# Betriebssysteme

Dieter Zöbel

Universität Koblenz-Landau
Fachbereich Informatik
Institut für Informatik

# Inhaltsverzeichnis

# Kapitel 1

# Einführung

Dieses Kapitel befasst sich im Wesentlichen mit

- der Aufgabenstellung von Betriebssystemen,
- den Epochen der Betriebssystem-Entwicklung,
- den Aufbauprinzipien von Betriebssystemen,
- der Einsatzbreite von Betriebssystemen,
- und der Genealogie wichtiger Betriebssysteme

## 1.1 Allgemeine Zielsetzung

Nutzbarmachung des Rechners

- Bereitstellung einer ergonomischen Benutzerschnittstelle
- Abstraktion eexAbstraktion von den technischen Einzelheiten des Rechensystems
- Anpassung an unterschiedliche und wechselnde Aufgabenfelder
- Schutz der Benutzer und des Rechensystems vor fehlerhafter oder zerstörerischer Benutzung
- Ausnutzung der Leistungsfähigkeit

# Definitionsversuch des Begriffs Betriebssystem (1)

Definition des Begriffs *Betriebssystem* anhand der DIN-Norm 44300 [1] :

> Die Programme eines digitalen Rechensystems, die zusammen mit den Eigenschaften der Rechenanlage die Grundlage der möglichen Betriebsarten des digitalen Rechensystems bilden und insbesondere die Abwicklung von Programmen steuern und überwachen.

Die Fähigkeiten einer Rechenanlage werden durch den Prozessor und seine *Betriebsmittel* bestimmt [35] :

> Als Betriebsmittel in digitalen Rechensystemen bezeichnen wir die Prozessoren, die Speicher (Haupt- und Hintergrundspeicher), E/A-Geräte, Editoren, Übersetzer, Benutzer- und Dienstprogramme.

## Definitionsversuch des Begriffs Betriebssystem (2)

Bedeutung des Betriebssystems nach Elmasri et alii [13]:

> Operating systems are the heart of every computer system. The OS provides services for users and programmers that make it possible to utilize a computer without having to deal with the low-level, difficult-to-use hardware commands.

# Rechensystem und Betriebsmittel

Ein *Rechensystem* besteht typischerweise aus einem Prozessor oder mehreren Prozessoren, verschiedenen Speichern wie Haupt- und Hintergrundspeicher, Bediengeräten wie einer Maus, einer Tastatur und einem Bildschirm. Weitere Geräte sind beispielsweise Video- und Sound-Karten sowie der Anbindung an Rechnernetze durch Netzwerkkarten.

Alle die erwähnten Bestandteile eines Rechensystems werden als *Betriebsmittel* bezeichnet.
Nach Stuart [38] ist die Aufgabe eines Betriebssystems:

- *... manage resources ...*
- *... provide services ...*
- *... serve as interface ...*

# Schnittstellen zum Betriebssystem

- Schnittstelle für den Benutzer (meist GUI[1])
- Eingabe über Kommandofenster (z.B.: Kommandos eingebettet in eine Skriptsprache)
- Programmierschnittstelle für den Entwickler (meist betriebssystemspezifische oder generalisierende APIs[2] sowie GUI-Builder)
- Konfigurations- und Managementschnittstelle (meist über die SPI[3])

[1] Graphical User Interface
[2] Application Programmer's Interface
[3] System Programmer's Interface

# Rollen zum Betriebssystem (1)

- Nutzer eines bereits vorinstallierten Betriebssystems
  - Organisation des Dateibaums
  - Aufspielen von neuer Software
- Selbständiger Nutzer eines Betriebssystems
  - Einbau neuer Geräte
  - Installation und Konfiguration von Treibern

# Rollen zum Betriebssystem (2)

- Anwendungsentwickler
  - Nutzung von APIs zum Betriebssystem
  - Kenntnis der grundsätzlichen Dienste und Funktionen des Betriebssystems
  - Nutzung der GUI-Schnittstellen
- Administration von Nutzern eines Betriebssystems oder mehrerer Betriebssysteme
  - Einrichten von Benutzerkonten und Vergabe von Benutzerrechten
  - Schützen von Benutzern

# Rollen zum Betriebssystem (3)

- Entwickler von Komponenten eines Betriebssystems, beispielsweise eines Treibers
  - Kenntnis über die programmiertechnische Einbindung von Komponenten in das Betriebssystem
  - Kenntnis der SPI
- Entwerfer, Entwickler und Pfleger eines Betriebssystems
  - Umfassende Kenntnis über die softwaretechnische Struktur eines Betriebssystems
  - Verständnis für die Breite und Tiefe der Aufgaben und Notwendigkeiten bei Betriebssystemen

## 1.2 Epochen der Betriebssystem-Entwicklung

Tanenbaum [40] unterscheidet (bis 1995) vier Epochen der Betriebssystem-Entwicklung. Hier sollen sechs Epochen charakterisiert werden:

1. elektrische Röhre (bis 1955)
2. Transistor (1955-1970)
3. integrierte Schaltkreise (1970-1980)
4. hochintegrierte Schaltkreise (1980-1995)
5. Rechner im WAN[4] (1995-2010)
6. der allgegenwärtige Rechner (ab 2010)

[4] Wide Area Network

# Anmerkungen zu den Epochen (1)

zu 1.: elektrische Röhre (bis 1955)

- außerhalb jeglicher betriebswirtschaftlicher Nutzung
- angetrieben von Militär und Wissenschaft

zu 2.: Transistor (1955-1970)

- Kommerzialisierung im Bereich von Großunternehmen
- betriebswirtschaftliche Nutzung bei Banken und Versicherungen
- IT-Unternehmen werden zu Weltkonzernen

# Anmerkungen zu den Epochen (2)

zu 3.: integrierte Schaltkreise (1970-1980)

- Rechnereinsatz in fast allen Bereichen von Handel, Industrie und Verwaltung
- Informatik wird eigenständige Wissenschaft
- erste Diskussion gesellschaftlicher Auswirkungen der Datenverarbeitung
- Mikroprozessoren und Mikrocontroller

zu 4.: hochintegrierte Schaltkreise (1980-1995)

- Vordringen des Rechners auf den Büroarbeitsplatz und in die Privatsphäre
- Einführung und Dominanz des PC (bzw. Workstation)
- leistungsfähige lokale Netzwerke (LAN)
- Verfügbarkeit eines breitgefächerten und preisgünstigen Softwareangebotes
- Aufkommen des Problems der IT-Sicherheit

# Anmerkungen zu den Epochen (3)

zu 5.: Rechner im WAN (1995-2010)

- Rechner als Tor zur Welt: *Die Welt ist ein Dorf*
- leistungsfähige (Weitverkehrs-) Netzwerke (WAN)
- weltweites Dienstangebot verfügbar für den lokalen PC
- IT-Sicherheit wird zu einem zentralen Problem

zu 6.: der allgegenwärtige Rechner (ab 2010)

- Erreichung der Integrationsgrenze und Verbreitung der Mehrkernprozessoren
- Schrumpfung des Rechners
- ununterbrochene Erreichbarkeit
- Verschmelzung von Telefon- und Datennetzen und der zugehörigen Dienste
- IT-Sicherheitsprobleme bei eingebetten Systemen und Cyberphysikalischer Systeme (CPS)

# Meilensteine und Betriebssystem-Entwicklung

- Mehrprogrammbetrieb und Abstraktionsobjekt *Prozess*
- virtuelle Adressierung
- Mikrokern-Architektur
- Client-Server-Paradigma (beliebiger Ort der Diensterbringung, Problem: wie findet der Kunde (anonym) den Dienst (öffentlich))
- Virtualisierung (virtuelle Maschine)

# Beitrag des Fachgebietes Betriebssysteme

- Elementare und komplexe Dienste zur Nutzung von Rechensystemen
- Verfahren zur Nutzung von Betriebsmitteln
- Konzepte und Methoden der parallelen und verteilten Programmierung
- Operative und analytische Methoden der Leistungsbewertung

## 1.3 Aufbauprinzipien von Betriebssystemen

Zielsetzung:

- Abstraktion und Abstraktionsobjekt *Prozess*
- Kategorisierung von Aufbauformen
- Betrachtung von Betriebssystemen aus wechselnden Blickwinkeln

# Abstraktion

Prinzip:

- konzeptuell: Virtualisierung, d.h. das Vortäuschen einer vereinfachten Umgebung beispielsweise in der virtuellen Adressierung oder bei der virtuellen Maschine
- (programmier-)technisch: Schnittstelle, d.h. das Bereitstellen von Methoden, die das System auf der abstrakten Ebene der Parameter beschreiben, beispielweise durch die API oder SPI eines Betriebssystems

# Prozesse

Nach Stuart [38] dient ein Betriebssystem dazu, Prozesse auszuführen:
*An operating system is a set of one or more programs which provides a set of services that interface applications to computer hardware and which allocates and manages resources shared among multiple processes.*

Das Betriebssystem selbst ist aus Prozessen aufgebaut [38] :
*We refer to these programs in execution as processes.*

Schnittstellen für Prozesse:

- Erzeugen und Entfernen von Prozessen,
- Kommunikation und Synchronisierung zwischen Prozessen,
- Festlegen und Ändern von Prozesseigenschaften

# Klassifizierung der Aufbauformen

- Monolithischer Aufbau
- Schalenorientierter Aufbau
- Virtuelles Aufbauschema
- Mikrokern-basierer Aufbau

### 1.3.1 Monolithischer Aufbau

Das Betriebssystem stellt sich für den Benutzer wie ein geschlossenes Programmsystem dar, das über *Systemaufrufe* Aufträge entgegenimmt und bearbeitet.
*Betriebsmodi*, die von der Hardware unterstützt werden:

- Benutzermodus (*user mode*)
- System- oder Supervisormodus

Im Systemmodus verfügt das ausführende Programm über alle Rechte, beispielsweise den Zugriff auf alle Speicheradressen und die Ausführung alle vorhandenen Befehle des Prozessors. Im Benutzermodus gibt es diesbezüglich Einschränkungen.

### 1.3.2 Schalenorientierter Aufbau

Prinzip der funktionalen Hierarchisierung der Aufgaben (Prozesse) innerhalb von Schalen: Die Schnittstellen einer Schale bieten ihre Dienst ausschließlich den Prozessen höheren Schalen an.

Ebenso möglich ist eine Schalenorientierung, die sich an der Nähe und Abhängigkeit von der Hardware orientiert.

### 1.3.3 Virtuelles Aufbauschema

Konzept: Ein Betriebssystem B wird als Prozess eines darunterliegenden Betriebssystems A ausgeführt.
Vorteile:

- Aufteilung der einen Hardware an mehrere gleichzeitig ausgeführte Betriebssysteme
- Ausübung weitergehender Schutzfunktionen durch das darunterliegende Betriebssystem
- Vereinfachung der Administration von Systemen

# Virtualisierung: Definitionsversuch

... [32] steckt hinter der Virtualisierung die Grundidee, dass sich verschiedene Betriebssysteme eine Hardware teilen und sich dabei jede Betriebssysteminstanz so verhält wie auf einem physikalischen Rechner. Dadurch wird also ein gleichzeitiger Betrieb verschiedener Systeme unabhängig voneinander auf der Basis derselben Hardware möglich.

Durch verschiedene Software- und/oder hardware-Techniken, abhängig von der Implementierung, werden dabei Betriebssysteminstanzen voneinander abgeschirmt, um parallel koexistieren zu können.

### 1.3.4 Mikrokern-basierer Aufbau

Definition des Begriffs *Kern*: Der Teil des Betriebssystems, der für dessen Funktion *unverzichtbar* ist. D.h, es gibt Anteile des Betriebsystems im Kern und außerhalb.

Ein *Mikrokern-Betriebssystem* liegt dann vor, wenn nur der unverzichtbare Anteil im Systemmodus abläuft.

Bestandteile des Mikrokerns:

- Prozessplanung und -verwaltung
- Interprozesskommunikation
- Prozessbezogene Teile der Speicherverwaltung
- Unterbrechungsverwaltung

## Weitere Definitionen des Begriffs Kern

Zum Begriff Kern:

- *... the part of the system software executing in supervisor state is called kernel, or nucleus of the operating system.* [29], S. 58
- *... is that the operating system is the one programm running all times on the computer (usually called the kernel) ...* [36], S. 5
- *All of the operations involving processes are controlled by a portion of the operating system variously called its nucleus, core, or kernel ... it is the most intensively used code. For this reason, the nucleus ordinarily remains in primary storage ... The nucleus disables interrupts while it is responding to an interrupt ...* [10], S. 61

Zum Begriff Mikrokern:
*Man isoliere eine möglichst kleine Menge von essenziellen Systemfunktionen im eigentlichen Kern.* [5]

# 1.4 Einsatzbreite von Betriebssystemen

Betriebsarten (klassische):

- Stapelbetrieb
- Dialog-Betrieb (*time-sharing*)
- Echtzeitbetrieb

Betriebsarten (neuere):

- Serverbetrieb
- *embedded*, *mobile* und *ubiquitous*-Betrieb
- Virtualisierungs-Betrieb

# Betriebsarten (1)

- Stapelbetrieb
  Übergabe zusammenhängender Auftragsfolgen
  z.B.: Änderung der Kundenstammdaten aufgrund gesetzlicher Änderungen
- Dialog- oder Gesprächsbetrieb (auch *interactive mode*)
  Wiederkehrende Beauftragung eines Rechensystems mit Teil- bzw. Folgeaufträgen, wobei der Auftraggeber (Benutzer) den Eindruck haben soll, allein im Besitz des Rechensystems zu sein
- Serverbetrieb (*server mode*) oder Data Center
  Rechensystem erledigt die über Netzwerk eintreffenden Aufträge (z.B. mittels Fernaufruf) und verfügt über ein spezifisches Spektrum an Kompetenzen
  z.B.: der Buchungsserver der Deutschen Bahn

# Betriebsarten (2)

- Echtzeitbetrieb (*real-time mode*)
  Aufträge an das Rechensystem sind unter Einhaltung von Zeitbedingungen zu erledigen
  z.B.: Antiblockiersystem (ABS)
- eingebettete Systeme
  - das Rechensystem tritt nicht offen sichtbar in Erscheinung
  - Auftragserteilung erfolgt nur mittelbar über den Benutzer (z.B. durch Betätigen der Bremse)
  - erfüllt autonom eine Auftrag (in diesem Zusammenhang als *mission* bezeichnet)
- Allgegenwärtige Systeme (*ubiquitous systems*)
  Rechensystem als universeller Helfer, jederzeit und überall
  - Weiterentwicklung mobiler Betriebssysteme
  - spezialisierte Benutzerschnittstelle
  - Anpassung der Dienstleistung an Ort, Zeit und Situation

# Entwicklungsformen von Echtzeitbetriebssystemen

Darstellung anhand von Echtzeitbetriebssystemen

- Laufzeitsysteme (*executives*)
  Minimalfunktionen eines Betriebsystems, insbesondere um Vorhersagbarkeit (*predictability*) zu sichern
  Leistungen: Prozessumschaltung, Synchronisierung, Zeitfunktionen, dafür jedoch keine oder kaum E/A-Funktionalität
  z.B.: VRTX, RTKernel
- Modifikationen existierender Betriebsysteme
  z.B.: in der Unix-Welt: AIX, RT-Linux, LynxOS, verschiedene Linux-Distributionen (z.B. Ubuntu), Android?
  z.B.: in der Microsoft-Welt: WindowsCE
- Eigenentwicklungen von Betriebsystemen
  z.B.: iRMX, QNX, Tornado, EUROS, ITRON, OSEK

## Standarisierung: POSIX

Ziel: Schaffung von Kompatibilitäten, was die Portabilität von Programmen über unterschiedliche Betriebsysteme hinweg angeht.

Wichtigster Standard: POSIX[5] [6]
Zielsetzung der IEEE: Definition einer anwendungsspezifischen Programmierschnittstelle (API[7]) zum Betriebssystem (nicht notwendigerweise UNIX), erster Vorschlag (*draft*) 1985 vom IEEE-Komittee 1003

[5] (engl.: *portable operating system interface for computer environments*)
[6] Neben POSIX gibt es aus Japan den TRON-Standard (*The Real-time Operating system Nucleus*) eingesetzt in Mobiltelefonen, Digitalkameras und anderen elektronischen Geräten
[7] (engl.: *application programmers interface*)

# Familie von IEEE/ISO POSIX-Standards

| Working Groups | Charter |
|---|---|
| POSIX.0 | Open-system architecture |
| POSIX.1 | Posix application interface |
| POSIX.2 | Shell and command utilities |
| POSIX.3 | Testing and verification methods |
| POSIX.4 | Real-time extensions to POSIX |
| POSIX.5 | Ada binding to POSIX.1 |
| ... | ... |
| POSIX.13 | Realtime application environment profiles |
| POSIX.14 | Multiprocessing application environment profiles |
| ... | ... |
| POSIX.20 | Ada binding to POSIX.4 |
| POSIX.21 | Distributed real-time |

## **POSIX 1003.1b** *real-time*

Merkmale[8]

- *semaphores*
- *shared memory*
- *locking processes in RAM*
- *memory mapped files*
- *asynchonous I/O*
- *high resolution clocks and timers*
- *signals*
- *scheduling*

[8] POSIX 1003.1b korrespondiert zu POSIX.4

# Vergleich von Prozessen mit Threads

**Prozess** (oft auch Task genannt)

- kann mehrere Threads enthalten
- eigener Adressraum
- besitzt Betriebsmittel z.B. offene Dateien
- Kommunikation mit anderen Prozessen über Pipes oder Nachrichten

**Thread**

- ist einem Prozess zugeordnet
- gemeinsamer Adressraum
- teilt Betriebsmittel mit den anderen Threads des Prozesses
- Kommunikation mit anderen Threads über den gemeinsam Speicher

# Mobile Betriebssysteme

Ein solches Betriebssystem ist gekennzeichnet durch:

- eingeschränkte bzw. spezialisierte E/A-Schnittstelle
- einschränkte Speicher- und Rechensysteme
- energiegewahre Nutzung des Rechensystems

Typische Mobile Betriebsysteme:

- SymbianOS (Nokia und andere)
- Windows (Mobile) Phone basierend auf dem Kern von WindowsCE
- iOS
- Linux-Derivate (z.B. Montavista)
- Google Android[9]

Zielsysteme:

- portable Telefone
- personal digital assistant (PDA)
- Smartphones
- Pocket PC

[9] Letzte Version *Android 5.0 Lollipop* vom 3.11.2014

## 1.5 Genealogie wichtiger Betriebssysteme

Während früher die Großrechner-Betriebssysteme die Entwicklung maßgeblich bestimmten, wird die Entwicklung der Betriebssysteme etwa ab 1980 durch zwei unterschiedliche Entwicklungslinien dominiert:

- Unix-basierte Betriebssysteme
- Microsoft-basierte Betriebssysteme

# Literaturverzeichnis


[1] *DIN 44300, Normen über Informationsverarbeitung*. Beuth Verlag, Berlin, 1975.

[2] Albrecht Achilles. *Betriebssysteme*. eXamen.press. Springer-Verlag, Berlin, 2006.

[3] G. R. Andrews. *Concurrent Programming*. The Benjamin/Cummings Publishing Company, 1991.

[4] G. Bengel, Chr. Baun, M. Kunze, and K.-U. Stucky. *Masterkurs Parallele und Verteilte Systeme*. Vieweg+Teubner, Wiesbaden, 2008.

[5] Lothar Borrmann. Kleine und kleinste Betriebssysteme mit Mikro- und Nanokernen. *Informationstechnik und Technische Informatik it+ti, Oldenbourg Verlag*, 38(2):18–25, 1996.

[6] L. Bougé. On the existence of symmetric algorithms to find leaders in networks of communicating sequential processes. *Acta Informatica*, 25:179–201, 1988.

[7] N. Carriero and D. Gelernter. Linda in context. *Communications of the ACM*, 32(4):444–458, 1989.

[8] E.G. Coffman, M.J. Elphick, and A. Shoshani. System deadlocks. *Computing Surveys*, 3(3):67–68, 1971.

[9] P.J. Courtois, D.L. Heymans, and F. Parnas. Concurrent control with readers and writers. *CACM*, 14(10):667–668, October 1974.

[10] H. M. Deitel. *Operating Systems*. Addison Wesley, Reading, Massachusetts, 1990.

[11] P.J. Denning. The working set model for program behavior. *CACM*, 11(5):323–333, May 1968.

[12] E. W. Dijkstra. Cooperating sequential processes. In F. Genuys, editor, *Programming Languages*, pages 43–112. Academic Press, New York, 1968.

[13] Ramez Elmasri, Gil Carrik, and David Levine. *Operating Systems - A Spiral Approach*. McGraw-Hill, Boston, 2009.

[14] M. Flynn. Very high speed computing systems. *Proc. of the IEEE*, 54(9):1901–1909, 1966.

[15] Jörg Hettel and Man Tien Tran. *Nebenläufige Programmierung mit Java*. Dpunkt Verlag, Heidelberg, 2016.

[16] C. A. R. Hoare. Communicating sequential processes. *Communications of the ACM*, 21(8):666–677, 1978.

[17] C.A.R. Hoare. Monitors: An operating system structuring concept. *CACM*, 17(10):549–557, October 1974.

[18] Matthias Homann. *OSEK - Betriebssystem-Standard für Automotive und Embedded Systeme*. mitp-Verlag, Bonn, 2005.

[19] Cay Horstmann. *Big Java - Late Objects*. John Wiley & Sons, New York, 2013.

[20] D.G. Kendall. Stochastic processes occuring in the theory of queues and their analysis by the method of embedded Markov chains. *Annals of Math. Statistics,*, 24:338–354, 1953.

[21] Nathan Koyzra. *Mastering concurrency in GO*. Packt-Publishing, Bermingham, United Kingdom, 2014.

[22] Thomas Kuhn. *Die Struktur der wissenschaftlichen Revolution*. Surkamp Verlag, Frankfurt, 1969.

[23] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, 1978.

[24] J. D. C. Little. A proof of the queueing formula $L = \lambda W$. *Operations Research*, 9:383 – 387, 1961.

[25] F. Mattern. Virtual time and global states of distributed systems. In M. Cosnard, editor, *Workshop on Parallel and Distributed Systems*, pages 215–226, Chateau de Bonas, France, October 1988. Elsevier.

[26] Jörg Mühlbacher. *Betriebssysteme*. Universitätsverlag Rudolf Trauner, Linz, 2009.

[27] Nicholas Ng and Nobuko Yoshida. Static deadlock detection for concurrent go by global session graph synthesis. In *Proceedings of the 25th International Conference on Compiler Construction, CC 2016, Barcelona, Spain, March 12-18, 2016*, pages 174–184, 2016.

[28] John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron. Scalable parallel programming with CUDA. *ACM Queue*, 6(2):40–53, 2008.

[29] Gary Nutt. *Operating Systems - A Modern Perspective*. Addison-Wesley, Reading, 1997.

[30] D. Patterson, G. Gibson, and R. Katz. A case for redundant arrays of inexpensive disks (RAID). In *Proceedings of the ACM SIGMOD Conference on Management of Data*, San Francisco, June 1988.

[31] G. L. Peterson. Myths about the mutual exclusion problem. *Information Processing Letters*, 12(3):115–116, June 1981.

[32] Hans-Joachim Picht. *Xen Kochbuch*. O'Reilly Verlag, K¨oln, 2009.

[33] Gerals J. Popek and Robert P. Goldberg. Formal requirements for virtualizable third generation architecture. *Communications of the ACM*, 7(7):412–421, July 1974.

[34] Thomas Rauber and Gundula Rünger. *Parallele Programmierung*. Springer-Verlag, Berlin, 2nd. ed. edition, 2007.

[35] Lutz Richter. *Betriebssysteme*. Teubner Verlag, Stuttgart, 1985.

[36] Abraham Silberschatz and Peter B. Galvin. *Operating System Concepts*. Addison-Wesley, Reading, 4 edition, 1994.

[37] Daniel J. Sorin, Mark D. Hill, and David A. Wood. *A Primer on Memory Consistency and Cache Coherence*. Morgan and Claypool Publishers, Williston, Vermont, 2011.

[38] Brian L. Stuart. *Principles of Operating Systems: Design and Application*. Thomson Learning, Boston, Mass., 2008.

[39] A. S. Tanenbaum. *Operating Systems*. Prentice-Hall International Editions, Englewood Cliffs, NJ, 1987.

[40] Andrew Tanenbaum. *Moderne Betriebssysteme*. Hanser Verlag, München, 1995.

[41] Michael Weber. *Verteilte Systeme*. Spektrum Akademischer Verlag, Heidelberg, 1998.

[42] Anthony Williams. *C++ Concurrency in Action - Practical Multithreading*. Manning Press, New York, 2012.

[43] Nikolaus Wirth. *Programming in Modula-2*. Springer-Verlag, Heidelberg, 1985.

[44] Dieter Zöbel. The deadlock problem: A classifying bibiography. *Operating Systems Review*, 17(4):5–15, 1983.