Minimalkerne – ein Grundbaustein für multimediafähige Betriebssysteme

H.-A. Schindler
TU Ilmenau, Institut für Praktische Informatik und Medieninformatik
FG Verteilte Systeme und Betriebssysteme

Email: ali@prakinf.tu-ilmenau.de

Oktober 1998

Zusammenfassung

Die meisten neuen Betriebssysteme – insbesondere auch die zur Unterstützung kontinuierlicher Medien – verwenden sogenannte Minimalkerne. Diese ermöglichen wesentlich besser strukturierte und wesentlich flexiblere Betriebssystemkonstruktionen. Durch neuere Arbeiten konnte widerlegt werden, daß die hiermit verbundenen Leistungsverluste so drastisch sein müssen, wie die Minimalkerne der ersten Generation annehmen ließen. Bezüglich der in einem Minimalkern zu realisierenden Funktionalität besteht bisher jedoch kein Konsens. Jedoch gibt es bestimmte Eigenschaften, die ein für multimediafähige Betriebssysteme verwendbarer Minimalkern haben sollte.

1 Allgemeines

Entsprechend /Burkow94/ sind die meisten Betriebssysteme zur Unterstützung kontinuierlicher Medien auf der Grundlage von Mikrokernen aufgebaut. Das gilt prinzipiell für einen Großteil neuer Betriebssystementwicklungen überhaupt. Hier sind auch so bekannte Namen zu nennen wie z. B. Mach, Windows NT, Chorus, Amoeba u. a.

Der Grund für die Verwendung von Minimalkernen liegt prinzipiell darin, daß sie eine „saubere“ Strukturierung der auf ihrer Grundlage konstruierten Betriebssysteme ermöglichen und gegenüber den älteren Makrokern-Betriebssystemen eine wesentlich verbesserte Flexibilität aufweisen.

Eine gute Strukturierung hat bekanntermaßen eine verbesserte Wartbarkeit und Modifizierbarkeit zur Folge. Eine spezielle Form der Modifizierbarkeit ist auch die Portierbarkeit auf andere Hardware.

Die verbesserte Flexibilität entsteht dadurch, daß ein großer Teil der Betriebssystemfunktionalität nicht mehr im Betriebssystemkern realisiert ist, sondern in Form eigenständiger Prozesse, die normalerweise als Server bezeichnet werden.

Hierdurch hat man nicht nur die Möglichkeit, solche Prozesse schnell durch andere zu ersetzen, d. h. die Eigenschaften des Gesamt-Betriebssystems zu ändern, sondern auch mehrere Serverprozesse mit prinzipiell der gleichen Aufgabe gleichzeitig zur Verfügung zu haben, also dem Nutzer verschiedene Varianten des gleichen Dienstes anzubieten. Als Beispiel sei die gleichzeitige Realisierung verschiedenartiger Dateisysteme genannt.

Ein typisches, auch heute noch oft praktiziertes Beispiel ist auch die Emulation verschiedener traditioneller Betriebssysteme oberhalb des Mikrokerns, z. B. oft als eine typische Anwendung des Mach-Mikrokerns herausgestellt.

2 Aufgaben eines Betriebssystemkerns

Bevor weiter auf Minimalkerne eingegangen wird, soll kurz auf den Sinn und die Rolle des Betriebssystemkerns überhaupt eingegangen werden.

Der Betriebssystemkern ist ein Teil der Betriebssystem-Software, der unmittelbar auf der Rechnerhardware aufsetzt und immer eine bestimmte Menge – meist niederer – Dienstleistungen für die strukturmäßig darüber liegenden Prozesse erbringt. – Der Umfang der jeweils im Kern realisierten Dienstleistungen unterscheidet dabei die einzelnen Kerne, trennt Mikro- und Makrokerne und differenziert auch einzelne Mikrokerne.

Die Prozesse des Betriebssystems – egal, ob es sich um Nutzerprozesse oder Systemprozesse handelt – also z. B. solche, die ebenfalls noch Betriebssystemfunktionalität realisieren, können die vom Kern realisierten Dienste durch sogenannte Kernaufrufe (bei Makrokernsystemen meist nur Systemaufrufe genannt) in Anspruch nehmen. Diese werden typischerweise mit Hilfe des Interruptsystems implementiert, um eine saubere organisatorische Trennung zwischen Kern und Prozessen zu realisieren.

Hinzu kommt, daß die meisten Prozessoren aus Sicherheitsgründen wenigstens zwei Modi unterscheiden, die sich u. a. durch die Anzahl der benutzbaren Befehle und der zugänglichen Prozessorregister unterscheiden. Im sogenannten Systemmodus (Supervisor-Modus) sind alle Befehle und Register uneingeschränkt benutzbar, während der sogenannte Benutzermodus bestimmte Beschränkungen auferlegt. – Die Zuordnung ist nun so, daß der Betriebssystemkern auf jeden Fall im Systemmodus läuft, während die Nutzerprozesse grundsätzlich im Benutzermodus laufen, bestimmte Dinge grundsätzlich also nur mit Hilfe von Aufrufen letztlich des Systemkerns realisieren können.

Aus dem bisher Dargestellten dürfte auch klar geworden sein, daß der Betriebssystemkern der Teil der Rechnersoftware ist, der von **allen** anderen Softwarekomponenten (Prozessen) benötigt wird und von diesen auch gemeinsam benutzt wird /Liedtke95b/. Dies hat Einfluß auf die Auswahl dessen, was unbedingt in einem Minimalkern enthalten sein muß.

3 Leistungsaspekte

Betrachtet man die dynamischen Vorgänge bei der Abarbeitung von Nutzerprogrammen an einer Rechenanlage, die entsprechend dem gewählten Schedulingalgorithmus heute typischerweise verschachtelt abgearbeitet werden, so muß offensichtlich ein häufiges Umschalten zwischen einzelnen Prozessen und den Prozessormodi (bei der Inanspruchnahme von Dienstleistungen) vorgenommen werden. Hierzu wird eine bestimmte Menge Zeit verbraucht, die nicht für nützliche Arbeit (Abarbeitung von Nutzerprozessen) zur Verfügung steht und als Overhead bezeichnet wird.

Die Umschaltung von einem Prozeß auf einen anderen wird auch als Kontextwechsel bezeichnet, wobei unter Prozeß-Kontext alle Informationen verstanden werden, die im Betriebssystem über einen Prozeß vorliegen. Bei einem Kontextwechsel müssen Informationen über den jetzt auszuführenden Prozeß an bestimmte Stellen gebracht werden, während die Informationen über den zuvor ausgeführten Prozeß gerettet werden müssen. Dies wird heute typischerweise durch das Umladen von Registerinhalten realisiert.

Spaltet man nun die früher in einem Makrokern enthaltene Betriebssystemfunktionalität in einen Minimalkern und (zahlreiche) Serverprozesse auf, so ist zwangsläufig eine erhöhte Anzahl von Kontextwechseln zu verzeichnen. Damit ist die Verwendung von Minimalkernen prinzipiell durch einen erhöhten Overhead gekennzeichnet.

Dies wurde durch Leistungsmessungen von mit Mikrokernen aufgebauten Systemen bestätigt, insbesondere waren z. B. die mit Mach gewonnenen Messungen so ernüchternd, daß Mikrokerne prinzipiell in den Ruf gerieten, ein drastisch verschlechtertes Leistungsverhalten des Systems zur Folge zu haben und aus Leistungsgründen von ihrer Benutzung Abstand genommen wurde.

Neuere Untersuchungen, insbesondere /Liedtke93/, /Liedtke95b/ und /Liedtke96/, die von einer genauen Analyse der Kerne und des Hardware-Software-Gesamtsystems (Prozessor und Minimalkern) ausgehen, zeigen daß die Leistungsverluste längst nicht so groß sein müssen, wie sie die Kerne der inzwischen sogenannten ersten Minimalkerngeneration (Mach u. a.) verursachten. Hierbei wurden die von den Prozessorspezifika abhängigen theoretisch zu erwartenden Verluste bestimmt, und es wurde versucht, die Kernalgorithmen so zu gestalten, daß die theoretischen Werte möglichst auch erreicht wurden.

Als Ergebnis konnte demonstriert werden, daß die theoretisch ermittelten Werte im Prinzip auch praktisch erreichbar sind – allerdings nur, wenn man von dem Prinzip abweicht, auch Mikrokerne möglichst prinzipiell prozessorunabhängig zu gestalten. Die heute verwendeten Prozessoren sind recht unterschiedlich konstruiert, so daß es darauf ankommt, jeweils optimale Algorithmen für das Hardware-Software-Gesamtsystem zu finden. Dabei gibt es sogar deutliche Unterschiede zwischen Mitgliedern der gleichen Prozessorfamilie, z. B. den Intel-Prozessoren 80486 und 80586 (Pentium). Daß es prinzipiell möglich ist, jeweils geeignete Algorithmen zu finden, zeigt die Geschichte des von Liedtke entwickelten Mikrokerns L3 bzw. L4 (Weiterentwicklung). Dieser existiert inzwischen in jeweils modifizierter Form für eine Reihe verschiedener Prozessoren (vergl. /Liedtke95b/, /Haertig+97/, /Heiser+97/).

4 Welche Bestandteile hat ein Minimalkern?

Von großer Wichtigkeit ist die Beantwortung der Frage, welche Leistungen tatsächlich in einem Minimalkern realisiert sein sollten.
Bisher gibt es dazu keine übereinstimmende Meinung der Betriebssystemkonstrukteure und tatsächlich existieren unter der Bezeichnung Minimalkern bzw. Mikrokern Systeme von sehr unterschiedlichem Umfang. In Anspielung auf den Term Mikrokern charakterisieren inzwischen Begriffe, wie Nanokern, Picokern und Femtokern (vergl. z. B. /Preikschat97/ und /Borrmann96/) die Tendenz, die Kerne immer weiter verkleinern zu wollen.

Nach /Liedtke95b/ ist entscheidendes Kriterium für die in einem Minimalkern zu realisierenden Konzepte die Funktionalität, da eigentlich wünschenswerte Beweise für Minimalität, Notwendigkeit und Vollständigkeit nicht möglich sind. Keinesfalls sollten als Kriterium z. B. Leistungsbetrachtungen (Performance) herangezogen werden, wie dies bei einigen Systemen getan wurde.

In /Liedtke95b/ heißt es dazu präziser: „Ein Konzept wird innerhalb des Mikrokerns nur toleriert, wenn seine Verlagerung aus dem Kern heraus (unter der Voraussetzung alternativer Implementierungen) die Implementation der vom System verlangten Funktionalität verhindern würde.“

Als wesentlich wird hier angesehen, daß Subsysteme außerhalb des Kerns nicht von anderen solchen Subsystemen zerstört oder beeinträchtigt werden dürfen und daß zwei solche Subsysteme sichere Kommunikationskanäle zueinander aufbauen können, die von dritten nicht beeinträchtigt werden können (Sicherheitsaspekte). Hieraus werden Überlegungen für den Aufbau von Mikrokernen unter bestimmten weiteren Voraussetzungen abgeleitet.

Entsprechend /Borrmann96/ und in Übereinstimmung mit anderen Arbeiten beziehen sich die in einem Minimalkern angebotenen Leistungen darauf, Grundfunktionalität für:

- Prozeßverwaltung,
- Prozeßsynchronisation,
- Behandlung von Traps und Interrupts sowie
- Interprozeßkommunikation (IPC)

zu schaffen.

Nach /Liedtke95b/ sind jedoch für diese im Minimalkern ausgeführten prinzipiellen Leistungen durchaus verschiedene Realisierungen und Abstraktionen möglich. Angeführt ist z. B., daß ein Minimalkern u. a. eindeutige Identifizierer für irgend etwas liefern muß. Aber je nach Konzept können für „irgend etwas“ so relativ unterschiedliche Dinge wie Threads oder Tasks oder Kommunikationskanäle stehen.

5 Multimedia-Aspekte

Hier geht es um die Frage, welche Dinge an einem Mikrokern wichtig sind, wenn dieser zum Aufbau eines Betriebssystems verwendet werden soll, welches kontinuierliche Medien, wie Audio oder Video unterstützen soll.

Ein Multimediasystem ist bekanntermaßen als eine Art Echtzeitsystem anzusehen, wobei die Forderungen jedoch etwas anders geartet sind, als bei den sogenannten harten Echtzeitsystemen.

Neben zu erzielenden Mindestparametern – damit eine Verarbeitung von Audio- und insbesondere Videodaten überhaupt erst möglich wird – kommt es hierbei nicht allein auf die Geschwindigkeit an sich an, sondern darauf, daß bestimmte Abarbeitungsgarantien übernommen werden können und auf diese Weise eine bestimmte sogenannte Dienstgüte (QoS) realisiert werden kann.

Hierzu ist einmal die Verwendung modifizierter Algorithmen in einzelnen Betriebssystemkomponenten erforderlich (z. B. Scheduler), zum anderen müssen neue Komponenten eingebracht werden, die u. a. die benötigten Betriebsmittelanteile errechnen (z. B. QoS-Manager, vergl. /Hyden94/) und eine Überwachung der tatsächlich verwendeten Anteile durchführen (Motoring).

Wie im vorigen Abschnitt angeführt, sind in einem Minimalkern jedoch vor allen Dingen Grundmechanismen enthalten. Von z. B. der Betriebssystemaufgabe „Verschachtelte Ausführung unterschiedlicher Prozesse bzw. Threads entsprechend ihrer Spezifika“ finden sich dort nur Primitiven für Prozessorzuordnung und -entzug, während die eigentlichen strategischen Schedulingentscheidungen in einem Serverprozeß oberhalb des Mikrokerns realisiert werden.

Ebenso sollten QoS-Management und Verwaltung anderer Betriebsmittel in entsprechenden Serverprozessen oberhalb des Minimalkernes ausgeführt werden.

Dagegen ist die sogenannte Interprozeßkommunikation (IPC) eine Angelegenheit für die typischerweise Mikrokern-Primitiven zur Verfügung stehen (vergl. z. B. auch /Liedtke93/). Da bei Audio- und Videobearbeitung oft große Datenmengen unter zeitlichen Beschränkungen übergeben werden müssen, kommt es dabei auf besondere Effizienz der Grundalgorithmen an, insbesondere sollten für diese Fälle z. B. echte Kopieroperationen vermieden werden.

Welche Dinge bleiben nun Besonderheit eines speziell für die Multimediabearbeitung zugeschnittenen Minimalkerns, nachdem viele für ein solches Gesamt-Betriebssystem charakteristischen Dinge offensichtlich in Serverprozessen oberhalb des Kerns zu realisieren sind?

Das sind:

1. Verwendung effizienter Algorithmen:
   Hierbei geht es darum, den Overhead so gering wie möglich zu halten. Diese Aufgabe ist hier um so wichtiger, da die Verarbeitung von Multimediadaten eine gewaltige Belastung des Rechnersystems darstellt.

2. Deterministisches, zeitlich abschätzbares Verhalten des Minimalkerns:
   Diese Eigenschaft ist eigentlich noch wichtiger als die erstgenannte, da eine Übernahme von Bearbeitungsgarantien prinzipiell nur auf Grundlage eines zumindest in Grenzen abschätzbaren Verhaltens aller Bestandteile des (Betriebs-) Systems übernommen werden kann.
   Ein wichtiger Faktor ist in diesem Zusammenhang auch die Bearbeitung von Interrupts und Traps. Die vollständige Bearbeitung dieser sofort nach Vorliegen des Requests, wobei oft gegenseitige Unterbrechungen zugelassen sind, führt zu prinzipiell nicht vorhersagbaren Verzögerungen und kann somit Fristverletzungen zeitkritischer Vorgänge (und auch den Verlust von Interrupts) nach sich ziehen. Prinzipiell sind hier Registration und Bearbeitung voneinander zu trennen, letztere muß von den Betriebsmittel-Schedulern mit eingeplant werden.
   Eine besonders gute Lösung dieses Problems liegt offensichtlich beim Minimalkern PXROS (vergl. z. B. /Maurer96/) vor, der eigentlich insbesondere für Echtzeit-Steuerungssysteme entwickelt wurde.

Das bei multimediafähigen Betriebssystemen tatsächlich auf Minimalkerne als Grundbausteine gesetzt wird, zeigen alle dem Autor bekannten neueren Projekte mit dieser Zielrichtung.

So besitzen die in /Burkow94/ erwähnten Projekte SUMO und RT-Mach einen Mikrokern, da sie auf die Grundsysteme Chorus bzw. Mach aufbauen. Das Multimedia-Betriebssystemprojekt der TU Dresden DROPS (Dresden Real-Time Operating System), z. B. in /Haertig97/, wird auf den Minimalkern L4 von Liedtke /Liedtke95b/ aufgebaut, während das noch laufende große Gemeinschaftsprojekt Pegasus auf dem eigens entwickelten Betriebssystemkern Nemesis aufsetzt (z. B. in /Leslie+97/ und /Roscoe95/), der dort als sehr kleiner Kern charakterisiert wird.

Literatur

/Borrmann96/
Borrmann, Lothar: Kleine und kleinste Kerne: Betriebssysteme mit Mikro- und Nanokernen
(Z) it+ti 38/2 (1996), S. 18 – 25

/Burkow94/
Burkow, Tatjana M.: Operating System Support for Distributed Multimedia Applications –
A Survey of Current Research
University of Cambridge, Pegasus paper 94-8, June 1994

/Haertig+97/
Härtig, Herrmann/Liedtke, Jochen u. a.: The Performance of u-kernel-Based Systems
16th ACM Symposium on Operating System Principles
in: (Z) ACM OSR 31/5 (December 1995) S. 66 – 77

/Hyden94/
Hyden, Eoin Andrew: Operating System Support for Quality of Service
PH. D. Thesis, University of Cambridge, 1994

/Heiser+97/
Heiser, Gernot u. a.: Implementation and Performance of the Mungi Single-Address-Space
Operating System
Forschungsbericht UNSW-CSE-TR-9704 der University of New South Wales, Australia, June 1997

/Leslie+97/
Leslie, Ian M./McAuley, Derek/Hyden, Eoin Andrew/Roscoe, Timothy u. a.:
The Design and Implementation of an Operating System to Support Distributed Multimedia Applications
Forschungsbericht University of Cambridge, June 1997

/Liedtke93/
Liedtke, Jochen: Improving IPC by Kernel Design
14th ACM Symposium on Operating System Principles
in: (Z) ACM OSR 27/5 (December 1993) S. 175 – 188

/Liedtke95b/
Liedtke, Jochen: On Micro-Kernel Construction
15th ACM Symposium on Operating System Principles
in: (Z) ACM OSR 29/5 (December 1995) S. 237 – 250

/Liedtke96/
Liedtke, Jochen: Toward Real Microkernels
(Z) CACM 39/9 (September 1996) S. 70 – 77

/Maurer96/
Maurer, Dieter: PXROS – ein skalierbarer Mikrokern mit besonderen Echtzeiteigenschaften
(Z) it-ti 38/2 (1996) S. 26- 32

/Preikschat97/
Schroeder-Preikschat, Wolfgang: State of the Art Operating System Technology
Tutorial-Unterlagen, HICCS-30, Hawaii, Maui, 1997

/Roscoe95/
Roscoe, Timothy: The Structure of a Multi-Service Operating System
PH. D. Thesis, University of Cambridge, 1995

Hinweis:

Die Literaturangaben sind auf ein Minimum beschränkt. Umfangreichere Literaturangaben zur behandelten Thematik befinden sich beim Autor.