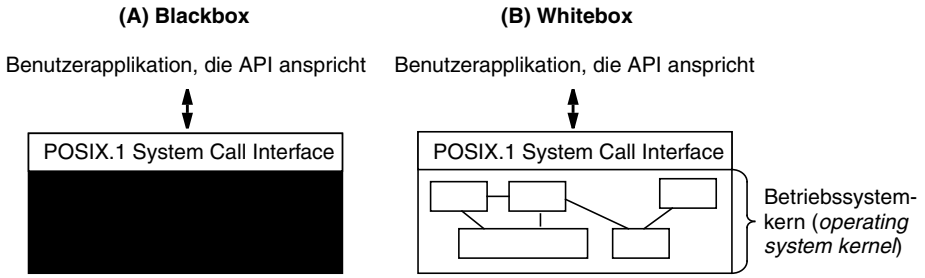

Abb. 1–5 *Black- und Whitebox-Betrachtung (Beispiel: Unix)*

Damit verbunden sind die Entwurfs- und Konstruktionsprinzipien, die einen erst dann interessieren, wenn man über die Blackbox-Betrachtung hinausgeht. Eine wesentliche Frage ist dabei die Art und Weise, wie die Betriebssystemsoftware strukturiert ist. In der Theorie kennt man *drei Grundstrukturen*, denen sich konkrete Betriebssysteme zuordnen lassen: *monolithische, geschichtete* und *Mikrokernsysteme*. Diese werden durch Strukturen für Multiprozessor- und Verteilte Systeme ergänzt. Zuerst soll jedoch auf die Funktionsweise und Bedeutung der Benutzer-/Kernmodus-Umschaltung eingegangen werden, da sie bei der Betrachtung dieser Strukturen eine zentrale Rolle spielt.

1.5.2 Benutzer-/Kernmodus

Als hardwarenahe Softwarekomponente ist ein Betriebssystem eng mit den Möglichkeiten der unterliegenden Plattform verbunden. Es haben sich mit den Jahren unterschiedliche Leistungsklassen von Prozessoren und zugehöriger Hilfslogik etabliert:

- *Mikrocontroller*: Es handelt sich hierbei um einfache Mikroprozessoren, die primär in sehr einfachen eingebetteten Systemen (*embedded systems*) eingesetzt werden. Um die Kosten gering zu halten, verfügen sie lediglich über einen Prozessor mit wenig oder gar keinen weiterführenden Mechanismen zur Unterstützung eines Betriebssystems. Hingegen sind sie zusammen mit verschiedenen Peripherieeinheiten (E/A, Kommunikation, Zeitgeber usw.) in