Betriebssysteme

Eine Einführung

von
Prof. Dr. Uwe Baumgarten
und
Prof. Dr. Hans-Jürgen Siegert

Institut für Informatik
TU München

6., überarbeitete, aktualisierte und
erweiterte Auflage

Oldenbourg Verlag München Wien

Prof. Dr. Uwe Baumgarten
Studium der Informatik und Promotion an der Universität Bonn. Habilitation 1993 an der Universität Oldenburg in Informatik. 1993/4 Vertretung einer Professur für Betriebssysteme an der Universität Bremen. Seit Herbst 1994 Professor für Informatik an der Technischen Universität München. Arbeitsgebiete: Vernetzte Systeme, Betriebssysteme, Mobile Systeme.
siegert@in.tum.de

Prof. Dr.-Ing. Hans-Jürgen Siegert
Studium der Physik und Promotion an der Technischen Universität Stuttgart. Ab 1967 bei AEG-Telefunken und Computergesellschaft Konstanz; zunächst Systemberater, später Abteilungsleiter und für die Entwicklung der Grund- und Anwendungssoftware der Großrechner sowie für die Entwicklungsrechenzentren verantwortlich. Seit 1975 Professor für Informatik an der Technischen Universität München. Arbeitsgebiete: Zunächst Betriebssysteme und Rechnernetze, dann Echtzeitsysteme und Robotik.
baumgaru@in.tum.de

Postanschrift der Verfasser:
Institut für Informatik, Technische Universität München, D-80290 München.

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.

Rosenheimer Straße 145, D-81671 München
Telefon: (089) 45051-0
oldenbourg.de

Lektorat: Margit Roth
Herstellung: Dr. Rolf Jäger
Satz: DTP-Vorlagen des Autors
Coverentwurf: Kochan & Partner, München
Coverausführung: Gerbert-Satz, Grasbrunn
Gedruckt auf säure- und chlorfreiem Papier
Gesamtherstellung: Druckhaus „Thomas Müntzer“ GmbH, Bad Langensalza

ISBN 3-486-58211-9
ISBN 978-3-486-58211-6

Inhaltsverzeichnis

Vorwort

Zur sechsten Auflage

Die ersten Auflagen des Buchs „Betriebssysteme: Eine Einführung" sind noch in der Reihe Handbuch der Informatik des Oldenbourg-Verlags erschienen. Herausgeber dieser Reihe waren die Herren Prof. Dr. A. Endres, Prof. Dr. H. Krallmann und Dr. P. Schnupp. Ab der vierten Auflage war das Buch eigenständig.

Von Anfang an lagen dem Buch moderne, saubere und zukunftsweisende Konzepte für Betriebssysteme zu Grunde. Deshalb gab es bisher auch keinen Anlass, die bewährte Grundstruktur des Buches zu verändern. Jedoch wurde inhaltlich auch die sechste Auflage auf den aktuellsten Stand der Technik gebracht. Sie stellt daher eine vollständige Überarbeitung, Aktualisierung und Erweiterung dar.

Das Buch gibt eine fundierte Einführung in die Grundlagen und Grundkonzepte moderner Betriebssysteme. Es wendet sich insbesondere an Informatiker, an Ingenieure, an Studierende der Informatik und Informationstechnik im Haupt- und Nebenfach, an Systemprogrammierer und nicht zuletzt an alle Anwender der Datenverarbeitung, die sich für Betriebssysteme interessieren oder Kenntnisse darüber benötigen. Das Buch ist für das eigenständige Beschäftigen mit dem Stoff sehr gut geeignet.

Ein Betriebssystem gehört zu jedem Rechensystem, sei es ein kleines eingebettetes Systems, ein Personal-Computer oder ein Hochleistungsrechner. Neben den weit verbreiteten (Standard-) Betriebssystemen, wie UNIX (LINUX) oder Windows, bieten viele Hersteller noch eigene Betriebssysteme für ihre Rechensysteme an. Entsprechend vielfältig sind die konkreten Ausprägungen. In dem vorliegenden Buch werden die allgemein gültigen Grundkonzepte dargestellt, die sich im Laufe der Jahre für Betriebssysteme entwickelt haben. Diese Grundkonzepte werden in allgemeinere Abstraktionen und Zusammenhänge eingebunden, so dass sich ein vertieftes Verständnis sowie eine Begründung und Bewertung ergibt. An einigen Stellen erschien es den Verfassern für das Verständnis didaktisch wichtig, konkrete Systemdienste und wichtige Abläufe algorithmisch darzustellen. Diese Algorithmen sind weitgehend umgangssprachlich formuliert. Wei-

terhin werden an vielen Stellen zur Vertiefung und als Beispiel konkrete Realisierungen bei Windows oder Linux geschildert. Dies dient auch als Brückenschlag zwischen den Grundkonzepten und dem täglichen Umfeld der Anwender.

Im Hinblick auf die Zielgruppe der Leser ist der Stil des Buches beschreibend, ohne mathematisch gefasste theoretische Modelle. Trotzdem ist die Formulierung sehr genau überlegt, sehr knapp und möglichst präzis.

Das Buch ist für ein sequenzielles Lesen gedacht. Trotzdem können einzelne Kapitel auch unabhängig von anderen gelesen und verstanden werden. Das detaillierte Register, die umfangreichen Literaturangaben und ein Lexikon korrespondierender deutscher und englischer Begriffe unterstützen das Studium des Buches, aber auch den flüchtigen Leser, der sich nur schnell über einen Sachverhalt informieren möchte. Ein frühzeitiger Blick in das vorgenannte Lexikon wird im Hinblick auf die verwendete Terminologie in diesem Buch und zur eigenen Orientierung in einer durch englische Fachbegriffe geprägten Umwelt empfohlen.

Ein besonderer Service des Oldenbourg-Verlags und der Autoren:
Die Abbildungen dieses Buches können für Projektionsfolien oder zur Einbettung in Powerpoint-Präsentationen beim Oldenbourg-Verlag über Internet (http://www.oldenbourg-wissenschaftsverlag.de) heruntergeladen werden.

Über jede Art von Rückkopplung, insbesondere über Kommentare, Anregungen und Kritiken via Email würden wir uns sehr freuen.

Die Zusammenarbeit mit dem Oldenbourg-Verlag war immer angenehm und problemlos. Dafür bedanken wir uns, insbesondere bei Frau Margit Roth für das Lektorat Informatik.

München, im Oktober 2006

Uwe Baumgarten
Hans-Jürgen Siegert

1 Einführung

In diesem Kapitel wird, ausgehend von einer typischen, aber immer noch interessanten, Rechnerbenutzung in der Frühzeit der Rechentechnik, der Begriff Betriebssystem definiert. Dann werden die grundlegenden Begriffe Prozess, Objekt, Verwalter und Betriebsmittel eingeführt. Danach werden die Aufgaben eines Betriebssystems besprochen. Dies führt zu einer ersten funktionellen Strukturierung. Außerdem zeigt sich dabei, dass ein Betriebssystem seine Aufträge gemäß eingestellten Betriebszielen abwickeln muss. Diese Betriebsziele können in Konflikt zueinander stehen. Das Kapitel schließt mit einer Skizzierung wichtiger Grundkonzepte, die dann in den späteren Kapiteln ausführlich besprochen werden.

1.1 Definition Betriebssystem

frühe Rechnerbenutzung

Einleitend betrachten wir eine typische Rechnerbenutzung Ende der 50er Jahre. Zu dieser Zeit standen dem Benutzer noch keine mächtigen, rechnergestützten Hilfsmittel zur Abwicklung seines Auftrags zur Verfügung. Der Benutzer musste alle Schritte im Arbeitsablauf selbst explizit steuern und zum richtigen Zeitpunkt veranlassen. Die typischen Schritte in einem Arbeitsablauf waren die folgenden:

Arbeitsablauf

- Das Anwendungsprogramm wurde in der Regel in Assemblersprache geschrieben. Hierbei wurde die Lage des Programms und aller Standardprozeduren im Arbeitsspeicher festgelegt.
- Das Anwendungsprogramm und die benötigten Daten wurden auf Lochstreifen abgelocht.
- Der Benutzer reservierte die Anlage für eine gewisse Zeitspanne, meist eine Woche im Voraus, für sich. Typisch war eine Zeitspanne zwischen einer halben und zwei Stunden. Die Betriebszeit einer Rechenanlage wurde dadurch in einzelne Blockzeiten gegliedert. Innerhalb einer solchen Blockzeit stand die Anlage einem Benutzer exklusiv zur Verfügung. Der Benutzer musste dann in der für ihn reservierten Blockzeit in den Maschinenraum gehen und die Anlage über Tasten und Schalter des Bedienpultes betreiben.

- Der Assembler, auf Lochstreifen vorliegend, wurde ab einer vorgegebenen Speicheradresse in den Arbeitsspeicher eingelesen.
- Das zu übersetzende Anwendungsprogramm, ebenfalls auf Lochstreifen vorliegend, wurde in den Lochstreifenleser eingelegt.
- Der Assembler wurde an seiner Anfangsadresse gestartet. Er las das Anwendungsprogramm ein, übersetzte es in Maschinenbefehle und konvertierte die Daten in die maschineninterne Darstellung. Das übersetzte Anwendungsprogramm wurde am Ende der Assemblierung auf Lochstreifen ausgegeben.
- Die Lochstreifen mit den benötigten Standardprozeduren wurden aus der Programmbibliothek entnommen. Wie der Begriff Bibliothek auch nahe legt, war damals die Programmbibliothek auch wirklich durch einen Schrank mit Fächern, in denen sich die Lochstreifen mit den Standardprogrammen und Standardprozeduren befanden, realisiert.
- Das übersetzte Anwendungsprogramm und die benötigten Standardprozeduren wurden gemäß dem erstellten Speicherbelegungsplan eingelesen.
- Der Lochstreifen mit den Eingabedaten für das Programm wurde in den Lochstreifenleser eingelegt.
- Das Anwendungsprogramm wurde an der bei der Programmierung festgelegten Anfangsadresse gestartet. Während des Laufs wurden die Eingabedaten vom Lochstreifen gelesen und die Ergebnisse wieder auf Lochstreifen ausgegeben. Die Ergebnisse konnten dann später an einem Fernschreibgerät ausgedruckt werden.
- Der Beginn und das Ende der benützten Blockzeit wurden in ein Logbuch eingetragen.
- Eventuell aufgetretene Fehler wurden in ein Fehlerlogbuch eingetragen.

Probleme

Die Zusammensetzung des gewünschten Ablaufs aus solchen Einzelschritten und die Steuerung dieses Ablaufs lag, wie bereits erwähnt, voll in der Verantwortung des Benutzers. Durch diese enge Einbeziehung eines Menschen in diesen technischen Ablauf entstanden eine Reihe von Problemen, insbesondere:

- große Leerzeiten der Maschine,
- eine hohe Fehleranfälligkeit des Ablaufs und
- die Notwendigkeit, dass der Benutzer während der gesamten Auftragsbearbeitung zur Bedienung der Maschine anwesend war.

Es ist daher nicht überraschend, dass schon sehr früh Anstrengungen unternommen wurden, diese Abläufe nach Anweisung des Benutzers durch den Rechner selbst ausführen zu lassen. So entstanden erste einfache Kommandosprachen (Shells) und Betriebssysteme[1].

Definition Betriebssystem

Ein Betriebssystem steuert also gemäß den vorangehenden Ausführungen den Ablauf der Auftragsbearbeitung für einen Benutzer. Es plant die Reihenfolge der Auftragsbearbeitung für verschiedene Benutzer. Es lädt die Programme in den Arbeitsspeicher und startet sie. Es stellt Standarddienste bereit, insbesondere für den Transport von Daten zwischen Programmen und Geräten. Sind die Rechner in Rechnernetze integriert, dann ist als Standarddienst auch die Kommunikation mit anderen Programmen in anderen Rechnern notwendig. Laufen mehrere Programme gleichzeitig ab, koordiniert und synchronisiert das Betriebssystem den Zugriff auf gemeinsame Betriebsmittel. Typische gemeinsame Betriebsmittel sind alle Hardware-Komponenten und die Dateien. Ein Betriebssystem erfasst die verbrauchte Rechenleistung und registriert aufgetretene Probleme in der Hardware oder Software in einer Fehlerdatei.

Die Definition eines Betriebssystems nach DIN 44300 geht von der Beschreibung seiner Aufgabe und seiner Stellung in einer Programmhierarchie aus: Das Betriebssystem wird gebildet durch die Programme eines digitalen Rechensystems, die zusammen mit den Eigenschaften der Rechenanlage die Grundlage der möglichen Betriebsarten des digitalen Rechensystems bilden und insbesondere die Ausführung von Programmen steuern und überwachen.

In der Wikipedia Enzyklopädie (WIKI06a) findet sich die Definition: „Ein Betriebssystem ist die Software, die die Verwendung (den Betrieb) eines Computers ermöglicht. Es verwaltet Betriebsmittel wie Speicher, Ein- und Ausgabegeräte und steuert die Ausführung von Programmen. Betriebssystem heißt auf Englisch operating system (OS). Dieser englische Ausdruck kennzeichnet den Sinn und Zweck: Die in den Anfängen der Computer stark mit schematischen und fehlerträchtigen Arbeiten beschäftigten Operatoren[2] schrieben Programme, um sich die Arbeit zu erleichtern; diese wurden nach und nach zum operating system zusammengefasst. Betriebssysteme bestehen in der Regel aus einem Kern (englisch: Kernel), der die Hardware des Computers verwaltet, sowie grundlegenden Systemprogrammen, die dem Start des Betriebssystems und dessen Konfiguration dienen."

Weitere, unterschiedliche Definitionen finden sich auch in den Lehrbüchern über Betriebssysteme. In den nachfolgenden Abschnitten werden Aspekte, die ein Betriebssystem charakterisieren, detaillierter behandelt.

---

[1] Eine frühe Verwendung des Begriffs findet sich in (WIEH64).

[2] In diesem Buch wird immer der Begriff Operateur anstelle von Operator verwendet. Da es sich hier um ein Zitat handelt, wurde der dort verwendete Begriff Operator beibehalten.

1.2 Prozesse

Programmlauf

Bei der Durchführung von Aufträgen in einem Rechensystem laufen Programme ab. Ein Programmlauf führt Befehle aus, den Programmcode. Hierbei werden unveränderliche Daten (Konstanten) und veränderliche Daten (Variablen) verwendet. In vielen Systemen sind gleichzeitig mehrere Programmabläufe mit demselben Programmcode möglich. Beispielsweise könnte ein C-Übersetzer in den Arbeitsspeicher geladen sein, den mehrere Benutzer gleichzeitig für die Übersetzung ihrer C-Programme benützen. Hierbei sind der Code des Übersetzers und die Konstanten gemeinsam für alle Benutzer. Jeder Benutzer hat aber seinen eigenen Datenbereich für Programmvariablen. Für jeden Benutzer wird eine eigene Übersetzung unabhängig von den anderen Benutzern ausgeführt. Damit gibt es für jeden Benutzer einen eigenen Programmlauf. Man muss also scharf zwischen Programmcode und Programmlauf unterscheiden. Natürlich liegen auch dann getrennte Programmläufe der Benutzer vor, wenn jeder Benutzer seinen eigenen Programmcode ausführt.

Prozess

Wir nennen den Ablauf eines Programms einen Rechenprozess oder kurz Prozess, falls dieser Ablauf durch das Betriebssystem verwaltet wird. Prozesse können prinzipiell parallel ablaufen. Jedem Prozess ist ein Prozessadressraum zugeordnet. Dieser stellt die einem Prozess zugeordneten Speicheradressen dar. Beim obigen Beispiel sind der Programmcode des C-Übersetzers, die Konstanten und die prozessspezifischen Variablen im jeweiligen Prozessadressraum.

Leichtgewichtsprozess
Thread

Nicht jedem Prozess muss ein eigener Prozessadressraum zugeordnet sein. Es gibt Prozesse, die den Prozessadressraum eines anderen Prozesses mitverwenden. Solche Prozesse nennt man Leichtgewichtsprozesse oder Threads, falls sie vom Betriebssystem verwaltet werden. Gründe für die Verwendung von Leichtgewichtsprozessen sind:

- Alle Leichtgewichtsprozesse, die in einem gemeinsamen Adressraum ablaufen, können effizient und problemlos auch auf alle Daten im gemeinsamen Adressraum zugreifen. Dies ist beispielsweise bei Client-Server-Systemen interessant. Der Server (Auftragnehmer, Diensterbringer) arbeitet beispielsweise auf einem bestimmten Datenbestand. Der Client (Auftraggeber, Dienstnehmer) verlangt eine Dienstleistung vom Server, die mehrere Interaktionen zwischen Server und Client erfordert, also einen Ablauf darstellt. Ein solcher Ablauf wird sinnvollerweise durch einen Leichtgewichtsprozess im Server realisiert.
- Die Zeit zur Erzeugung eines Leichtgewichtsprozesses ist wesentlich kürzer als die zur Erzeugung eines Prozesses mit eigenem Adressraum.

Daher können Leichtgewichtsprozesse auch für kurzfristige Aufgaben erzeugt und verwendet werden.

- Wird der Rechnerkern einem Prozess mit eigenem Adressraum zugeteilt, dann müssen meist Speicherbereiche des Prozesses in den Arbeitsspeicher geladen und nicht mehr benötigte Speicherbereiche anderer Prozesse verdrängt werden. Beim Wechsel zwischen Leichtgewichtsprozessen in demselben Adressraum kann man damit rechnen, dass zumindest einige Speicherbereiche des vorhergehenden Leichtgewichtsprozesses weiter relevant sind. Es müssen also oft keine oder zumindest weniger Speicherseiten ausgetauscht werden. Dies erlaubt einen schnelleren Prozesswechsel bei geringerem System-Overhead.

Subprozess

Sonstige Abläufe, die nicht Verwaltungseinheit des Betriebssystems sind, können darüberhinaus bei geeigneter Struktur der Prozesse noch innerhalb von Prozessen ablaufen. Wir nennen solche Abläufe auch Subprozesse (bei Microsoft: Fiber (SOLO00)). Da Subprozesse keine Verwaltungseinheit des Betriebssystems sind, wird die Zuteilung von Betriebsmitteln, insbesondere die Zuteilung des Rechnerkerns, an Subprozesse nicht durch das Betriebssystem, sondern durch den Prozess selbst vorgenommen. Das Betriebssystem hat also keine Kenntnis von der Existenz von Subprozessen.

Benutzerprozess, Systemprozess

Prozesse, die Aufträge des Benutzers abwickeln, nennt man auch Benutzerprozesse. Prozesse, die ausgewählte Dienstleistungen des Betriebssystems erbringen, nennt man Systemprozesse. Systemprozesse haben oft spezielle Rechte, laufen beispielsweise im Systemmodus, und können so als Komponenten des Betriebssystems betrachtet werden. Alle nicht als Prozesse realisierten Komponenten des Betriebssystems bilden den Betriebssystemkern. Dieser wird auch als Basisschicht des Betriebssystems bezeichnet. Damit ergibt sich der prinzipielle Aufbau eines Betriebssystems gemäß Abbildung 1.1.

1.3 Objekte und Verwalter

Objektklasse

Für die Beschreibung eines Betriebssystems spielt der Begriff Objekt eine wichtige Rolle. Er wird mit teilweise abgewandelter Bedeutung auch in anderen Bereichen der Informatik verwendet. Objekte sind Hardware- oder Software-Komponenten, wie Datenstrukturen oder Prozesse. Bestimmte Objekte werden durch das Betriebssystem verwaltet. Jedes Objekt ist genau einer Objektklasse zugeordnet. Anstelle des Begriffs Objektklasse findet sich häufig auch der Begriff Objekttyp. Für eine Objektklasse sind die möglichen Operationen auf den zugeordneten Objekten definiert. Wir nennen diese Operationen auch Zugriffsdienste oder kurz Dienste.

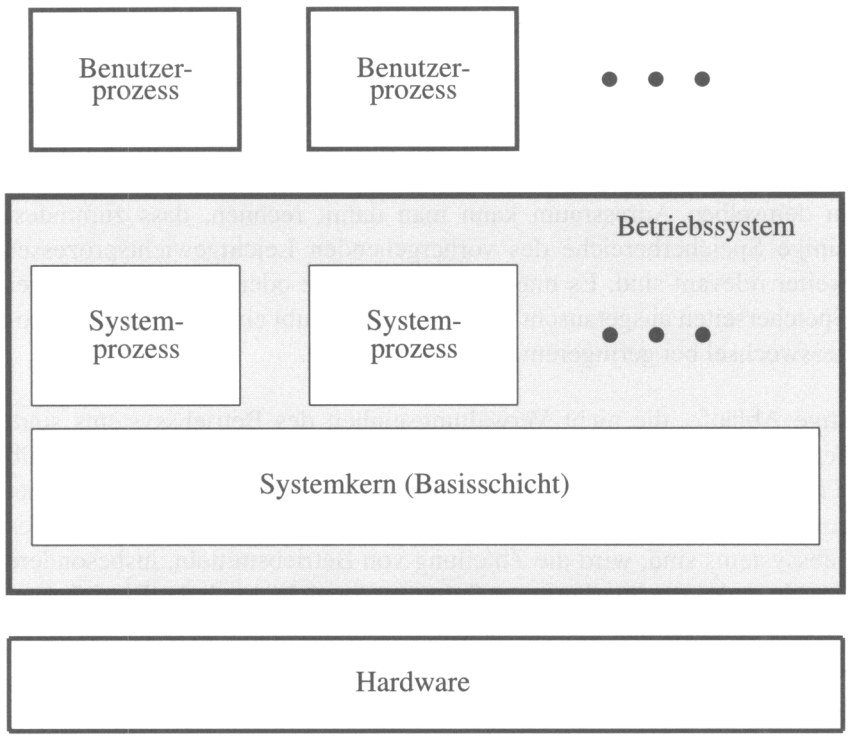

Abbildung 1.1: Prinzipieller Aufbau eines Betriebssystems

Objektverwalter

Die Betriebssystemkomponenten, die für einen bestimmten Objekttyp zuständig sind, heißen Objektmanager oder Objektverwalter bzw. Objektverwaltungen. Ein Verwalter führt Buch über den Zustand der Objekte eines Objekttyps, stellt die Dienste dafür bereit, löst Zugriffskonflikte auf und optimiert die Nutzung des Objekts.

Treiber

Verwalter für die Geräte einer Rechenanlage nennt man oft auch Gerätetreiber oder kurz Treiber. Diese Begriffsbildung ist dadurch zu erklären, dass bei den früheren einfachen Betriebssystemen die Abstraktion Verwalter und dessen vielfältige Dienste noch keine Rolle spielte. Hier sah man einen „Geräteverwalter" vorwiegend nur als eine Prozedur, die den Dienst „Transport der Daten von und zu dem Gerät" erbrachte, also das Gerät „betrieb".

Objekthierarchie

Ein Objekt einer bestimmten Objektklasse wird möglicherweise durch oder mit Hilfe von Objekten einfacherer Objektklassen realisiert. Das bedeutet, dass ein Verwalter seine Dienste auch durch Zugriff auf Dienste anderer Objektverwalter erbringt. Man erhält so eine Hierarchie von Objekten, die

aufeinander aufbauen, beginnend von einfachen, durch die Hardware definierten Objekten bis hin zu den sehr komplexen Objekten, wie Dateien. Die Objekthierarchie ist ein mögliches Strukturierungskonzept für den Entwurf von Betriebssystemen. Weiten Teilen dieses Buches liegt dieses moderne Konzept zu Grunde.

Betriebsmittel

Auch der Begriff Betriebsmittel lässt sich über Objekte definieren. Hier steht nicht der zugehörige Verwalter mit seinen Diensten im Vordergrund des Interesses, sondern die Tatsache, dass Objekte bei der Ausführung von Programmen benötigt werden. Vom Betriebssystem verwaltete Objekte, die ein Prozess für seine Arbeit benötigt, nennt man Betriebsmittel. Beispiele für Betriebsmittel sind: Komponenten der Hardware, einzelne Variablen, Dateien oder (Dienstleistungs-)Prozesse.

1.4 Aufgaben und Komponenten

Nachfolgend werden die Aufgaben und die Komponenten eines Betriebssystems ausgehend von verschiedenen Gesichtspunkten hergeleitet.

Benutzergruppen

Ein Rechensystem muss Schnittstellen für verschiedene Benutzergruppen bzw. für einen einzelnen Benutzer in verschiedenen Rollen bereitstellen. Die wichtigsten Gruppen sind Anwender, Administratoren, Bediener und das Wartungspersonal. Der Anwender ist der normale Benutzer eines Rechensystems, der Anwendungsprogramme ablaufen lässt. Der Administrator ist für die benutzerübergreifenden Einstellungen des Rechners zuständig. Er verwaltet beispielsweise die BIOS-Einstellungen, die Internetzugänge oder die Benutzer. Auch das Zufügen oder Entfernen von Software-Paketen kann in seine Zuständigkeit fallen. Zur Verwaltung der Benutzer gehört insbesondere das Zulassen der Benutzer und die Festlegung ihrer Rechte. Der Bediener (Operator, Operateur) bedient die zentralen und dem normalen Benutzer nicht zugänglichen Ein/Ausgabegeräte in einem Rechenzentrum und überwacht die Abläufe im Rechensystem. Zur Bedienung der Ein/Ausgabegeräte gehört insbesondere das Wechseln der Datenträger. Das Wartungspersonal ist für die Installation neuer Hardware und insbesondere für die Fehlerdiagnose und -behebung bei Hard- und Software verantwortlich.

Betrachten wir zwei Grenzfälle: Den privaten PC und das Rechenzentrum.

Bei einem privaten PC nimmt ein Benutzer immer alle Rollen, soweit das seine Kenntnisse erlauben, ein. Die verschiedenen Rollen erfordern je nach Einstellungen des Betriebssystems jedoch verschiedene Rechte, so benötigt er für Dienste, die den Administratoren vorbehalten sind, Administrator-Rechte. Da die Benutzeroberfläche für die verschiedenen Rollen weitgehend

gleich ist, ist es dem Benutzer oft gar nicht bewusst, dass er verschiedene Rollen einnimmt.

Wird ein Rechensystem, beispielsweise ein Höchstleistungsrechner, in einem Rechenzentrum betrieben, dann werden die genannten Rollen von unterschiedlichen Personen wahrgenommen. Der Anwender kommt typischerweise nicht direkt an die Maschine. Er kann nur sein Programm ablaufen lassen. Der Bediener ist typischerweise ein Angestellter des Rechenzentrums, der vor Ort die Rechenanlage bedient. Die Wartung erfolgt typischerweise durch Personen des Rechenzentrums und Personen des Herstellers. Es ist also leicht einzusehen, dass hier jede Benutzergruppe eines Rechensystems spezielle Bedürfnisse hat und daher eine eigene Schnittstelle benötigt. Diese muss wegen des Umfangs und der Komplexität der jeweiligen Aufgabe sehr leistungsfähig und komfortabel sein. Man denke nur an die umfangreichen Netze oder an riesige, durch Roboter bediente Datenarchive.

Schichtenmodell

Ein Betriebssystem stellt das Bindeglied zwischen der Hardware einerseits und den Benutzerprozessen andererseits dar. In den Benutzerprozessen laufen die Anwenderprogramme ab. Letztere setzen auf dem Laufzeitsystem und dem Kommandointerpreter (Shell) auf. Der Benutzer kann auch direkt Kommandos geben. Hierfür kann er einen textuell orientierten Kommandointerpreter oder eine grafische Schnittstelle (Graphical User Interface GUI) benutzen. Beispiele für solche grafischen Schnittstellen sind Windows bei Microsoft oder KDE bei Linux. Damit entsteht ein grobes Schichtenmodell gemäß Abbildung 1.2. Im Schichtenmodell sind spezielle Dienstleistungsprozesse, die im Benutzermodus arbeiten, nicht explizit dargestellt. Diese können entweder als Systemprozesse betrachtet werden oder sie können auf gleicher Ebene neben den Benutzerprozessen angeordnet werden, wenn man den Arbeitsmodus als Anordnungskriterium betrachtet. Die Dienstleistungsprozesse werden nicht von einem Benutzer gestartet, sondern von dem Betriebssystem, meist schon bei der Initialisierung. Sie erfüllen bestimmte Systemaufgaben, wie beispielsweise Anmelden (Login) der Benutzer oder Spool-Dienste.

Aufgabe Laufzeitsystem

Das Laufzeitsystem stellt die Programmierschnittstelle für den Anwendungsprogrammierer bereit (Application Programming Interface API). Das Laufzeitsystem enthält Prozeduren, die in der Regel zur Übersetzungszeit aus Bibliotheken geholt und an das Anwenderprogramm gebunden werden. Auch eine dynamische Anbindung erst zur Laufzeit ist bei gemeinsamen Bibliotheken (shared libraries) möglich. Die Systemdienste werden nie direkt vom Anwenderprogramm aus aufgerufen, sondern stets über Dienste des Laufzeitsystems. Hierdurch wird das Anwenderprogramm portabler, da es unabhängiger von der wirklichen Betriebssystemschnittstelle wird. Außerdem sind die Dienste des Laufzeitsystems so realisiert, dass möglichst

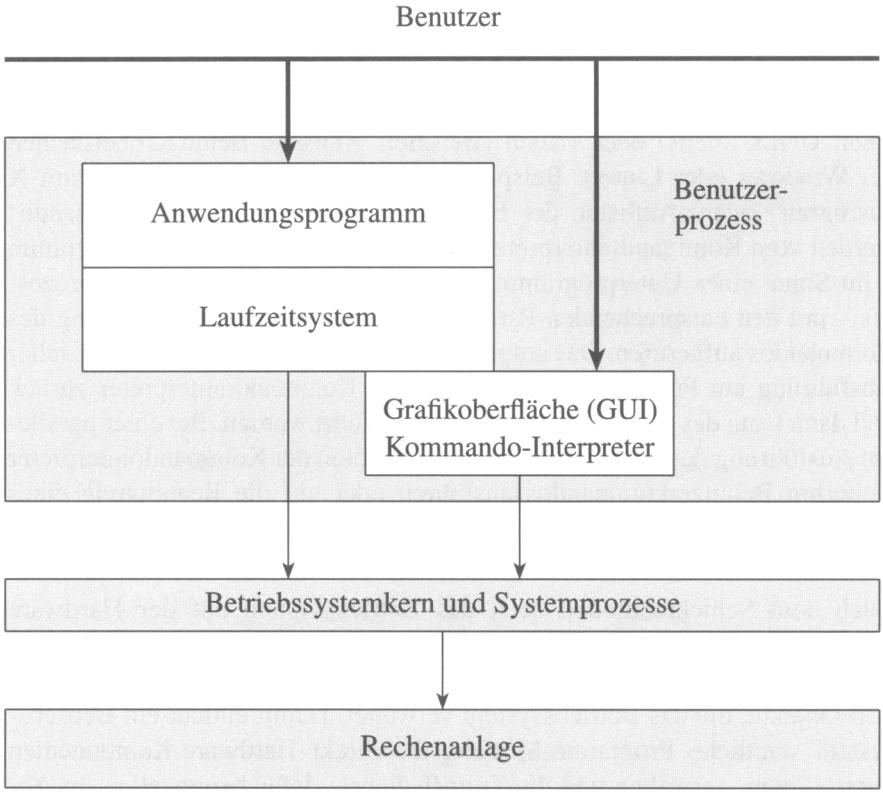

Abbildung 1.2: Grobes Schichtenmodell eines Rechensystems

selten auf das Betriebssystem zugegriffen wird. Hierdurch wird das Anwenderprogramm schneller, da der Kontextwechsel bei einem Aufruf des Betriebssystems im Vergleich zu dem eines Unterprogramms viel Zeit kostet. Ein Beispiel ist das sequenzielle, zeichenweise Lesen aus einer Datei. Hier wird sinnvollerweise nicht für jedes Zeichen das Betriebssystem aufgerufen, sondern das Zeichen wird aus einem prozesslokalen Zeichenpuffer geholt. Erst wenn der Puffer erschöpft ist, wird über einen Systemaufruf der Puffer wieder mit neuen Zeichen gefüllt. Die Festlegung der Schnittstellen des Betriebssystems und die des Laufzeitsystems sind Entwurfsentscheidungen. Für ein Betriebssystem kann es verschiedene Laufzeitsysteme mit unterschiedlichen Programmierschnittstellen geben.

Aufgabe Kommando-interpreter

Der Kommandointerpreter läuft in einem Benutzerprozess ab. Er wird bei der Initialisierung eines Benutzerprozesses nach dem Anmelden (Login) vom Betriebssystem als erste Anwendung gestartet. Er erwartet Kommandos vom Benutzer. Seine Oberfläche zum Benutzer ist entweder textuell (Beispiel: UNIX-Shells) oder visuell (Beispiel: grafische Benutzeroberflächen bei Windows oder Linux). Beispiele für Kommandos sind „Programm X ausführen" oder „Auflisten der Dateien im Ordner Y". Die Kommandos werden vom Kommandointerpreter analysiert und dann wird ein Programm – im Sinne eines Unterprogramms, eines Prozesses oder eines Subprozesses – mit den entsprechenden Parametern zur eigentlichen Ausführung des Kommandos aufgerufen. Das aufgerufene Programm kehrt bei sequenzieller Ausführung am Programmende wieder zum Kommandointerpreter zurück und dann kann das nächste Kommando ausgeführt werden. Bei einer parallelen Ausführung des Kommandos als Prozess kann der Kommandointerpreter weiterhin Benutzerkommandos ausführen oder auf die Beendigung eines von ihm gestarteten Prozesses warten. Ein Kommandointerpreter kann auch Kommandos, die in einer Datei gespeichert sind, ausführen (Shell-Skripte).

Betriebssystem als Verwalter der Hardware-Komponenten

Nach dem Schichtenmodell setzt das Betriebssystem auf der Hardware auf. Es muss daher die Dienste für die Zuteilung und die Nutzung von Hardware-Betriebsmitteln bereitstellen. Die Hardware-Komponenten sind also Objekte, die das Betriebssystem verwaltet. Damit enthält ein Betriebssystem sämtliche Programm-Module, die direkt Hardware-Komponenten kontrollieren, verwalten und die Zugriffsdienste dafür bereitstellen. In Abschnitt 3.1 werden die Hardware-Komponenten genauer vorgestellt. Wichtige Hardware-Komponenten für den Benutzer sind die Prozessoren, der Arbeitsspeicher, die Geräte und die Schnittstellen zu Rechnernetzen. Prozessoren, die Maschinenbefehle eines Benutzerprogramms ausführen, nennt man genauer auch Instruktionsprozessoren, um sie von den EA-Prozessoren abzugrenzen. EA-Prozessoren führen Datentransporte zwischen dem Arbeitsspeicher und den Geräten durch. Wir nennen Instruktionsprozessoren in diesem Buch immer Rechnerkerne.

Es gibt auch Hardware-Komponenten, die dem Benutzer nicht direkt zugänglich sind, da sie nur betriebssystemintern benötigt werden. Beispiele hierfür sind die Speicherabbildungstabellen oder die EA-Register. Für solche Hardware-Komponenten sind je nach Komplexität der Nutzung betriebssysteminterne Verwalter, einfache Dienstprozeduren oder nur Datenstrukturen mit Statusinformation vorhanden.

Betriebssystem als komfortable Programmier-umgebung

Nach dem Schichtenmodell setzen die Benutzerprozesse auf dem Betriebssystem auf. Unter diesem Gesichtspunkt ist es die Aufgabe eines Betriebssystems, eine komfortable und für die Programmierung bequeme Umgebung bereitzustellen. Dem Benutzer wird durch das Betriebssystem somit ein „vir-

tueller" Rechner bereitgestellt, der über wesentlich komplexere Objekte und Operationen als die reine Hardware verfügt. Beispiele für solche Objekte sind Prozesse, virtuelle Speicher, Dateien oder Nachrichten.

Kriterien für Objekte im Betriebssystem

Die Entscheidung, welche weiteren Objekte ein Betriebssystem anbietet, ist zum Teil durch die historische Entwicklung vorgezeichnet, primär aber auch eine Entwurfsentscheidung. Für die Entscheidung, ein Objekt im Betriebssystem und nicht im Laufzeitsystem oder Anwenderprogramm zu realisieren, sprechen folgende Punkte:

- Das Objekt ist für ein breites Spektrum von Anwendungen von grundlegender Bedeutung.
- Es ist ein gemeinsames Objekt verschiedener Benutzer.
- Es besteht Konkurrenz der Benutzer bei der Nutzung der Ressourcen für das Objekt.
- Die Zugriffe auf das Objekt müssen synchronisiert werden.

Das wichtigste Objekt, auf welches die obigen Punkte zutreffen, ist die Datei. Die Dienste für Dateien sind in der Komponente Datenverwaltung des Betriebssystems konzentriert (Kapitel 10).

Dienste variieren

Es ist jedoch beim Entwurf nicht nur die Entscheidung zu treffen, welche Objekte ein Betriebssystem enthalten soll, sondern es sind auch die Dienste festzulegen. Die Dienste variieren von System zu System stark. Erschwerend kommt hinzu, dass die Dienste in der Regel nicht sehr systematisch und objektorientiert entworfen sind und dass die Parameter nicht grundsätzlich implementierungs- und geräteunabhängig festgelegt werden. Es ist daher schwierig, allgemein gültige Dienste oder Dienstleistungsgruppen zu definieren. Die Schnittstellenproblematik wird besonders deutlich, wenn Rechner unterschiedlicher Hersteller in einem Rechnernetz betrieben werden sollen.

Betriebssystem als Prozessverwalter

Ein Betriebssystem verwaltet Prozesse. Prozesse benötigen zur Durchführung ihrer Aufgaben Betriebsmittel. Prozesse konkurrieren also um die Betriebsmittel. Das Betriebssystem hat damit aus der Sicht der Prozesse insbesondere folgende Aufgaben:

- Erzeugen und Löschen von Prozessen
- Zuteilung der Betriebsmittel an die Prozesse
- Synchronisation des Zugriffs auf die (virtuellen) Betriebsmittel

Betriebssystem als Kommunikationsmedium

In modernen Betriebssystemen ist die Kommunikation eines Prozesses mit anderen Prozessen, seien diese im gleichen Rechner oder in anderen Rechnern eines Rechnernetzes, von zentraler Bedeutung. Das Betriebssystem muss daher vielfältige Kommunikationsmechanismen zwischen Prozessen bereitstellen (Kapitel 7).

Messung Abrechnung

Ein Betriebssystem, insbesondere eines für mehrere Benutzer, sollte die verbrauchte Rechenleistung, generell die Belegung der Betriebsmittel durch die Benutzer, messen und aufzeichnen. Solche Messungen sind die Basis für eine Kostenermittlung zur Abrechnung. Weitere Messungen, insbesondere der Wartezeiten der Prozesse beim Zugriff auf Betriebsmittel und der Auslastung der Komponenten eines Rechensystems, sind für die Einstellung von Strategieparametern (Tuning) für die Betriebsmittelzuteilung erforderlich. Messungen geben auch Hinweise auf die Engpässe in einem System. Diese können möglicherweise durch Änderungen oder Erweiterungen der Konfiguration behoben werden.

Betriebssystemkern und Systemprozesse

Die Aufteilung der Aufgaben und Komponenten eines Betriebssystems auf den Betriebssystemkern und die Systemprozesse werden später in Kapitel 9 genauer behandelt. Es besteht dabei ein weiter Spielraum für Entwurfsentscheidungen.

1.5 Betriebsziele

Erklärung

Prozesse benötigen für ihre Arbeit Betriebsmittel. Gibt es in einem Rechensystem mehrere konkurrierende Prozesse, dann muss das System Strategien zur Zuteilung der Betriebsmittel enthalten. Die Strategien sollen so gewählt werden, dass bestimmte Betriebsziele erreicht werden.

einfache Betriebssysteme

Betriebssysteme, die von der Konstruktion her nur einen Benutzer unterstützen können, finden sich im Bereich der Personalcomputer; Beispiele dafür sind MS-DOS oder ältere Versionen von Windows. Solche Betriebssysteme enthalten keine Zuteilungsstrategien für Betriebsmittel. Betriebsmittel werden, sofern noch frei, dem anfordernden Prozess immer zugeteilt. Der Prozess gibt die Betriebsmittel wieder ab, wenn er diese nicht mehr benötigt. Sind benötigte Betriebsmittel nicht (mehr) vorhanden, dann ist dies ein Fehler des Benutzers.

Betriebsmittelentzug

In Betriebssystemen mit mehreren unabhängigen Prozessen, seien diese für einen oder verschiedene Benutzer arbeitend, treten die Prozesse in Konkurrenz um die Betriebsmittel. Die Prozesse fordern die Betriebsmittel vom Betriebssystem an. Dieses teilt den Prozessen nach einer Zuteilungsstrategie die geforderten Betriebsmittel zu. Hierbei erhält der Prozess das Betriebsmittel oft so lange bis er es freiwillig wieder abgibt. Manche Betriebsmittel sind aber so wertvoll, dass man in bestimmten Situationen nicht warten kann,

bis der Prozess das Betriebsmittel freiwillig abgibt. Eine solche Situation liegt vor, wenn ein wichtigerer Prozess auf das Betriebsmittel wartet. Daher sind für bestimmte Betriebsmittel, beispielsweise die Rechnerkerne, die Hardware-Schnittstellen und die Systemdienste so gestaltet, dass das Betriebsmittel dem momentanen Besitzer zu einem beliebigen Zeitpunkt ohne Schaden entzogen werden kann. Der Entzug von Betriebsmitteln ist daher ebenfalls Bestandteil einer Betriebsmittelverwaltung und wird durch Verdrängungsstrategien gesteuert.

Betriebsziele

Das Zusammenwirken aller im Betriebssystem vorhandenen Strategien bestimmt letztendlich die Bearbeitungsfolge und den Fortschritt der Aufträge in dem Rechensystem. Die Strategien müssen daher so gewählt und ihre Parameter so eingestellt werden, dass eine möglichst optimale Anpassung an die Betriebsziele des Betreibers der Rechenanlage erreicht wird. Typische Betriebsziele sind hohe Auslastung der Komponenten oder gute Reaktionszeiten. Die Akzeptanz von Reaktionszeiten durch den Benutzer orientiert sich dabei an dem Umfang der verlangten Aufgabe. Für sehr kleine Aufgaben, wie die interaktive Bearbeitung von Texten, werden nicht wahrnehmbare Reaktionszeiten von Bruchteilen einer Sekunde verlangt. Die erforderlichen Reaktionszeiten sind also von Eigenschaften der Benutzeraufträge abhängig.

Konflikt

Die Betriebsziele stehen in Konflikt zueinander:

- Die Bevorzugung kleiner Aufträge bedingt eine Behinderung großer Aufträge.
- Eine gute Auslastung der Komponenten bedeutet schlechte Reaktionszeiten.

Dieser Sachverhalt lässt sich bereits an dem einfachen Beispiel einer Autovermietung einsehen:

- Gute Auslastung der Mietwagen bedeutet, dass praktisch immer alle Wagen im Einsatz sind. Eine gute Reaktionszeit bedingt aber, dass jederzeit ausreichend viele Mietwagen des gewünschten Modells zur Vermietung verfügbar sind.
- Werden bestimmte Kunden bevorzugt bedient und ist viel Betrieb, dann müssen andere Kunden dafür länger warten.

Die quantitative Untersuchung solcher Phänomene kann mit den Verfahren der Warteschlangentheorie bzw. durch simulative Methoden erfolgen. Siehe hierzu Kapitel 15.

1.6 Anforderungen und Grundkonzepte

An ein Betriebssystem stellt man typische Anforderungen. Zu ihrer Erfüllung wurden im Laufe der Jahre Grundkonzepte entwickelt. Eine Auswahl charakteristischer Anforderungen und daraus resultierende Grundkonzepte werden nachfolgend erläutert.

Spektrum der Dienste

Ein Betriebssystem muss normalerweise vielfältige Anwenderprogramme unterstützen. Deshalb müssen die von einem Betriebssystem unterstützten Objekte und die zugehörigen Dienste so umfassend, allgemein und vollständig sein, dass sie das bekannte und zukünftig zu erwartende Spektrum der Anwendungen gut abdecken können. Es ist daher wichtig, dass die Objekte in Schichten angeordnet sind und dass möglichst nur die oberste Schicht der Objekte und Dienste anwendungsabhängig ist. Tiefere Schichten müssen anwendungsneutral sein. Damit kann man bei Bedarf leichter neue, anwendungsorientierte Objekte in der obersten Schicht zufügen, ohne dass der Rest des Betriebssystems geändert werden muss.

problemorientierte Parameter

Die Dienste eines Betriebssystems, auch die nur betriebssystemintern genutzten Dienste tieferer Schichten, sollen leicht verständlich sein und soweit möglich gleich oder ähnlich bei verschiedenen Betriebssystemen. Sie sollten auch bei Weiterentwicklungen des Betriebssystems möglichst lange stabil bleiben. Deshalb ist ein wichtiger Aspekt bei der Definition der Dienste und ihrer Parameter, dass diese problemorientiert (benutzerorientiert) und nicht implementierungsorientiert sind.

Zuteilung Betriebsmittel Trennung Mechanismus und Strategie

Ein Betriebssystem, das den gleichzeitigen Ablauf mehrerer Prozesse unterstützt, muss den Prozessen Betriebsmittel zuteilen. Hierzu sind, wie in Abschnitt 1.5 bereits andiskutiert, Zuteilungs- und Verdrängungsstrategien erforderlich. Bei Betriebssystemen, die mehrere Benutzer, insbesondere Benutzer verschiedener Benutzerklassen, unterstützen, müssen die Strategien besonders ausgefeilt sein und ausreichend viele Parameter enthalten, damit eine Anpassung an die lokalen Betriebsziele möglich ist. Der Betriebssystemkern sollte deshalb möglichst nur die Mechanismen einer Grundstrategie implementieren. Darauf aufsetzend sollten dann in höheren Schichten unterschiedlichste Strategien realisierbar sein.

Zugriffsschutz

Der Zugriff auf Objekte eines Betriebssystems, beispielsweise auf Dateien oder auf Arbeitsspeicherbereiche, muss kontrolliert werden, da die Objekte benutzerspezifische Daten enthalten, die nicht allgemein zugänglich sein dürfen, oder da es nicht zulässig sein darf, dass ein Benutzer Daten eines anderen Benutzers löschen kann. Einzelne Benutzer haben daher unterschiedliche Zugriffsrechte auf ein Objekt. Beispiele für Zugriffsrechte sind: Leserecht, Schreibrecht oder Ausführungsrecht. Das Ausführungsrecht bedeutet, dass beispielsweise eine Datei ein Programm enthält, das

der Benutzer ausführen darf. Zu der Beschreibung eines Objekts gehört daher eine Zugreiferliste. Dies ist eine Liste der Benutzer und ihrer Zugriffsrechte bezüglich des Objekts. In manchen Systemen, beispielsweise in UNIX, wird der Zugriff nur für Benutzergruppen und nicht für einzelne Benutzer differenziert. Es gibt sogar Systeme, bei denen überhaupt kein benutzerspezifischer Zugriffsschutz möglich ist. Dies stellt eine ernsthafte Sicherheitslücke dar. Man sollte daher auch in den Systemen, die nur einen einzigen Benutzer unterstützen, differenzierte Zugriffsrechte vorsehen. So entsteht ein Schutz gegen Viren und eigene Fehler, beispielsweise wenn durch einen Fehler in einem Programm versucht wird, Dateien des Betriebssystems zu überschreiben.

langfristige Datenhaltung

Durch ein Betriebssystem müssen permanente, d.h. langlebige Datenbestände der Benutzer gehalten werden. Diese Datenbestände sind gegen unberechtigten Zugriff und gegen Verlust zu schützen.

Synchronisation

Ein Betriebssystem steuert, von einfachsten Systemen abgesehen, viele parallel ablaufende Vorgänge (Prozesse). Dies sind einerseits parallele Abläufe in der Hardware und andererseits parallele oder quasiparallele Abläufe in der Software. Diese Vorgänge müssen beim Zugriff auf gemeinsame Betriebsmittel synchronisiert werden.

Speicherhierarchien

Ein Betriebssystem muss eine Hierarchie von Informationsspeichern verwalten. Der Grund dafür ist, dass schnelle Speicher deutlich teurer und kleiner sind als langsame. Daten, die häufig gebraucht werden, müssen unbedingt auf schnellen Speichern liegen; Daten, die selten gebraucht werden, können auf langsame Massenspeicher ausgelagert werden. Typisch ist eine vierstufige[1] Hierarchie bestehend aus:

- den Archivsystemen und Massenspeichern, beispielsweise den Magnetbändern, Bandkassetten, CDs oder DVDs.
- dem Hintergrundspeicher, insbesondere dem Festplattenspeicher
- dem Arbeitsspeicher
- den Cache-Speichern in der Hardware

Der Cache-Speicher ist der schnellste und kleinste Speicher in dieser Hierarchie. Im Cache-Speicher des Rechnerkerns wird beispielsweise ein aktueller Ausschnitt der Daten im Arbeitsspeicher gehalten. Cache-Speicher werden von der Hardware verwaltet, alle anderen Speicher durch das Betriebssystem. Datenbestände bzw. Ausschnitte aus Datenbeständen werden durch

[1] Die Register des Rechnerkerns sind bei dieser Betrachtung ausgeklammert, da sie allein vom Rechnerkern verändert werden und so das Betriebssystem im Rahmen der Verwaltung von Speicherhierarchien darauf keine Rücksicht nehmen muss.

das Betriebssystem (automatisch) zwischen Speicherhierarchien bewegt. Ein Beispiel für einen automatischen Transport von Daten zwischen Festplattenspeicher und Arbeitsspeicher durch das Betriebssystem ist der dynamische Seitenwechsel.

Evolution

Ein Betriebssystem bzw. eine Betriebssystemlinie hat eine Lebensdauer von vielen Jahren. In dieser Zeit ändern sich die Dienstleistungen und die Betriebsziele möglicherweise erheblich. Es ist daher eine konsequente Modularisierung, beispielsweise gemäß dem Konzept der Verwalter, unbedingte Voraussetzung. Nur so kann der Aufwand für Änderungen in einem erträglichen Rahmen gehalten werden. Da Änderungen auch den Benutzer betreffen können, ist es erforderlich, auch dessen Aufwendungen bei Änderungen zu minimieren. Eine Methode besteht darin, den Anwendungsprogrammen keine direkten Zugriffe auf das Betriebssystem zu erlauben, sondern die Betriebssystemdienste nur über Bibliotheken des Laufzeitsystems verfügbar zu machen. Damit können ältere Schnittstellen beibehalten werden, wenn sie durch Bibliotheksprogramme auf die neuen Schnittstellen des Betriebssystems abgebildet werden. Man hat also eine Entkopplung der Betriebssystemschnittstelle für die Anwendungsprogramme von der realen Betriebssystemschnittstelle. So bleibt bei Änderungen der realen Betriebssystemschnittstelle das Anwenderprogramm in der Regel unverändert und muss nur neu gebunden werden.

Selbstadaption
Plug and Play

Das Betriebssystem hat direkte Schnittstellen zur Hardware. Die Konfiguration der Rechenanlage ändert sich aber von System zu System und auch im Laufe der Zeit. Typische Änderungen, auf die ein Betriebssystem eingerichtet sein muss, sind u.a.: verschieden große Arbeitsspeicher, wechselnder Anschluss der Geräte an die Kanäle, unterschiedliche Geräte und unterschiedliche Treiberkarten für Geräte. Ein Betriebssystem muss also leicht an ein breites Spektrum von Konfigurationen angepasst werden können. Gewünscht wird, dass beim Start des Betriebssystems die vorhandenen Komponenten der Hardware festgestellt und ihre Eigenschaften abgefragt werden können. So kann sich ein Betriebssystem selbständig an die jeweilige Konfiguration anpassen. Zur Laufzeit kann sich dieser Vorgang dynamisch wiederholen, wenn neue Hardware angeschlossen oder vorhandene entfernt wird. Dies setzt natürlich das Vorhandensein entsprechender Software-Module (Objektverwalter) im Betriebssystem voraus. Erleichtert wird die Adaption u.a. durch Konfigurationstabellen und Virtualisierung.

Konfigurationstabelle

Ein Betriebssystem enthält die aktuelle Konfiguration der Rechenanlage konzentriert in Konfigurationstabellen. Hier stehen sowohl die vorhandenen Komponenten als auch ihre Eigenschaften in einer möglichst anwenderorientierten Form. Alle Komponenten des Systems entnehmen die konfigurations- bzw. gerätespezifischen Daten solchen Konfigurationstabellen.

Virtualisierung

Ein Betriebssystem virtualisiert die Hardware-Komponenten, insbesondere die Geräte. Geräte mit ähnlichen Eigenschaften werden zu einer Klasse zusammengefasst. Man erhält so virtuelle Geräte. Virtuelle Geräte werden oft auch logische Geräte genannt. Die Eigenschaften der virtuellen Geräte stellen eine Abstraktion der Eigenschaften realer Geräte dar. Sie können eine Obermenge oder auch nur eine Teilmenge der Eigenschaften realer Geräte enthalten. Ein Beispiel ist das virtuelle Gerät „virtueller Festplattenspeicher". Hier sind beispielsweise die Anzahl der logischen Blöcke und ihre Größe durch den Anwender unter Einschränkungen wählbar. Die logischen Blöcke sind fortlaufend nummeriert. Ein solcher virtueller Festplattenspeicher wird auf die Spuren bzw. Sektoren eines oder mehrerer realer Festplattenspeicher abgebildet.

Beschreibungstabellen für Geräte

Es existieren Abbildungen zwischen den Eigenschaften virtueller Geräte und den Eigenschaften zugeordneter realer Geräte. Der Benutzer und die oberen Schichten des Betriebssystems arbeiten immer mit einem virtuellen Gerät. Sie benützen also nur die Eigenschaften des virtuellen Geräts und sind damit bezüglich einer ganzen Geräteklasse grundsätzlich von den realen Geräteeigenschaften unabhängig. Erst unmittelbar vor dem Zugriff auf das reale Gerät wird das virtuelle Gerät auf ein reales Gerät abgebildet. Genauso werden die Eingaben von einem realen Gerät sofort in diejenigen des virtuellen Geräts abgebildet. Werden die einzelnen Abbildungen jeweils durch eigene Programmsequenzen realisiert, so ist bei jeder Änderung auch eine Änderung des Programms notwendig. Dies bedeutet hohen Aufwand, Fehleranfälligkeit und lange Reaktionszeiten. Besser ist eine Lösung, bei der das Programm unabhängig von speziellen Geräten einer Geräteklasse ist, und alle Abbildungsvorschriften durch spezielle Konfigurationstabellen, die Geräte-Beschreibungstabellen, festgelegt werden. Die Geräte-Beschreibungstabellen sollten ebenfalls sehr anwenderorientiert aufgebaut sein.

Einheitliche Schnittstellen

Die Virtualisierung hat neben der genannten Geräteunabhängigkeit auch noch das Ziel, für den Anwender besser geeignete und bequemer zu nutzende Eigenschaften zu erhalten. Man kann dies auch als Bildung neuer abstrakter, aber noch sehr Hardware-orientierter Objekttypen betrachten. Damit wird die Virtualisierung ein zentrales und wichtiges Prinzip bei der Konstruktion von Betriebssystemen. Sie wird nicht nur in den Hardware-nahen Schichten mit Erfolg eingesetzt, sondern auch bei höheren Objekten. Ein Beispiel hierfür ist die Dateischnittstelle in UNIX: Der Zugriff auf Dateien und der Zugriff auf Ein/Ausgabegeräte, wie Drucker, erfolgt im Benutzerprogramm mit denselben Systemdiensten.

Portabilität

Ein Betriebssystem muss meist auf unterschiedlichen Hardware-Plattformen laufen. Durch folgende Maßnahmen wird die Portierung auf wechselnde Plattformen erleichtert:

- Virtualisierung der Geräte und der Programmierschnittstelle der Hardware;
- absolut Hardware-unabhängige Programmierung möglichst vieler Teile des Betriebssystems, beispielsweise steht in vordefinierten Konstanten wo sich bestimmte EA-Register befinden, wie diese aufgebaut sind, wie lang eine ganze Zahl ist, wie viele Bits ein Byte enthält oder wie ein bestimmtes Zeichen, das über die Taststur eingegeben wird, maschinenintern heißt;
- Konzentration der verbleibenden Hardware-abhängigen Teile in ein Interface-Modul oder in die Hardware beschreibende Dateien zum Einbinden bei der Übersetzung des Betriebssystems.
- Einsatz höherer Sprachen und weiterer Werkzeuge bei der Realisierung des Betriebssystems.

Kommunikation Netze

Rechner sind heute in vielfältige Netze integriert. Damit erhält die Kommunikation zwischen Prozessen in einem Rechner und zwischen Prozessen in verschiedenen Rechnern hohe Bedeutung. Die Protokolle für die Kommunikation sind je nach dem Einsatzgebiet, der Ausstattung des Partners und den geforderten Qualitätsmerkmalen ganz unterschiedlich. In einem Betriebssystem muss daher eine Vielfalt von Protokollen in verschiedenen Protokollebenen verfügbar sein. Geeignete Protokolle sollten durch das System automatisch bei einer Kommunikation ausgewählt werden. Besonderer Aufmerksamkeit muss der möglichst guten Nutzung der Bandbreite (Datenrate) geschenkt werden. Durch die Protokolle und andere Leistungsverluste in dem Betriebssystem kann die physikalisch mögliche Datenrate des Netzes nicht voll ausgenutzt werden. Daher sind besonders leistungsfähige Konzepte für die Kommunikation erforderlich.

Client-Server-Modell

In einem Rechnernetz können die Betriebsmittel, die ein Prozess benötigt, lokal sein oder sie können sich irgendwo im Rechnernetz befinden. Ein Prozess benutzt dann zum Zugriff auf diese entfernten Betriebsmittel die Dienste eines anderen Prozesses im Rechnernetz. Der eine Prozess ist dann Auftraggeber (Client), der andere Prozess ist Auftragnehmer (Server). Man kommt so zu dem Client-Server-Modell zur Beschreibung von Beziehungen zwischen Prozessen. Für die Abwicklung solcher Beziehungen haben sich bestimmte Protokolle etabliert, beispielsweise der entfernte (abgesetzte) Prozeduraufruf (RPC, remote procedure call).

2 Klassifizierung

Es gibt ein sehr weites Spektrum des Einsatzes von Rechensystemen. Damit gibt es auch eine große Vielfalt an Betriebssystemen. Diese unterscheiden sich in ihrem Anwendungsbereich, in ihrer Leistung und in den Realisierungskonzepten. Allerdings sind diese drei Aspekte nicht unabhängig voneinander. Für bestimmte Leistungs- und Anwendungsbereiche haben sich bestimmte Realisierungskonzepte entwickelt. Die nachfolgende Klassifizierung berücksichtigt nur wenige Merkmale. Es handelt sich also um eine sehr grobe Klassifizierung. Typische Konzepte und Eigenschaften der Klassen werden knapp geschildert. Hierbei wird auf ein Mehrbenutzerbetriebssystem Bezug genommen. Ein solches Betriebssystem liegt den Ausführungen in diesem Buch zu Grunde.

Die Klassifizierung spiegelt auch die historische Entwicklung der Rechensysteme wider, so dass einige Klassen in heutigen Rechensystemen nur noch eine untergeordnete Rolle spielen. Die Klassifizierung wird für nicht vernetzte Rechensysteme vorgestellt. Sie kann analog auf (vernetzte) verteilte Systeme übertragen werden.

2.1 Mehrbenutzer-Mehrprozesssysteme

mehrere Prozesse

Ein Mehrbenutzer-Mehrprozesssystem liegt allen Ausführungen in diesem Buch zu Grunde. Das Betriebssystem unterstützt die gleichzeitige oder quasigleichzeitige Ausführung vieler Prozesse für mehrere Benutzer. Dabei können einem Benutzer mehrere Prozesse gehören. Hierdurch kann dieser an mehreren Teilaufgaben gleichzeitig arbeiten. Außerdem können Systemaufgaben, wie die Abwicklung von Kommunikationsprotokollen, parallel zu den Arbeiten der Benutzer durchgeführt werden.

Gründe für Prozesse

Parallelarbeit, also das Vorhandensein von Prozessen, ist u.a. bei folgenden Situationen erforderlich:

- Das Rechensystem muss in ein Rechnernetz integriert werden. Die Realisierung der Dienste und der Protokolle für die Kommunikation erfordern für eine akzeptable Realisierung ein Mehrprozesssystem.

- Der Benutzer möchte während vollständig definierter und langsam ablaufender Vorgänge, beispielsweise während einer Druckausgabe, einer Übersetzung, einer Druckaufbereitung, einer grafischen Ausgabe oder einer Dateiübertragung zwischen Rechnern, eine andere Arbeit durchführen. Auch hier sind akzeptable Lösungen nur bei mehreren Prozessen realisierbar.
- Der Benutzer benötigt für seine Arbeit gleichzeitig verschiedene Informationsquellen oder er arbeitet an mehreren (Teil-)Aufgaben parallel oder im Wechsel. Solche Arbeitstechniken sind bei vielen Anwendungen erforderlich, beispielsweise bei der rechnergestützten Konstruktion oder bei der Büroautomatisierung.

Anmerkung: Die Realisierung von parallelen Abläufen nur mit Leichtgewichtsprozessen oder mit Subprozessen ist keine Lösung, da die Abläufe bei verschiedenen Anwendungen nicht genügend unabhängig wären und Systemabläufe in Benutzerprozesse integriert werden müssten. Bestimmte parallele Abläufe kann man in Betriebssystemen also nur als Prozesse mit eigenem Kontext (Schwergewichtsprozesse) realisieren.

mehrere Benutzer

Die gleichzeitige Nutzung des Systems durch mehrere Personen erfordert entsprechende Eigenschaften des Betriebssystems (siehe Kapitel 1), wie beispielsweise erhöhte Anforderungen an die Betriebsmittelverwaltung und die Sicherheit. Solche Eigenschaften sind immer in hohem Maße wünschenswert, selbst dann, wenn nur ein Benutzer das System privat betreibt. Beispiele für solche Betriebssysteme bei PCs sind UNIX mit seinen Varianten und die neuen Microsoft Betriebssysteme. Auch Betriebssysteme für klassische Universalrechner und Server sind – evtl. mit Einschränkungen – Mehrbenutzer-Mehrprozesssysteme.

2.2 Einprozesssysteme

ein Ablauf

Die Prozesse bilden die parallelen Abläufe in einem Betriebssystem. Ein Einprozesssystem kann dementsprechend immer nur einen Ablauf im System unterstützen. Es gibt damit keine Konkurrenz um Betriebsmittel, da alle Abläufe streng sequenzialisiert werden. Ein solches System ist also nach heutigen Maßstäben höchstens für einen einzelnen Benutzer geeignet. Hierbei besteht für ihn aber die – eigentlich nicht akzeptable – Einschränkung, dass er keine parallelen Tätigkeiten ausführen kann. So kann er beispielsweise nicht Dateien drucken und gleichzeitig noch Texte im Rechner bearbeiten.

einfacher Aufbau

Ein Einprozesssystem für einen Benutzer kann allerdings wesentlich einfacher aufgebaut sein als ein Mehrbenutzerbetriebssystem. Es entfallen beispielsweise die Prozessverwaltung, die Rechnerkernverwaltung, die In-

terprozesskommunikation, die Synchronisation beim Zugriff auf Daten, alle Strategien für die Zuteilung oder den Entzug von Betriebsmitteln, der Zugriffsschutz und alle Einrichtungen zur Behandlung oder Vermeidung von Verklemmungen. In der Regel werden zusätzlich auch alle noch verbleibenden Funktionen des Betriebssystems deutlich einfacher und mit mehr Einschränkungen verbunden sein. Damit sind solche Systeme nur für spezielle Rechensysteme oder geringe Anforderungen geeignet. Ein Beispiel ist das Betriebssystem MS-DOS.

2.3 Einbenutzersysteme

Grenzfall Mehrbenutzersystem

Wir können Einbenutzersysteme auf der Basis eines Ein- oder eines Mehrprozesssystems realisieren. Einbenutzersysteme auf der Basis von Einprozesssystemen wurden oben besprochen. Ein Einbenutzersystem auf der Basis eines Mehrprozesssystems könnte einfach als Grenzfall eines Mehrbenutzersystems betrachtet werden, in dem immer nur ein Benutzer zugelassen wird. Damit wäre zwischen einem Einbenutzersystem auf der Basis eines Mehrprozesssystems und einem Mehrbenutzersystem kein signifikanter Unterschied mehr. Dies ist in der Praxis oft so, beispielsweise wenn ein PC immer durch denselben Benutzer betrieben wird.

Falls ausdrücklich ein Einbenutzerbetriebssystem entwickelt werden soll, muss die Tatsache, dass es stets nur einen Benutzer gibt, so weit wie irgendwie möglich ausgenutzt werden. Hierbei ist das Ziel, ein einfaches, kleines und schnelles Betriebssystem zu erreichen. Hierzu gibt es die nachfolgend geschilderten Ansatzpunkte.

keine Parallelarbeit

Es wird auf Parallelarbeit auf der Ebene der Anwenderprogramme verzichtet. Dann gibt es dort nur einen sequenziellen Ablauf, in den Unterbrechungsbehandlungen eingeschoben werden. Spezielle Dienste, die als Bestandteil des Betriebssystems betrachtet werden, können im Rahmen von Unterbrechungsbehandlungen schrittweise parallel zu den Anwendungen ablaufen. Ein Beispiel für einen solchen speziellen Dienst ist das Drucken parallel zu den Anwenderprogrammen. Dieser Weg ist allerdings unbefriedigend (siehe die oben aufgeführten Gründe für Mehrprozesssysteme).

Vereinfachung trotz Parallelarbeit

Da der Verzicht auf Prozesse demnach kein sinnvoller Weg ist, betrachten wir nachfolgend nur noch Einbenutzersysteme mit mehreren Prozessen. Damit bleiben nur noch wenige Bereiche übrig, die für eine Vereinfachung des Betriebssystems für professionelle Arbeitsplätze in Frage kommen. Wichtige Beispiele sind: die Betriebsmittelverwaltung einschließlich der Zuteilungsstrategien, der Zugriffsschutz sowie die Reduktion der Funktionalität, beispielsweise bei den Fehlerbehandlungen, bei der Virtualisierung oder bei den Dateisystemen.

Vereinfachungen bei Betriebsmittelzuteilung

Da alle Abläufe in dem Rechner durch einen einzigen Benutzer veranlasst werden, können Konflikte bei der Konkurrenz um Betriebsmittel durch den Benutzer gelöst werden. Damit entfallen alle Strategien zur Festlegung der Bearbeitungsfolge von Aufträgen und zur Zuteilung von Betriebsmitteln. Allerdings ist in größeren Systemen ein solches Vorgehen für den Benutzer nicht erfreulich. Er hat große Probleme, wenn er sich beim Einsatz vieler großer und unterschiedlicher Anwenderprogramme immer im Voraus klar werden muss, ob und welche Konflikte auftreten können.

Vereinfachungen beim Zugriffsschutz

In einem Einbenutzersystem hat der Benutzer volle Kontrolle über alle angeschlossenen Peripheriegeräte. Da auch alle gespeicherten Daten einem einzigen Benutzer gehören, könnte prinzipiell auf alle Einrichtungen zum Zugriffsschutz in dem Rechensystem verzichtet werden. Es wären also auch keine privilegierten Befehle erforderlich. Die Erfahrung hat jedoch gezeigt, dass ein völliger Verzicht auf den Zugriffsschutz zu einem sehr fehleranfälligen und unsicheren Arbeiten führt. Der Benutzer ist nämlich sehr stark daran interessiert, sich vor eigenen Fehlern zu schützen. Beispiele:

- Der Benutzer möchte wichtige Daten vor versehentlichem Überschreiben schützen.
- Der Benutzer möchte sicher sein, dass seine Ergebnisse korrekt sind. Grundvoraussetzung hierzu ist, dass alle System- und Anwenderprogramme richtig arbeiten. Eine Voraussetzung hierzu ist, dass auch diese Programme gegen unbeabsichtigte Veränderungen, beispielsweise auf Grund eines aufgetretenen Fehlers oder einer Fehlbedienung, geschützt sind. Damit kann in einem Einbenutzersystem zwar auf eine ausgefeilte Identifizierung der Benutzer und auf Maßnahmen gegen hoch entwickelte Angriffe auf das Schutzsystem verzichtet werden, es ist aber zumindest ein einfacher Zugriffsschutz erforderlich.

Systeme ohne guten Zugriffsschutz sind außerdem sehr anfällig gegen Computerviren und Trojanische Pferde.

Reduktion der Funktionalität

Eine generelle Reduktion der Funktionalität der Dienste eines Einbenutzerbetriebssystems ist nicht möglich, da auch dort große und anspruchsvolle Anwendersysteme unterstützt werden müssen, die entsprechend komplexe Dienste des Betriebssystems erfordern. Auch werden für spezielle Systeme nur dann spezielle Anwendungen verfügbar sein, wenn der Markt genügend groß ist. Damit sind die Schnittstellen des Betriebssystems zur Anwendung nicht mehr frei festlegbar.

Konsequenz

Die Ausführungen zeigen, dass es nicht sinnvoll ist, spezielle Einbenutzersysteme zu entwickeln und einzusetzen, sondern auch für persönliche Rechner mit nur einem einzelnen Benutzer Mehrbenutzerbetriebssysteme zur Verfügung zu stellen.

2.4 Stapelverarbeitende Systeme

Bedeutung

Stapelverarbeitung war die typische und nahezu ausschließliche Betriebsform für Universalrechner in den 60er Jahren. Die Stapelverarbeitung ist auch heute noch bei Höchstleistungsrechnern eine wichtige Betriebsform.

Definition

Bei der Stapelverarbeitung wird eine Folge von Stapelaufträgen sequenziell oder parallel bearbeitet. Ein Stapelauftrag ist ein vollständig definierter Auftrag eines Benutzers mit allen Steueranweisungen, Programmen und Daten. Er kann ohne Eingriffe des Benutzers bearbeitet werden. Damit erfolgt eine zeitliche Entkopplung der Anwesenheit des Benutzers von der Bearbeitung seines Auftrags. Die Bearbeitung kann also nach Strategien eines Rechenzentrums, beispielsweise mit dem Ziel einer guten Auslastung des Rechners, frei gewählt werden.

Der Begriff stammt aus der Zeit der Lochkartenverarbeitung. Die Steueranweisungen, Programme und Daten lagen auf Lochkarten vor. Der Auftrag eines Benutzers bestand also aus einem Lochkartenstapel. Nachfolgend sind einige wichtige Varianten von Betriebssystemen für die Stapelverarbeitung aufgeführt.

einfaches, bandorientiertes System

Ein einfaches, bandorientiertes Stapelverarbeitungssystem der 60er Jahre bearbeitet eine Folge von Stapelaufträgen sequenziell entsprechend der Reihenfolge der Stapel im Lochkartenleser. Dieser realisiert also die Eingabewarteschlange eines solchen Systems. Das System besitzt keinen Festplattenspeicher und enthält daher benutzerspezifische Daten nur auf Magnetbändern. Vor Beginn jedes neuen Auftrags wird das System initialisiert. Die Magnetbänder mit den benutzerspezifischen Daten werden aufgespannt. Es werden dann die aufeinander folgenden Karten (Eingabezeilen) eines Auftrags nach Bedarf gelesen. Hierbei werden die Steueranweisungen ausgeführt und die zugehörigen Daten verarbeitet. Bei der Bearbeitung des Auftrags erfolgt die Ausgabe der Ergebnisse schritthaltend auf den Drucker und/oder die Magnetbänder. Am Ende des Auftrags werden die Magnetbänder des Benutzers ausgeschleust. In die Bearbeitung eines Auftrags kann also keine andere Auftragsbearbeitung eingeschoben werden. Die Aufträge werden so in der Reihenfolge des Eintreffens bearbeitet. Die Strategie wird auch als FCFS-Strategie bezeichnet (first come first served). Auf das Eintreffen neuer Aufträge kann erst nach Ende eines Auftrags reagiert werden. Aufträge werden also nicht unterbrochen. Man nennt eine solche Auftragsbearbeitungsstrategie eine nichtpräemptive Strategie. Da immer nur ein Auftrag in Bearbeitung ist, kann das Betriebssystem vom Typ Einprozessbetriebssystem sein.

Erweiterung durch Spooling

Ein offensichtlicher Nachteil dieser sehr einfachen Stapelverarbeitung ist, dass auf Grund der sequenziellen Arbeitsweise die Maschine schlecht ausgelastet wird. Während der Datentransporte zwischen dem Arbeitsspeicher und den langsamen zeichenorientierten Geräten treten große Wartezeiten des Rechnerkerns auf. Dies lässt sich durch das so genannte Spooling beheben. Das Akronym SPOOL kommt von „simultaneous pheripheral operation on-line". In einem Stapelverarbeitungssystem mit Spooling wird ausgenutzt, dass das Lesen und Schreiben auf die blockorientierten Magnetbänder schneller ist als auf zeichenorientierte Geräte. Es gibt also im Wesentlichen drei parallele Abläufe (Abbildung 2.1). In einem Eingabe-Spoolprozess werden Lochkartenstapel vom Lochkartenleser gelesen und der Inhalt wird unverändert in eine Eingabedatei auf einem Magnetband geschrieben. In einem Ausgabe-Spoolprozess werden die Zeichen aus einer Ausgabedatei, die auf Magnetband gespeichert wurde, auf einem Drucker oder Lochkartenstanzer ausgegeben. In dem eigentlichen Bearbeitungsprozess werden die Stapelaufträge ausgeführt. Hierbei werden Leseaufträge für den Lochkartenleser auf Leseaufträge für Zeilen der Eingabedatei auf Magnetband abgebildet und entsprechend werden auch Ausgabeaufträge für den Drucker oder Lochkartenstanzer auf Schreibaufträge für die Ausgabedatei abgebildet. Dies ist ein frühes Beispiel für die Virtualisierung von Geräten.

Für die Spoolprozesse wurden vielfach auch eigene kleine Rechner verwendet. Natürlich können die Daten mehrerer Benutzer auf dem Eingabeband und auf dem Ausgabeband gespeichert werden, so dass die Bänder nur gewechselt werden müssen, wenn sie voll sind.

Spooling heute

Für langsame Ausgabegeräte findet sich ein Spooling in fast allen heutigen Betriebssystemen. Hierbei steht jedoch weniger die Erhöhung des Durchsatzes im Vordergrund, sondern mehr die Entkopplung der Arbeit des Benutzers vom langsamen Ablauf der Ausgabe, beispielsweise wird ein Dokument gedruckt während der Benutzer an einer anderen Aufgabe arbeitet.

Erweiterung durch Mehrprogrammbetrieb

Der andere bereits erwähnte Nachteil der einfachen Stapelverarbeitungssysteme ist, dass neue Aufträge erst nach Bearbeitung des vorherigen Auftrags begonnen werden können. Dies ist besonders ärgerlich, wenn ein kurz laufender und wichtiger Auftrag ansteht, aber gerade ein lang laufender und unwichtiger Auftrag bearbeitet wird. Dieser Nachteil wird durch Mehrprogrammbetrieb behoben: Im Zeitmultiplex, also quasiparallel, werden mehrere Stapelaufträge bearbeitet. Dabei sind die Strategien zur Auftragsbearbeitung so gewählt und parametrisiert, dass für das Rechensystem ein weites Spektrum von Betriebszielen eingestellt werden kann.

In einfacheren Stapelverarbeitungssystemen mit Mehrprogrammbetrieb ist nur eine feste Anzahl von Strömen von Stapelaufträgen erlaubt. Diese Auftragsströme werden parallel bearbeitet. Innerhalb jedes Stromes wer-

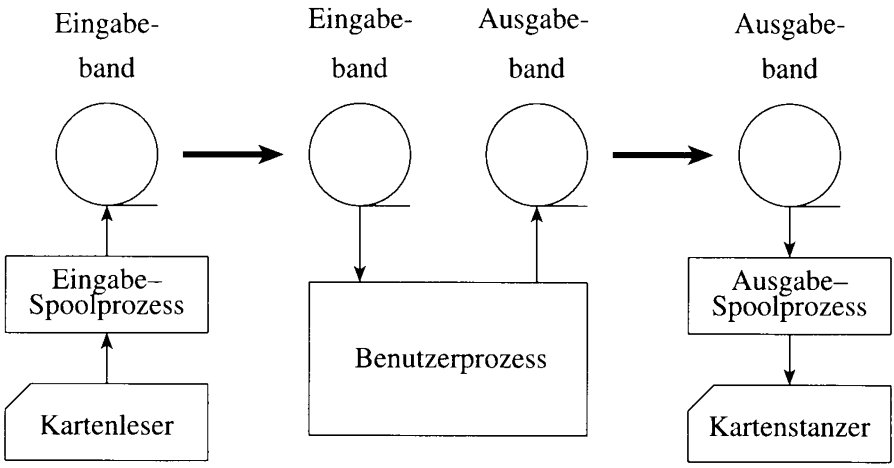

Abbildung 2.1: Spooling in den 60er Jahren

den die Stapel in der Reihenfolge ihres Auftretens bearbeitet. Häufig sind die Ressourcen des Systems fest auf die Auftragsströme aufgeteilt. Damit fällt das Problem der Konkurrenz um die Betriebsmittel weg. Die Bearbeitung der verschiedenen Auftragsströme kann mit unterschiedlicher Priorität erfolgen. Damit lässt sich die gewünschte bessere Reaktionszeit für kurze oder wichtige Aufträge erreichen. Bei allgemeineren Anforderungen an die Flexibilität der Auftragsbearbeitung können außer im Bereich Betriebsmittelzuteilungsstrategien kaum noch Vereinfachungen oder Abweichungen gegenüber einem Mehrbenutzerbetriebssystem, das auch den Dialog unterstützt, erwartet werden.

Erweiterung durch Stapelfernverarbeitung

Ein weiterer Schritt in der historischen Entwicklung war die Einführung der Stapelfernverarbeitung. Hier können die Eingabestapel über ein Netz auf die Eingabedatei zur späteren Bearbeitung überspielt werden. Entsprechend können die Ergebnisse über einen abgesetzten Drucker ausgegeben werden. Die hierzu notwendigen Erweiterungen im Betriebssystem betreffen die Kommunikationsprotokolle, den Netzbetrieb, die Bedienung abgesetzter Stationen außerhalb des Rechenzentrums und die Gruppierung von Geräten zu logischen Stationen. Mit Letzterem wird erreicht, dass ohne explizite Steuerung des Benutzers eine Ausgabe auf dem Drucker der Station erfolgt, an der auch das Eingabegerät angeschlossen ist.

2.5 Timesharing-Systeme

Historische Entwicklung

Timesharing-Systeme erlauben die interaktive Arbeit mehrerer Benutzer über direkt am Rechner angeschlossene Endgeräte. Sie entstanden Ende der 60er und Anfang der 70er Jahre. Wichtige Grundlagen hierfür wurden an der Universität von Manchester (Atlas) und am Massachusetts Institute of Technology (Projekt MAC) erarbeitet. Aufbauend darauf entstand auch eines der ersten wirklich eingesetzten Timesharing-Systeme, Multics. In Deutschland wurde bei AEG-Telefunken Ende der 60er Jahre ebenfalls ein sehr fortschrittliches, prozessorientiertes Timesharing-Betriebssystem für das Rechensystem TR 440 entwickelt. Dieses war gleichzeitig ein Mehrprozessorbetriebssystem. Es war bis Anfang der 80er Jahre vorwiegend an deutschen Universitäten im Einsatz. Heutige Mehrprozessbetriebssysteme können als Weiterentwicklungen der Timesharing-Systeme betrachtet werden, da grundlegende Realisierungskonzepte übernommen wurden, beispielsweise das Prozesskonzept, die Rechnerkernvergabestrategien, der virtuelle Arbeitsspeicher oder Zugriffsschutzmechanismen. Eine kurze historische Darstellung findet sich beispielsweise in (ROSE69, DENN71).

Definition

Ein Timesharing-System gehört zur Klasse der Mehrprozess- und Mehrbenutzersysteme. Es erlaubt die heute ganz selbstverständliche, interaktive Arbeitsweise. Im Timesharing-Betrieb bedient der Rechner die Benutzer quasigleichzeitig. Sofern die Betriebsmittel nicht für alle Benutzer ausreichen, müssen die Betriebsmittel den Prozessen kurzzeitig zugeteilt und dann wieder entzogen werden, so dass jeder Benutzer laufend einen akzeptablen Arbeitsfortschritt erfährt. Wir betrachten unter diesem Gesichtspunkt nun einige Hardware-Komponenten.

Rechnerkern- und Arbeitsspeicherverwaltung

Der Rechnerkern und der Arbeitsspeicher müssen in einem Timesharing-System entziehbar sein. Es ist also eine Virtualisierung erforderlich. Die Techniken hierfür werden in nachfolgenden Kapiteln besprochen.

Festplattenspeicherverwaltung

Der Hintergrundspeicher (Festplattenspeicher) mit den Dateien der Benutzer ist ein nichtentziehbares Betriebsmittel, da eine dynamische Auslagerung von Datenbeständen auf Magnetbänder und eine spätere Wiedereinlagerung sowohl unter dem Gesichtspunkt der technischen Machbarkeit als auch unter dem der Leistung nicht akzeptabel ist. Man hat daher bei der Zuteilung von Speicherbereichen auf dem Hintergrundspeicher grundsätzlich auf Verklemmungen zu achten. Bei sehr großem Hintergrundspeicher ist jedoch ein Speicherengpass möglicherweise so unwahrscheinlich, dass das Betriebssystem davon ausgehen kann, dass immer ausreichend Speicher vorhanden ist. Wenn dies einmal nicht der Fall sein sollte, wird der Prozess mit Fehler abgebrochen. Als Alternative dazu findet man auch folgende Regelung zur Vermeidung von Verklemmungen: Für jeden Benutzerauftrag ist

der maximale Festplattenspeicherbedarf bekannt. Dann wird die Anzahl der gleichzeitig arbeitenden Benutzer so begrenzt, dass ihre Maximalforderungen stets erfüllt werden können.

Interaktionen und Betriebsmittelzuteilung

Ein wichtiges Problem der Timesharing-Systeme ist die Wahl geeigneter Zuteilungsstrategien für den Arbeitsspeicher und den Rechnerkern. Ein Rechenzentrum erwartet, dass die Komponenten eines Servers gut ausgelastet sind. Ein Benutzer erwartet, dass sehr kurze Interaktionen auch in Bruchteilen von Sekunden erledigt werden. Für die Zuteilungsstrategien, insbesondere auch für die Rechnerkernzuteilungsstrategie, entsteht damit die Forderung, dass die Ausführung einer neu eingetroffenen, kurzen Interaktion in die Ausführungen bereits laufender, langer Interaktionen eingeschoben werden muss. Da dem Betriebssystem jedoch die Dauer einer Interaktion nicht im Voraus bekannt ist, muss es bei jeder Interaktion damit rechnen, dass es sich um eine kurze Interaktion handeln könnte. Es muss also „auf Verdacht" mit der Bearbeitung beginnen. Eine geeignete und besonders einfache Rechnerkernzuteilungsstrategie ist das Zeitscheibenverfahren. Hier sind die Aufträge in eine Warteschlange eingereiht. Der erste Auftrag erhält eine Zeitscheibe. Ist der Auftrag in dieser Zeitscheibe nicht beendet, dann wird er an das Ende der Warteschlange gestellt und der nächste wird bedient. Dabei müssen Benutzerprozesse in den Arbeitsspeicher gebracht und wieder daraus verdrängt werden. Damit ist eine virtuelle Adressierung unverzichtbare Voraussetzung. Als Basis für die Verdrängung von Seiten wird das Working-Set-Modell verwendet. Details zu diesen Strategien und weitere Strategien finden sich in Kapitel 15.

Weitere Anforderungen

Neben den bisher besprochenen Eigenschaften der Timesharing-Systeme, sind vier weitere von zentraler Bedeutung:

- Es gibt eine große und wechselnde Anzahl von Benutzern mit einem breiten Aufgabenspektrum. Das System muss sich optimal aus der Sicht der Gesamtheit aller Benutzer und aus der Sicht jedes einzelnen Benutzers an die aktuelle Situation anpassen.
- Die Benutzer halten langfristig gespeicherte Datenbestände. Diese sind gegen Verlust und gegen unberechtigten Zugriff zu schützen.
- Die Benutzer arbeiten direkt am Rechensystem. Damit muss das Rechensystem sicher und zuverlässig sein. Eine Fehlersituation muss immer so abgefangen werden, dass ein praktisch sofortiges Weiterarbeiten mit einem definierten Ausgangszustand möglich ist. Es dürfen möglichst keine bereits ausgeführten Eingaben oder Arbeiten verloren gehen. Fehler eines Benutzers dürfen die Arbeit anderer Benutzer nicht beeinflussen.
- Es ist ein guter Zugriffsschutz und eine Zugangskontrolle zum System notwendig.

2.6 Transaktionssysteme

Definition

Ein (zentrales) Transaktionssystem, früher auch Teilhabersystem genannt, erlaubt im Gegensatz zu einem Timesharing-System keine freie Programmierung. Alle angeschlossenen Benutzer führen nur fest definierte und in der Regel kurze Interaktionen (Transaktionen) aus, beispielsweise Flugreservierungen. Jede Transaktion ist in sich abgeschlossen. Alle Transaktionen greifen auf denselben globalen Datenbestand zu. Das Transaktionssystem enthält also in der Regel ein Anwendungsprogramm, an das alle Terminals angeschlossen sind. Die Transaktionen der einzelnen Terminals werden dann von dem Anwenderprogramm in einer bestimmten Reihenfolge nacheinander ausgeführt. Die Anzahl der angeschlossenen Terminals kann allerdings sehr groß sein und in die Tausende gehen.

Struktur des Betriebssystems

Da die Bedienung der Terminals durch das Anwenderprogramm erfolgt, betreibt dieses aus der Sicht des Betriebssystems lediglich viele Geräte. Man kann sich vorstellen, dass das Anwenderprogramm beim Start des Systems geladen wird und dann dauernd im Arbeitsspeicher bleibt. Es hat zudem keine kritischen Betriebsmittelforderungen. Damit kann ein Betriebssystem für Transaktionssysteme eine an die Aufgabe angepasste, sehr einfache Struktur haben. Es kann auf die Übertragung von Daten von und zu vielen Terminals optimiert werden. Bei vielen Anwendungen wird jedoch kein spezielles Betriebssystem eingesetzt, sondern ein übliches, kommerziell verfügbares Betriebssystem, dem ein Transaktionsmonitor zugefügt wird.

2.7 Mehrprozessorsysteme

Parallelarbeit auf Prozessebene

Mehrprozessorsysteme haben zwei bis etwa acht Rechnerkerne, die auf einen gemeinsamen Arbeitsspeicher zugreifen. Zu den Mehrprozessorsystemen gehören auch die Systeme, bei denen die Rechnerkerne auf einem Chip sind und gemeinsame Ressourcen verwenden (Multicore-Architekturen). Die Systeme sind eng gekoppelt, da im Gegensatz zu lose gekoppelten Systemen keine wesentliche räumliche Trennung zwischen den Prozessoren vorliegt. Die Ebene der Parallelarbeit ist die parallele Ausführung mehrerer Prozesse, wobei diese nicht wie im Einprozessorsystem nur quasiparallel, sondern echt parallel ausgeführt werden. Da bei der Programmierung von Prozessen schon immer ein echt paralleler Ablauf vorausgesetzt werden musste (der Prozesswechsel kann an beliebiger Stelle erfolgen!), können die Konzepte eines Mehrprozessbetriebssystems für einen Prozessor leicht auf mehrere Prozessoren erweitert werden.

Leistungsbegrenzung

Durch ein Mehrprozessorsystem wird zunächst der Durchsatz der Anlage, aber nicht die Bearbeitung eines einzelnen Auftrags beschleunigt. Ein Ziel

ist aber auch, die einzelnen Programme zu beschleunigen. Dazu müssen die Anwendungsprogramme umgeschrieben werden. Sie müssen in mehrere Schwer- oder Leichtgewichtsprozesse zerlegt werden, die weitgehend konfliktfrei parallel bearbeitbar sind. Der Leistungszuwachs durch Zufügen eines neuen Prozessors in ein eng (speicher-) gekoppeltes System nimmt jedoch relativ schnell ab. Dies hat mehrere Ursachen:

- Bei Erhöhung der Parallelarbeit treten mehr EA-Wünsche je Zeiteinheit auf. Die EA-Leistung des Systems wird zum Engpass und der Durchsatz wird dadurch begrenzt.
- Die Rechnerkerne greifen auf den gemeinsamen Speicher zu. Die Geschwindigkeit der Arbeitsspeicher ist gegenüber der Prozessorgeschwindigkeit so gering, dass es trotz Cache beim Speicherzugriff zu Wartezuständen der Rechnerkerne aufeinander kommen kann. Diese Zugriffskonflikte vermindern die Leistung des Gesamtsystems.
- Die Leichtgewichtsprozesse in einer Anwendung greifen auf gemeinsame Daten zu. Je mehr Leichtgewichtsprozesse realisiert werden desto grösser wird die Wahrscheinlichkeit von Zugriffskonflikten, d.h. von Wartezuständen auf die Freigabe von Betriebsmitteln.
- Nimmt die Zahl der Prozessoren zu, ist auch eine Erhöhung der Zahl der Prozesswechsel zu erwarten. Dies hat eine Verlangsamung der Abläufe zur Folge.
- Beim Aufruf von Systemdiensten kommt es mit steigender Parallelarbeit vermehrt zu Wartezuständen. Wird der ganze Betriebssystemkern als exklusives Betriebsmittel realisiert, kann immer nur ein Prozess den Systemkern betreten. Dies ist besonders kritisch, selbst wenn alle Dienste mit Wartezuständen in Systemprozesse ausgelagert werden. Lässt man Parallelarbeit im Betriebssystem zu, dann wird das Betriebssystem sehr komplex und damit fehleranfällig. Trotzdem treten weiterhin Behinderungen auf, da es viele Systemlisten und andere Betriebsmittel gibt, die nur exklusiv verwendet werden dürfen. Diese müssen also für die Dauer der Nutzung für andere gesperrt werden.

Das wichtigste Ziel beim Entwurf eines Mehrprozessorbetriebssystem muss es also sein, eine möglichst hohe Parallelität bei möglichst geringer gegenseitiger Behinderung der Prozesse und der Rechnerkerne zu erreichen.

2.8 Vielprozessorsysteme

Vielprozessorsysteme sind meistens Hochleistungssysteme. Sie haben hunderte oder tausende von Prozessoren. Hier sind spezielle Verknüpfungen

der Prozessoren vorhanden. Oft gibt es keinen gemeinsamen Speicher. Die entscheidende Parallelität liegt nicht auf Prozessebene, sondern auf Maschinenebene, insbesondere auf Befehlsebene. Dadurch können viele Prozessoren zur Bearbeitung eines Auftrags eingesetzt werden. Die Betriebssysteme für solche Vielprozessorsysteme sind stark konfigurations- und herstellerabhängig. Sie werden daher in diesem Buch nicht behandelt.

2.9 Echtzeitsysteme und eingebettete Systeme

Einsatzgebiet

Echtzeitsysteme werden für die Überwachung und Steuerung technischer Prozesse und Geräte sowie für die Erfassung von Messdaten eingesetzt. Der Einsatz reicht von großen Industrieanlagen, wie Kraftwerken, über Flugzeuge bis zu kleinen Geräten im Heimbereich. Ein Spezialfall der Echtzeitsysteme sind die eingebetteten Systeme. Diese werden zur Steuerung von Geräten eingesetzt, beispielsweise bei Telefon, Automobil, Fernsehgerät, Waschmaschine, Kaffeeautomat, Werkzeugmaschine oder Roboter. Bei allen Einsatzbereichen steht die Forderung nach der garantierten und nachweisbaren Einhaltung von Zeitbedingungen im Vordergrund. Typisch ist, dass innerhalb einer maximalen Reaktionszeit auf ein Signal des technischen Prozesses oder der Umgebung reagiert werden muss, dass Messdaten in festen Zeitabständen abgefragt oder dass bestimmte Tätigkeiten zu bestimmten Zeitpunkten angestoßen werden müssen. Als allgemeine Literatur zu den Echtzeitsystemen sei (BOLC91, FARB94, ZOBE95, RZEH96, LIU00, CHEN02, LI03, LAPL04, WORN05) genannt.

spezielle Hardware

Die Hardware von Echtzeitsystemen (Prozessrechner) unterscheidet sich von der Hardware der Universalrechner durch das Vorhandensein von:

- Realzeituhren,
- Analog-Ein/Ausgängen,
- Digital-Ein/Ausgängen,
- einer Vielfalt angeschlossener Geräte, Messwertgeber, Messwertempfänger und Sensoren,
- einer großen Anzahl von Unterbrechungsebenen. Die große Zahl von Unterbrechungsebenen ist notwendig, da dadurch gewährleistet wird, dass auf wichtige oder zeitkritische Ereignisse auch während der Behandlung unwichtigerer Ereignisse reagiert werden kann.

Weitere Einschränkungen sind bei eingebetteten Systemen zu finden. Hier spielt der Preis des Prozessrechners eine entscheidende Rolle. Deshalb ist die

Registerlänge und die Größe des Arbeitsspeichers oft sehr (zu) gering. Plattenspeicher und Laufwerke für rotierende Wechseldatenträger fehlen meist; sie wären den harten Umweltbedingungen beim Einsatz der eingebetteten Systeme nicht gewachsen oder wären zu teuer oder zu schwer oder erforderten zu hohe Energie. Das Betriebssystem und die Anwendungsprogramme können vom Benutzer üblicherweise nicht in das Gerät geladen werden, deshalb sind diese zusammen mit den permanenten Daten in einem ROM (read only memory) gespeichert.

Einhaltung von Zeitbedingungen

In den Echtzeitsystemen wird die garantierte Einhaltung der Zeitbedingungen durch ein Bündel von Maßnahmen unterstützt:

- Es gibt eine ausgefeilte Zeitverwaltung. Ereignisse können zu bestimmten Zeitpunkten oder periodisch ausgelöst werden. Diese können zur Aktivierung oder Deaktivierung von Prozessen verwendet werden.
- Alle Abläufe und Wartezustände in dem Betriebssystem und den Anwendungsprogrammen werden zeitlich überwacht. Wird hierbei eine vorgegebene Zeitgrenze überschritten, dann wird der Vorgang unterbrochen und ggf. eine spezifische Fehlerbehandlung aufgerufen. Damit gibt es keine unbegrenzten Wartezustände oder Endlosschleifen. Das System ist also spätestens nach der Höchstwartezeit wieder ansprechbar.
- Es werden viele Unterbrechungsebenen vorgesehen. Um schnellste Reaktionszeiten zu erhalten, gibt es im Betriebssystem keine Standard-Unterbrechungsbehandlungen, die dann zu den Anwendungen verzweigen, sondern die Unterbrechungsbehandlungen sind individuell durch den Anwender programmiert. Im Gegensatz zu Universalrechnerbetriebssystemen ist man also gerade nicht an einer Klassenbildung und Virtualisierung der Geräte interessiert.
- Die Unterbrechungsbehandlungen und der Betriebssystemkern dürfen höchstens in unbedingt notwendigen Fällen ganz kurz eine globale Unterbrechungssperre setzen. Normalerweise müssen sie durch wichtigere Unterbrechungen unterbrechbar sein.
- Alle zeitkritischen Programme sind speicherresident und werden nicht verdrängt.
- Der Speicher wird aus Zeitgründen noch häufig direkt adressiert.
- Die Rechnerkernzuteilung erfolgt nach Prioritäten oder nach speziellen deterministischen Zuteilungsstrategien, die die verlangten Zeitbedingungen berücksichtigen. Beispiele und Untersuchungen dazu finden sich u.a. in (BLAZ94). Die Betriebsmittelzuteilungsstrategien der Timesharing-Systeme, die nur eine Aussage über die mittlere Reaktionszeit, aber keine Garantie für eine maximale Reaktionszeit geben, sind nicht brauchbar.

- Bei der Einbindung der Prozessrechner in Netze oder beim Anschluss von Geräten über Bus-Systeme werden Protokolle bevorzugt, die eine maximale Wartezeit bei einem Sendewunsch garantieren. Beispiele für solche Protokolle sind das Token-Ring-Protokoll oder die Feldbus-Protokolle (SCHW89, BEND92, ETSC05). Im Gegensatz dazu garantiert das weit verbreitete Ethernetprotokoll keine maximale Wartezeit.

keine freie Programmierung

Die Echtzeitbetriebssysteme enthalten im Übrigen viele Konzepte und Eigenschaften der Universalbetriebssysteme. Die Dienste sind jedoch eingeschränkter und einfacher gestaltet, damit die Rechenzeiten reduziert werden, so dass schnelle Reaktionszeiten schon mit geringem Hardware-Einsatz erreicht werden können. Letzteres ist insbesondere bei eingebetteten Systemen oder Systemen für die Luft- und Raumfahrt wichtig. Die Programme, die in einem Echtzeitsystem ablaufen, sind normalerweise schon zur Zeit des Entwurfs bekannt und ihre Eigenschaften gehen maßgeblich in die Untersuchungen des Zeitverhaltens des Systems ein.

Energieverbrauch

Eingebettete Systeme, insbesondere mobile Systeme, haben oft keine permanente oder überhaupt nie eine Verbindung zum Stromnetz. Im Betrieb kommt es deshalb darauf an, möglichst wenig Energie zu verbrauchen. Neben der Verwendung energiesparender Komponenten kommt dem Energiemanagement im Betrieb eine Schlüsselrolle zu. So werden beispielsweise zur Energieersparnis gerade nicht benötigte Komponenten abgeschaltet.

Zuverlässigkeit

Die Anwenderprogramme, das Betriebssystem und die Hardware müssen ihrem Einsatzgebiet entsprechende Anforderungen bezüglich Zuverlässigkeit und Fehlertoleranz erfüllen. Dies gilt in besonderem Maße, wenn bei einer Fehlfunktion Menschenleben in Gefahr geraten können. Dann sollte selbst bei katastrophalen Fehlern das System immer in einen sicheren Zustand übergehen. Siehe hierzu auch die Ausführungen von Lauber (LAUB81, GORK89, LAUB99, PHAM92).

Spezialbetriebssysteme

Für Echtzeitsysteme sind teilweise einfachste, herstellerspezifische Spezialbetriebssysteme möglich, da das weite Benutzerspektrum mit freier Programmierung nicht auftritt. Allerdings besteht auch bei den Echtzeitsystemen die überaus sinnvolle Tendenz, Standardsysteme einzusetzen, beispielsweise UNIX- oder Windows-Systeme. Dabei werden für die Anwendung nicht benötigte Komponenten entfernt und andere Komponenten, beispielsweise für das Energiemanagement, verfeinert bzw. zugefügt. Beispiele für Echtzeitbetriebssysteme sind LynxOS (LYNU01), QNX (QNX 01), VRTX (MENT01) oder Windows-CE (MICR01, MURR98).