Operating System Kernels

von Patrick Bitterling

Themenübersicht

- **Eine Einleitung über Kernel**
  - Begriffserklärung, Architekturen
- Kernel Subsysteme
  - Prozess-Scheduling, Speichermanagement, ...
- Der „Networking Stack“
- Alternativen zum Kernel
  - Das kernellose Betriebssystem KLOS

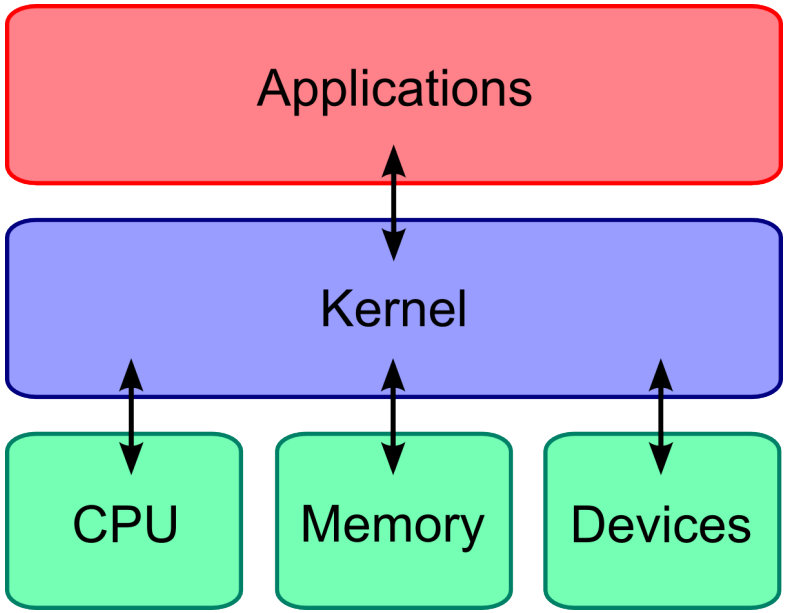
Begriffserklärung Kernel

-Kernel auch Betriebssystemkern

-„Kern der meisten Betriebssystem“

-Regelt wichtige Bereiche wie z.B. Speicher- und Prozessverwaltung, Gerätetreiber, System-Aufrufe, Dateisystem

-Dient als Schnittstelle zwischen Hard- und Software



Mikrokernel

-Erste Kernel-Architektur

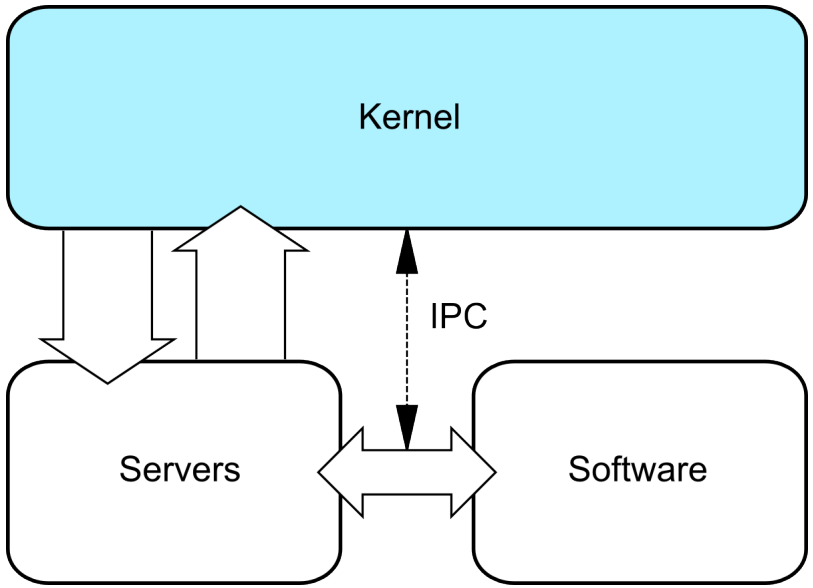
-Kernel mit wenigen Funktionen

-Erstellt Server um Software mehr Privilegien zu gewähren

-Bei Serverabsturz bleibt das System stabil

-Häufigere Kontextwechsel

-Negativ für Geschwindigkeit



Monolithischer Kernel

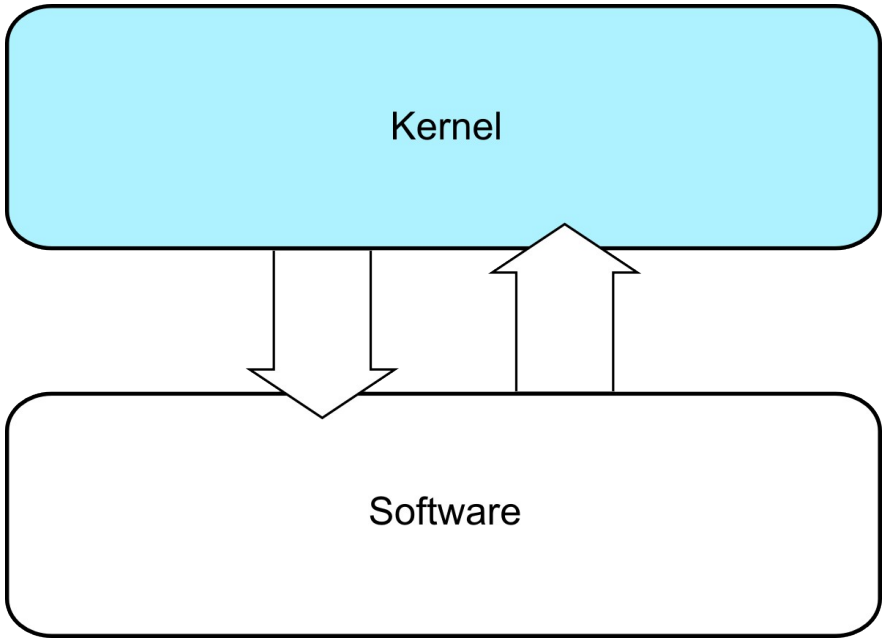
-Beinhaltet mehr Funktionen

-Ist größer und liegt komplett im Speicher

-Reduziert Kontextwechsel

-Gut für Geschwindigkeit

-Fehler haben größere Auswirkung (bis Systemabsturz)



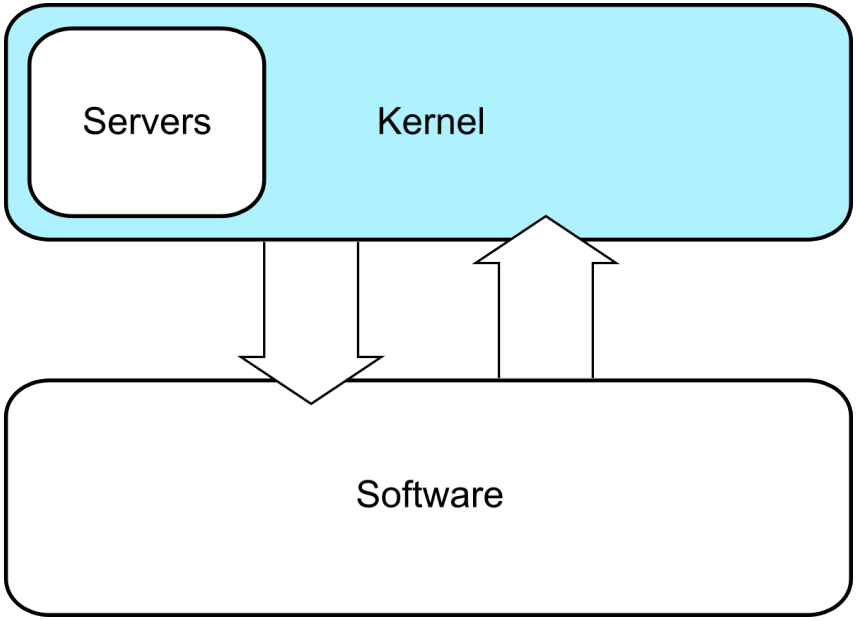
Hybridkernel

-Einiges im Kernelmodus, einiges im Benutzermodus

-Kompromiss zwischen Mikrokernel und monolithischen Kernel

-Möglichst wenige Kontextwechsel

-Bei Serverabsturz kein Systemabsturz



Themenübersicht

- Eine Einleitung über Kernel
  - Begriffserklärung, Architekturen
- **Kernel Subsysteme**
  - Prozess-Scheduling, Speichermanagement, ...
- Der „Networking Stack“
- Alternativen zum Kernel
  - Das kernellose Betriebssystem KLOS

Übersicht Kernel Subsysteme

- **Prozessmanagement/-scheduling**
- System Aufrufe
- Interrupts
- Kernelsynchronisation
- Speichermanagement

Multitasking

-Ein Prozess pro Prozessor(-kern)

-Simulation von vielen parallel laufenden Prozessen

-Heutzutage über preemptives Multitasking

-Prozess kann jederzeit unterbrochen werden

-Status des Prozesses wird gespeichert

-Kann später an dieser Stelle fortgesetzt werden

Prozessmanagement / -scheduling

	Linux	FreeBSD	Windows Vista
Prozess-Nomenklatur	Task	Task	Container (hat mind. 1 Thread)
Prozess – beinhaltet Informationen Adressräume, PID, ...	Ja	Ja	Ja
Scheduling	CFS	4BSD (neu ULE)	WTS
Benutzt Timeslices	Nein	Ja	Ja
Eltern-Kind Beziehung	Ja	Ja	Nein

Schedule mit Timeslice

-Jeder Prozess bekommt eine Priorität

- Höhere Priorität
  - Größeres Timeslice = mehr verfügbare Prozessorzeit
  - Darf Timeslice früher verbrauchen

-Unterbrochener Prozess darf später Timeslice verbrauchen

-Bei gleicher Priorität => Round Robin Verfahren

-Neue Prioritätsberechnung

Completely Fair Scheduler (CFS)

-Alle Prozesse gleich behandelt

-Wartende Prozesse bekommen Gutschrift (wait_runtime)

-Laufende Prozesse reduzieren ihr Gutschrift

-Sind in einem Rot-Schwarz-Baum sortiert

-Prozess mit höchster Gutschrift wird ausgeführt (ist ganz links im Baum)

Übersicht Kernel Subsysteme

-Prozessmanagement/-scheduling

-**System Aufrufe**

-Interrupts

-Kernelsynchronisation

-Speichermanagement

Systemaufrufe

-Ring-Schutzmechanismus

-Ermöglichung von Sicherheitsstufen

-Anwendungen sollen nicht direkt auf Hardware zugreifen können

-Prozesse abschotten

-Kommunikation zwischen den Ringen über Tore

-Privilegierter Modus (Kernel-Modus)

-Unprivilegierte Modi (Benutzer-Modi)

Systemaufrufe (2)

-Anwendung braucht Privilegien aus Kernelmodus

-Programmierschnittstelle (API) stellt Funktionen zur Verfügung

-API schreibt Nummer ins EAX-Register

-API sperrt EAX-Register und löst Software-Interrupt aus (Sprung in Kernelmodus)

-EAX-Register wird ausgelesen

-Entsprechende Aktion ausgeführt

-Springe wieder zurück

Übersicht Kernel Subsysteme

-Prozessmanagement/-scheduling

-System Aufrufe

-**Interrupts**

-Kernelsynchronisation

-Speichermanagement

Interrupts

-Kurzfristige Unterbrechung eines Programms
-Zeitkritische Arbeit ausführen
-Hardwareinterrupt asynchron
-Softwareinterrupt (Exception) synchron
-Wiederaufnahme vom vorherigen Programm

Interrupts (2)

ISR (Linux und Vista)

- Interrupt in 2 Teile (Hälften) aufgeteilt
- Die „obere Hälfte“ nimmt Interrupt entgegen
- Leitet Informationen zur „unteren Hälfte“
- Obere Hälfte beendet Hardware Interrupt
- Untere Hälfte führt die restliche Abarbeitung des Interrupt aus.

Interrupts (2)

ISR (Linux und Vista)

- Interrupt in 2 Teile (Hälften) aufgeteilt
- Die „obere Hälfte“ nimmt Interrupt entgegen
- Leitet Informationen zur „unteren Hälfte“
- Obere Hälfte beendet Hardware Interrupt
- Untere Hälfte führt die restliche Abarbeitung des Interrupt aus.

Interrupt Threads (FreeBSD)

- Sind Threads mit hoher Priorität
- Können von wichtigeren Threads unterbrochen (preemptiv) werden
- Sperren von Ressourcen im Kernel einfacher als bei ISR

Übersicht Kernel Subsysteme

-Prozessmanagement/-scheduling
-System Aufrufe
-Interrupts
-**Kernelsynchronisation**
-Speichermanagement

Kernelsynchronisation

-Prozesse wollen dieselben Ressourcen beanspruchen

-Kann zu schweren Fehlern führen (Race Condition)

Zeitpunkt	Transaktion 1	Gespeicherter Wert	Transaktion 2
1	Lese Wert W	W=1	-------
2	Vermindere Wert	W=1	Lese Wert W
3	Speichere Wert	W=0	Erhöhe Wert
4	--------	W=2	Speichere Wert

Kernelsynchronisation (2)

-Ressource muss geschützt werden

-Erster Prozess sperrt Ressource (Spinlock oder Semaphor)

Zeitpunkt	Transaktion 1	Gespeicherter Wert	Transaktion 2
1	Erlange Lock	W=1	-------
2	Lese Wert W	W=1	Versuch Lock
3	Vermindere Wert	W=1	Versuch Lock
4	Speichere Wert	W=0	Versuch Lock
5	Gib Lock frei	W=0	Erlange Lock
6	--------	W=0	Lese Wert W
7	--------	W=0	Erhöhe Wert
8	--------	W=1	Speichere Wert

Kernelsynchronisation (3)

-Spinlocks lässt andere Prozesse aktiv Warten

-Semaphor schickt andere Prozesse schlafen

-Problem bei preemptiven Kernels

-Lösung: atomare Operationen oder atomare Transaktionen

-Vista: Synchronisationsobjekt anstatt Spinlocks

-Objekte sagt wenn Ressource wieder verfügbar ist

Übersicht Kernel Subsysteme

-Prozessmanagement/-scheduling
-System Aufrufe
-Interrupts
-Kernelsynchronisation
-**Speichermanagement**

Speichermanagement

-Speichereinteilung in kleine Stückchen (Pages)

-Können verschoben (swap) werden zwischen RAM und virtuellen Speicher (Festplatte)

-Pages können von Prozessen geteilt werden

-Speichereinteilung zwischen Benutzer- und Kernelmodus

-Statisch (FreeBSD,Linux) oder dynamisch (Vista)

-Speicher voll -> swap lange nicht benutzte Pages

Themenübersicht

- Eine Einleitung über Kernel
  - Begriffserklärung, Architekturen
- Kernel Subsysteme
  - Prozess-Scheduling, Speichermanagement, ...
- **Der „Networking Stack“**
- Alternativen zum Kernel
  - Das kernellose Betriebssystem KLOS

Networking Stack

-Softwareimplementierung einer Sammlung von Protokollen

-Protokolle kommunizieren mit anderen Protokollen -> Schichtanordnung

-Modell für diese Schichten (TCP/IP und OSI)

-Unterste Ebene physikalische Übertragung

-Höhere Ebene fügt neue Funktionen hinzu

-Oberste Ebene bestimmt wie Programme diese Protokolle verwenden sollen

Themenübersicht

-Eine Einleitung über Kernel
  - Begriffserklärung, Architekturen
-Kernel Subsysteme
  - Prozess-Scheduling, Speichermanagement, ...
-Der „Networking Stack“
-Alternativen zum Kernel <=
  - Das kernellose Betriebssystem KLOS

Alternative zu einen kernelbasierenden OS

-Programmiere OS direkt für eine Hardwarekonfiguration

-Früher auf einen Computer angepasst

-Heute noch bei einigen Spielkonsolen und technischen Geräten wie z.B. Routern

-Entwurf eines OS mit wenigen Kontextwechsel (verbrauchen Rechenzeit)

-KLOS statt Kontextwechsel Segmentierung für ähnliche Sicherheit

Danke fürs Zuhören