Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

MIN Faculty
Department of Informatics
Scientific Computing

Proseminar

“Speicher- und Dateisysteme”

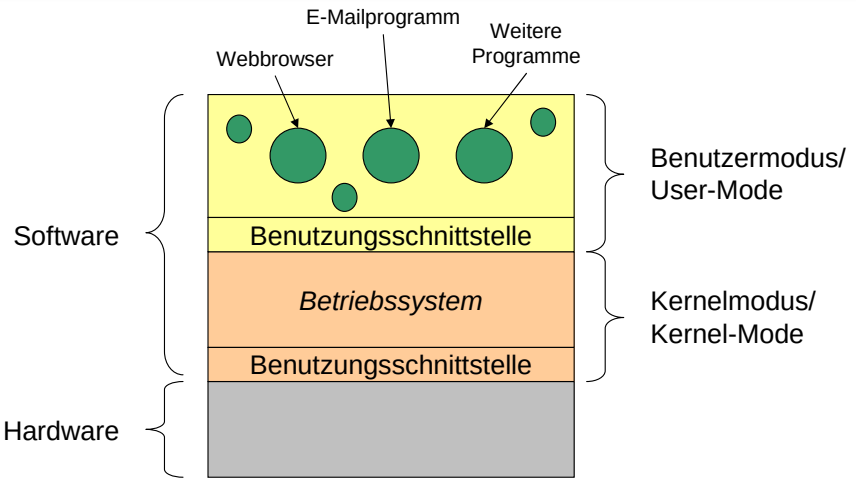
Betriebssystemschichten

(11.03.2011)

Bernd Ihnen

Übersicht

- Einleitung Betriebssysteme/ Übersicht Betriebssystemschichten
  - Mikrokernel
  - Monolithischer Kernel
  - Vergleich der Kernel
- Fallbeispiel Linux
  - Kernelaufbau
  - Dateisytemeinbindung
  - Caches
  - Schichtenarchitektur von NFS
- (Kurze) Übersicht Kernelarchitektur Minix
- Zusammenfassung

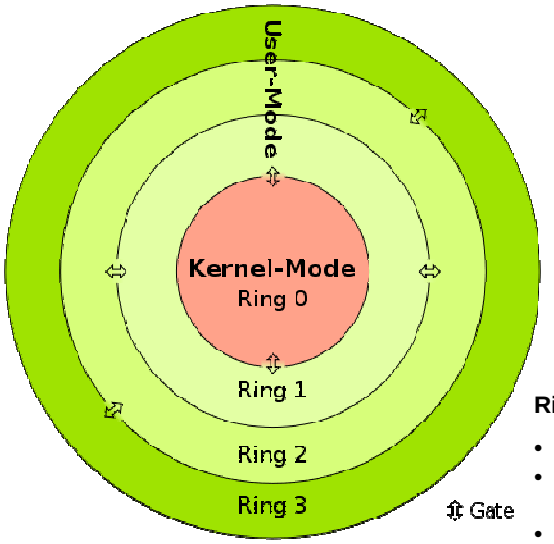


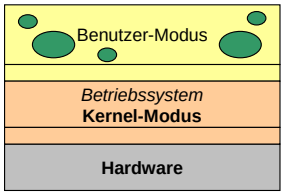
Gemäß **DIN** 44300 wird ein Betriebssystem wie folgt definiert:

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

Bild geändert: „Moderne Betriebssysteme", 3. Aufl. S. 30 (A.S. Tanenbaum)

- **Geschichte** (2. Weltkrieg, heute 4. PC-Generation)
- **Zweck**: Hardware nicht direkt ansprechen, Anwendungsprogrammierer, erweiterte Maschine und Ressourcenverwalter
- **Oberfläche** nicht BS, Shell oder GUI (Graphical User Interface)





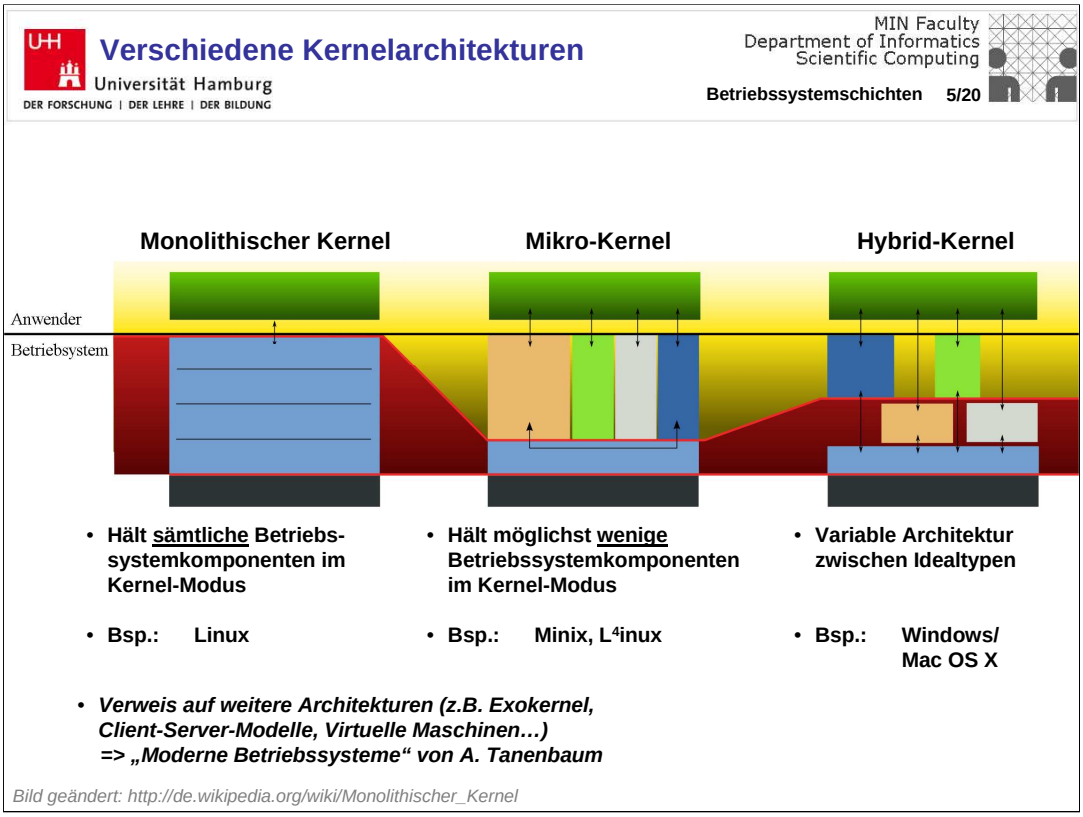
Ringaufbau eines x86-Prozessors:

- Ringe 1-3 im User-Mode aktiv
- Haben zunehmend weniger Zugriffsrechte auf die Hardware
- Kommunikation erfolgt z.B. durch Gates

Wichtige Konzeptionskomponenten: **Abstraktion und Modularisierung**

Bildquelle: http://de.wikipedia.org/wiki/Ring_(CPU)

- **Ziel**: Von der Hardware abstrahieren, Prozesse trennen
- **Linux/ WIN: Ca. 5 Mio Zeilen Code, 100 x 1.000 Seiten (1 Buch = Kernel)**
- **GUI/ Explorer 10x bis 20x größer**
- **Abstraktion/ Modularisierung** => Vereinfachung u. Ressourceneinteilung
- *Honeywell 6180: 1. System mit Beschränkungskonzept kannte acht Ringe*
- *Prozesse in bestimmten Ringen dürfen nicht Prozesse in anderen stören, haben ggfls. weniger Rechte*
- **Farbhinweis (orange = Kernel/ gelb = UserMode/ grau = Hardware)**



- Betriebssystemarchitektur: **Bindeglied** zw. Hardware u. Benutzungsoberfläche
- **Architektur** => jew. Betriebssystem
- Grafik zeigt **konzeptuelle Unterschiede**

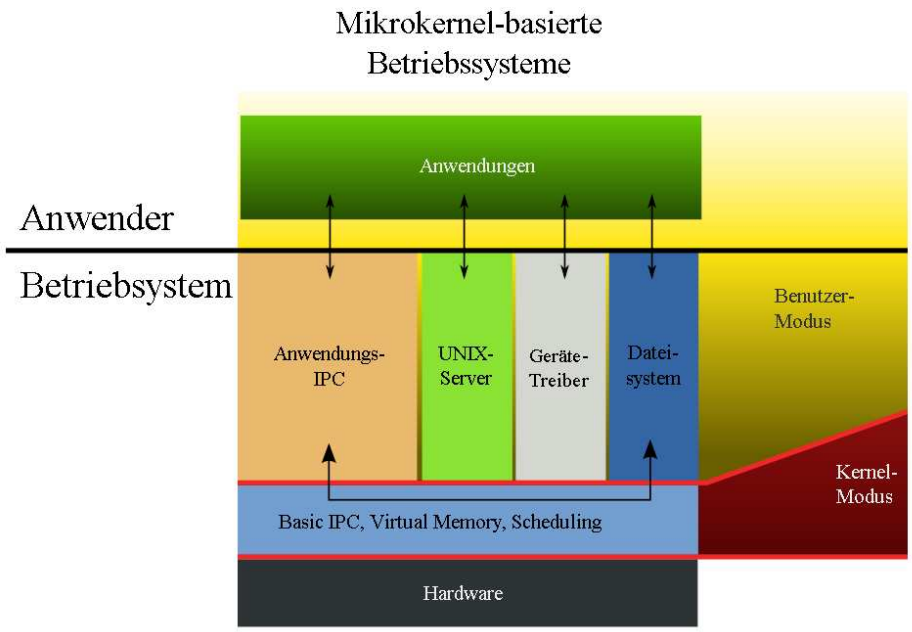


Bild geändert: http://de.wikipedia.org/wiki/Monolithischer_Kernel

- **Beispielvariante**
- Eigenschaft: Möglichst **wenige BS-Komponenten** im **Kernel**-Modus
- Ziel: **Hohe Ausfallsicherheit** u. **hohe Modularisierung seitens Architektur**
- IPC/ Interprozesskommunikation: **Kommunikation zw. Prozessen z.B. durch Pipes** , strikte **Trennung** des **Speicherbereiches**
- Virtual Memory: Vom tatsächlich vorhandenen Arbeitsspeicher unabhängig/ **physisch verteilt**, für Programm **zusammenhängend**,
- Vorgang **Abbildung** = **Paging**
- Scheduling/ Prozessplaner: Verwaltung **Prozesse**, **Reihenfolge**, **präemtiv** (**unterbrechend**) => gleichzeitig mehrere Prozesse mit **1 Kern**
- Treiber: Kommunikation zw. Software/ Hardware d. Schnittstellen
- Dateisystem: Eine d. **wichtigsten Aufgaben** BS, **unterschiedl. Funktionen**
- UNIX-Server: Erfüllt Funktion einer **standardisierten API** (Application Programming Interface), **Schnittstelle zum BS**, Modulares Subsystem einer UNIX-Umgebung nach POSIX-Standard (später)

Übersicht Monolithischer Kernel

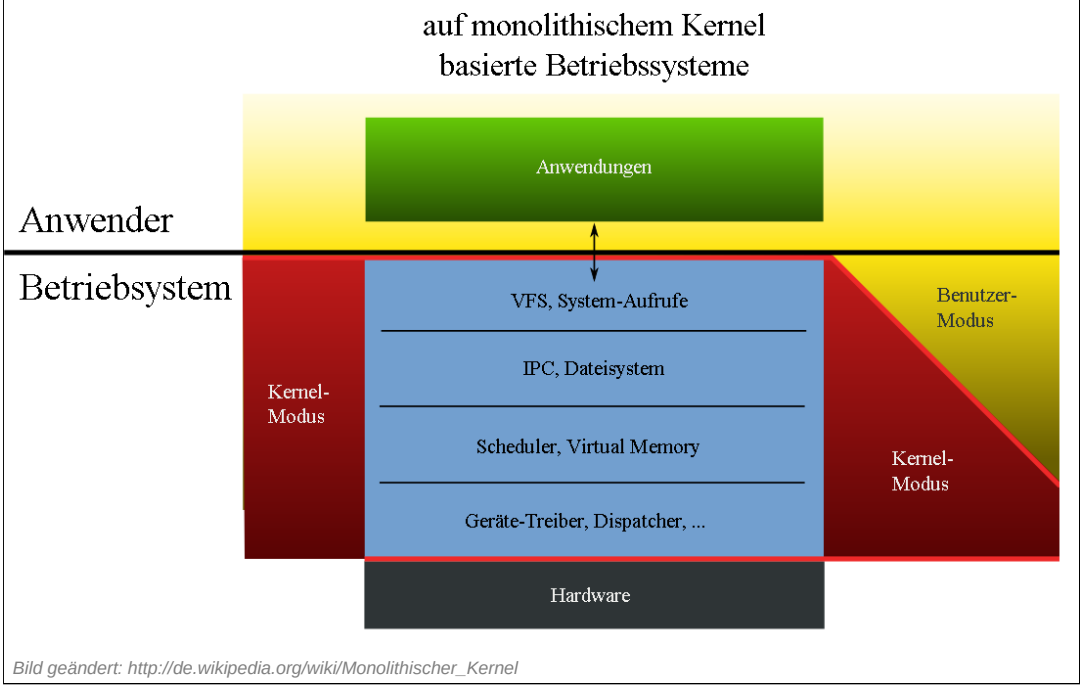


Bild geändert: http://de.wikipedia.org/wiki/Monolithischer_Kernel

- Eigenschaft: **Sämtliche BS-Komponenten** im **Kernel**
- Ziel: Nutzung **Geschwindigkeitsvorteil**
- **Weitere/ andere Komponenten**
- Dispatcher: **Zuteilung d. Prozesse an Scheduler**, Prozesse **entziehen/ übergeben** (Reihenfolge => Scheduler)
- **IPC** nur 1x
- VFS/ Virtuelles DateiSytem (file): **Einheitliche API f. unterschiedliche Dateisysteme**, **betriebssystemfremde Dateiformate** lesbar

Vergleich der Kernel

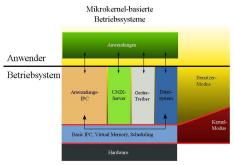
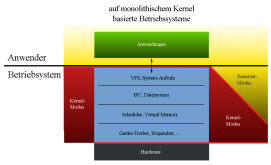
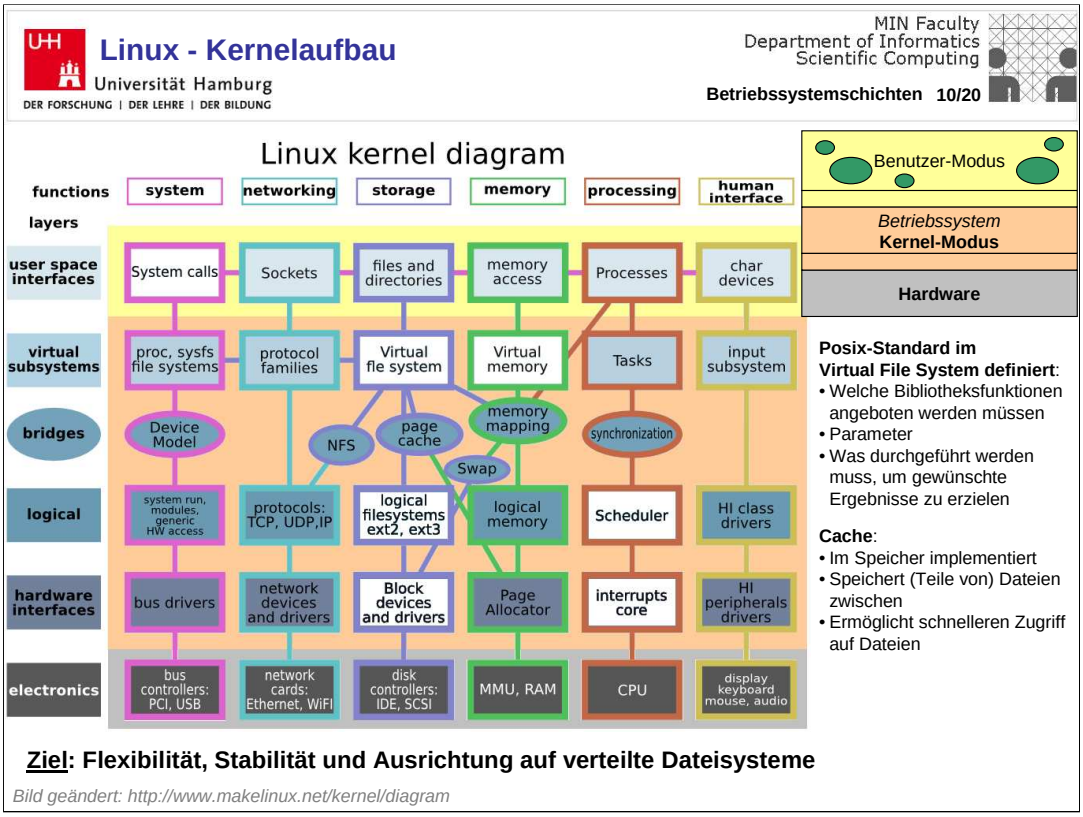
	Mikro-Kernel	Monolithischer Kernel
		
Vorteile:	• Hohe Ausfallsicherheit • Klares Schnittstellendesign • Kapselung der Komponenten	• Performancegewinn
Nachteile:	• Grundsätzlicher Geschwindigkeitsverlust durch Moduswechsel • Hoher Synchronisationsaufwand	• Ausfallrisiko bei Absturz einer Systemkomponente • Probleme beim Austausch von Systemkomponenten

Bild geändert: http://de.wikipedia.org/wiki/Monolithischer_Kernel

- **Mikrokernel** (MINIX, L4inux):
- Ausfall BS-Komponente kein **Absturz gesamtes System architekturbedingt**
- **Gerätetreiber** im User-Mode: Vorteil: **Kapselung, diverse Anbieter**, Nachteil: Z.T. **schwer programmierbar ohne volle Zugriffsrechte auf Hardware**
- **Synchronisationsaufwand Benutzerprozesse**, **erschwerte Koordination im Kernel-Modus, schwer optimierbar**
- **Monolithischer Kernel** (Linux):
- **Risiko** wenn **effektive Modularisierung verpasst**, **Absturz** System
- **Hybrid** (WIN, Mac):
- **Zwischenarchitektur, Komponentenansiedlung => Architektur BS**
- Ziel: **Vorteile** verbinden

Übersicht

- Einleitung Betriebssysteme/ Übersicht Betriebssystemschichten
  - Mikrokernel
  - Monolithischer Kernel
  - Vergleich der Kernel
- Fallbeispiel Linux
  - Kernelaufbau
  - Dateisytemeinbindung
  - Caches
  - Schichtenarchitektur von NFS
- (Kurze) Übersicht Kernelarchitektur Minix
- Zusammenfassung



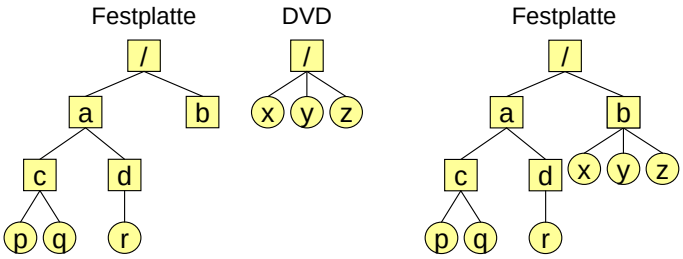
- *alte Grafik*
- ***Linus Benedict Torvald** Initiator LinuxKernel, Basis **UNIX**, Programmierung um PC besser zu verstehen*
- *Eigenschaft: **Monolith. Kernel**, **stark modularisiertes** BS gg. Nachteil Monolith. Kernel, schnelle **Performance**, Konfiguration **Minimales System**, Module können **dynamisch zugeschaltet** werden, **heute** auch Module im **User-Modus** (Bsp. **FUSE**), trotzdem Bsp. f. Monolith. Kernel*
- *Übersicht, Fokus,*
- *Block Devices/ Blockeinheiten**, Gerätetreiber: Speicherblockbasierte Hardware***
- *Dateisystem: Ursprünglich **ext2**, seit **2008 ext4***
- ***Virtuelles Dateisystem**: **Einheitliche API***

- *NFS/ Network File System: (Kurz) **Brücke** vom **lokalen** virtuellen Filesystem zu einem **im Netzwerk befindlichen** Dateisystem (später)*
- ***Swap**: Kann **ganze Prozesse** zwischen **Platte** und **Speicher verschieben**, wenn **physischer Speicher** f. aktive Prozesse **nicht mehr reicht**, auch möglich **nur Teile** der Datei zu **verschieben**, so dass nur noch **Seitentabllen**/ **Benutzerstruktur** eingelagert ist => gilt **trotzdem** als **im Speicher**, Text-, Daten-, Stacksegement wird bei Bedarf dynamisch geholt*

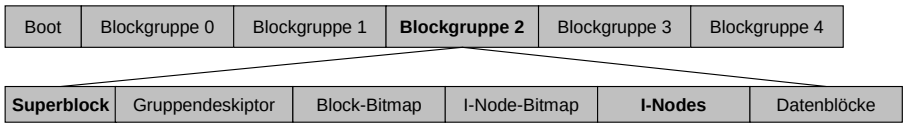
- ***Page Cache: Hält zuletzt genutzte Dateien, unterschiedliche Varianten***
- ***i.d.R. Algorithmus LRU (Least Recently Used)** bei dem der **am längsten nicht verwendete Block ausgelagert/** aus dem Cache gelöscht werden**.***
- *NFU (Not Frequently Used) = am wenigsten genutzt*
- *LRFU (Least Recently/ Frequently Used) = Kombination aus LRU/ NFU durch zusätzlichen Wert*
- ***Memory Map:***

Exkurs: Linux - Dateisystemeinbindung

Dateistruktur in Linux entstammt einer Wurzel, weitere Dateisysteme werden eingehängt



Ext2-Dateisystem von Linux



- ***Superblock* enthält Informationen über Aufbau des Dateisystems (auch Anzahl I-Nodes)**
- ***I-Nodes* beschreiben genau eine Datei, enthalten Verwaltungsinformationen einer Datei und Informationen über Blocklokation**

Bild geändert: „Moderne Betriebssysteme“, 3. Aufl. S. 901 und S. 909 (A.S. Tanenbaum)

- Dateisysteme **ähnliche Funktionalitäten**: **Verzeichnisse** absolut/ relativ erreichbar, unterschiedliche **Namenlänge**/ -**zeichen**
- **Linux**: Kein Zwang **Dateiendung**, eine Stammwurzel
- Dateisystemfunktionen in **Dateisystemstruktur** definiert
- **Superblock**: Kritische Informationen, beschädigt => Block unbrauchbar
- **I-Node**/ Indexknoten: Zur Verwaltung der Zuordnung zwischen Block und Datei => Attribute u. Plattenadressen, Linux: **Auch Verzeichnisse u. Geräte** durch Inode repräsentiert, Enthält **Zeiger** auf **inode_operationen** und Zeiger auf **Menge der auf diese Struktur anwendbaren Operationen**
- *File: Öffnet eine Datei, die einem Prozess zugeordnet ist, Zeiger auf Menge auf dieser Struktur anwendbaren Funktionen*
- *Dentry: Verzeichniseintrag, einzelne Pfadkomponente, vermeidet doppelte Abbildung bei Mehrfachnutzung*

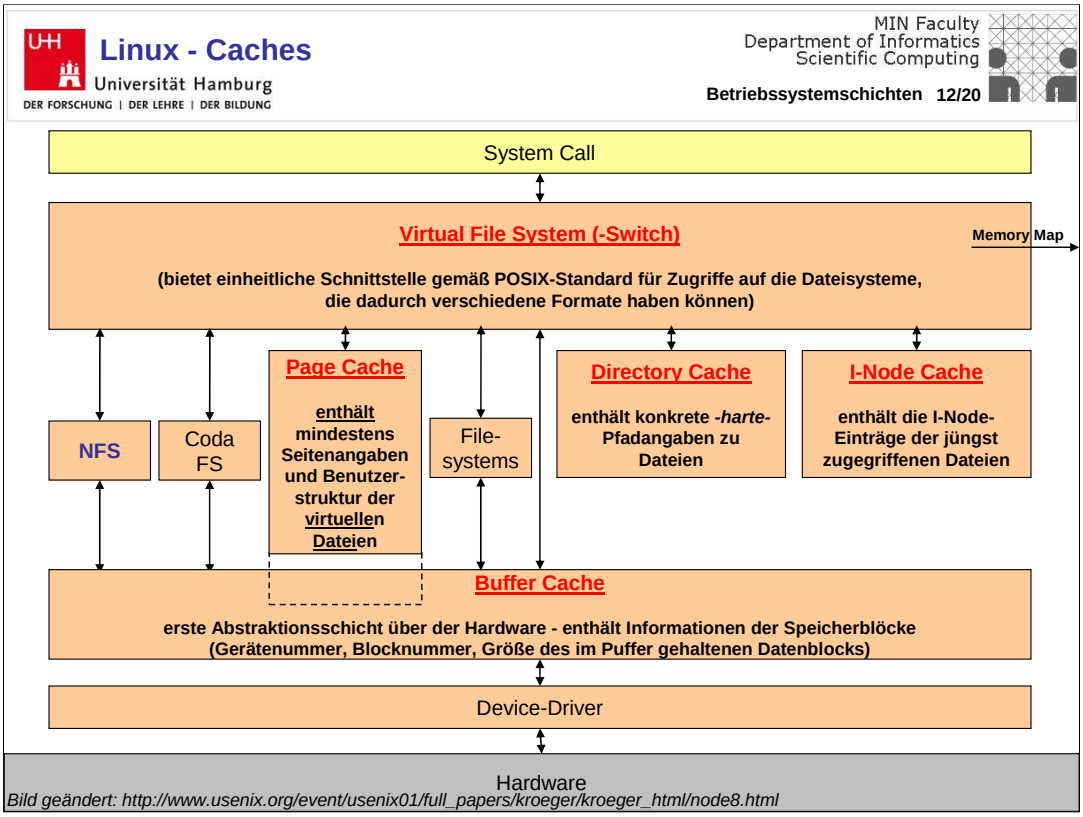


Bild geändert: http://www.usenix.org/event/usenix01/full_papers/kroeger/kroeger_html/node8.html

- VFS => **Unabhängigkeit vom Dateisystem** mit Hilfe **POSIX**-Standard, **fehlen Funktionen** => **Wrapper** eingesetzt mit fehlenden Funktionen

- Zusammenlegung Page Cache/ BufferCache seit **Kernel 2.4**
- Page Cache: Ziel **Performancesteigerung** bei **Mehrfachzugriff**, da nicht von **Platte sondern** aus dem **Speicher**, **Verwaltung d. Zugriffe durch Zeiger auf MemoryMap**
- DirectoryCache: **Pfadangaben** zu Dateien, **erneute Suche nicht linear** durch alle Dateien, sondern mittels **InodeVerweis, Vermeidung doppelter Abbildung bei Mehrfachnutzung**
- **InodeCache**: Inodes auf denen **dentry-Verweise** (einzelne Pfadkomponenten) liegen **bleiben**
- CodaFS: Netzwerkdateisystem

Linux – Schichtenarchitektur bei NFS

NFS (Network File System) fasst Dateisysteme auf verschiedenen Rechnern zusammen

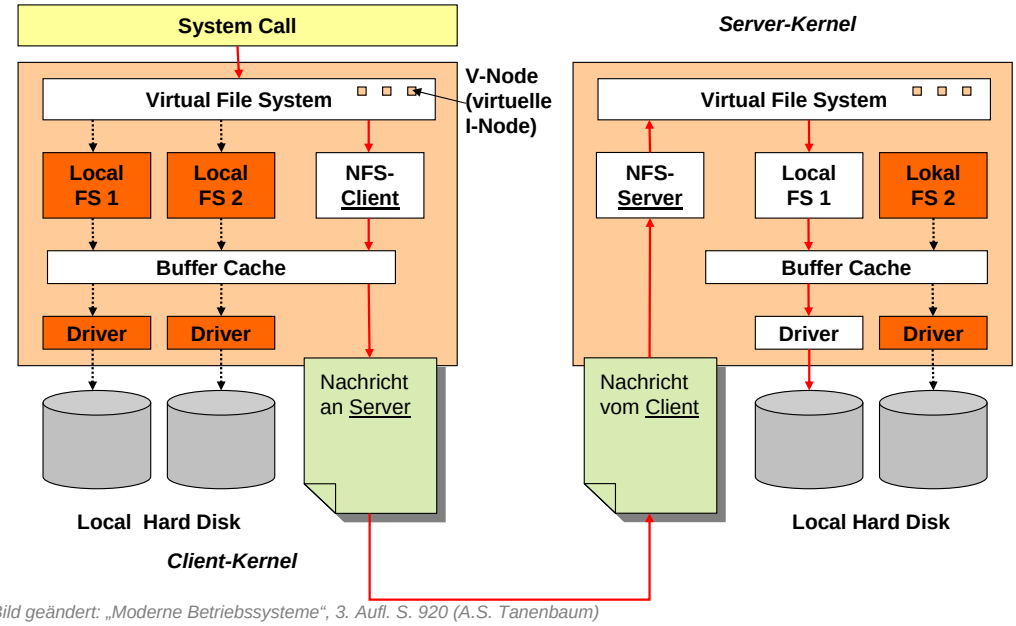


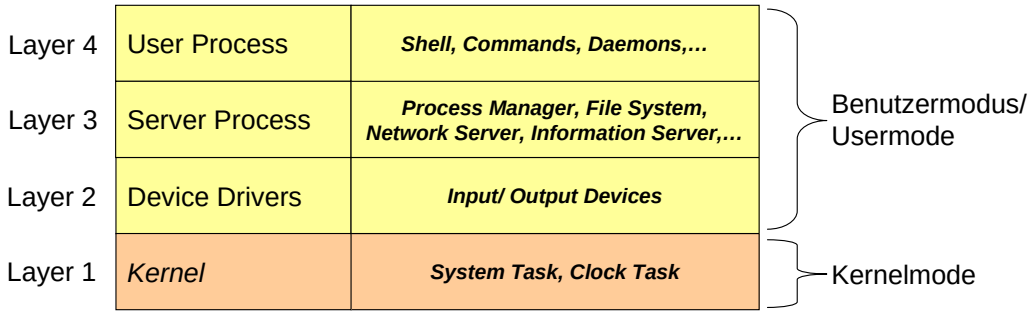
Bild geändert: „Moderne Betriebssysteme“, 3. Aufl. S. 920 (A.S. Tanenbaum)

- VFS enthält V-Node = virtuelle Inode, **Zeiger auf lokale Inodes oder entfernte Inodes (Remote Inodes/ R-Nodes) innerhalb NFS-Client**
- NFS: ermöglicht **Kommunikation** zu entfernten Dateisystemen
- **Automounting**: Erst bei Zugriff (evtl. redundante Dateien auf Servern)

- Einleitung Betriebssysteme/ Übersicht Betriebssystemschichten
  - Mikrokernel
  - Monolithischer Kernel
  - Vergleich der Kernel
- Fallbeispiel Linux
  - Kernelaufbau
  - Dateisytemeinbindung
  - Caches
  - Schichtenarchitektur von NFS
- (Kurze) Übersicht Kernelarchitektur Minix
- Zusammenfassung

Übersicht Kernelaufbau bei Minix

Mikrokernelaufbau bei MINIX



- **Entwickler: Andrew S. Tanenbaum**
- **Entstand durch Re-Implementierung eines UNIX-Systems mit dem Ziel ein einfaches Mikrokernel-System zu Lehrzwecken anzubieten.**
- **Kontrast zu Linux mit Monolithischem Kernel**

Bild geändert: http://imma.wordpress.com/2007/04/02/presentation-internal-structure-of-minix/

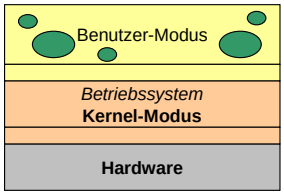
- Größtenteils beschriebene Funktionen
- Daemon: Programm im Hintergrund, das Dienste anbietet
- **Kontrastdarstellung**

- **Einleitung Betriebssysteme/ Übersicht Betriebssystemschichten**
  - **Mikrokernel**
  - **Monolithischer Kernel**
  - **Vergleich der Kernel**
- **Fallbeispiel Linux**
  - **Kernelaufbau**
  - **Dateisytemeinbindung**
  - **Caches**
  - **Schichtenarchitektur von NFS**
- **(Kurze) Übersicht Kernelarchitektur Minix**
- **Zusammenfassung**

Zusammenfassung

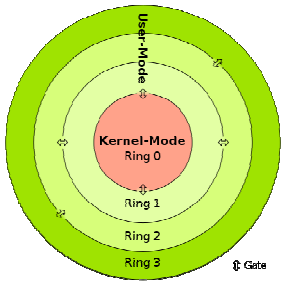
Betriebssystem

- **Hardware + Software zum Zugriff auf die Hardware**
- **Trennung in Kernel-Modus und Benutzer-Modus**
- **Geringere Zugriffsrechte auf Hardware im Benutzer-Modus**



Mikrokernelarchitektur (Minix, L4inux)

- **Für das Betriebssystem essenzielle Komponenten arbeiten im Kernel-Modus, alle anderen im Benutzer-Modus.**
- **Vorteil: Hohe Ausfallsicherheit, klares Schnittstellendesign**



Monolithischen Kernel (Linux)

- **Alle Komponenten arbeiten im Kernel-Modus**
- **Vorteil: Performance**

Hybrid-Kernel (Windows, Mac OS X)

- **Mischform**

- Was steckt unter Benutzeroberfläche von Betriebssystemen?
- **Modular oder in Schichten** aufgebaut
- Betriebssystem legt **Modus der Komponenten** fest
- **Ressourcenverwalter/ erweiterte Maschine**

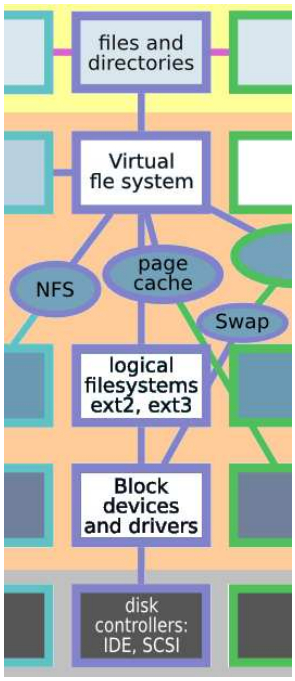
Zusammenfassung

Linux

- **arbeitet grundsätzlich mit Monolithischem Kernel (Aufweichung z.B. durch FUSE => User-Mode)**
- **Betriebssystemkomponenten modular aufgebaut**
- **Virtual File System ermöglicht Verwendung verschiedener Dateisysteme durch Einhaltung von POSIX-Standard**
- **Caches/ Buffer erlauben Zwischenspeicherung von Dateien (Page/ Buffer), deren Pfadangaben (Directory) oder deren Metadaten (z.B. I-Node, Gerätenummer,…)**
- **Einfache verteilte Dateisystemeinbindung durch NFS (Network File System)**

Minix

- **Stellt durch konsequentem Mikro-Kernel Kontrastarchitektur dar**



- **Performance, Modularisierung => Stabilität**
-

- **Abschluß: Welches System besser? Frage eher was ist besser geeignet?**
- **Derzeit Entwicklung zur Hybridarchitektur**

Bildquellen:

Header:
http://wr.informatik.uni-hamburg.de/start

Genutzte Bilder:
„Moderne Betriebssysteme“, 3. Aufl. S. 30 (A.S. Tanenbaum)
http://de.wikipedia.org/wiki/Ring_(CPU)
http://de.wikipedia.org/wiki/Monolithischer_Kernel
http://www.makelinux.net/kernel/diagram
„Moderne Betriebssysteme“, 3. Aufl. S. 901 und S. 909 (A.S. Tanenbaum)
http://www.usenix.org/event/usenix01/full_papers/kroeger/kroeger_html/node8.html
„Moderne Betriebssysteme“, 3. Aufl. S. 920 (A.S. Tanenbaum)
http://imma.wordpress.com/2007/04/02/presentation-internal-structure-of-minix

Vielen Dank für Eure Aufmerksamkeit!