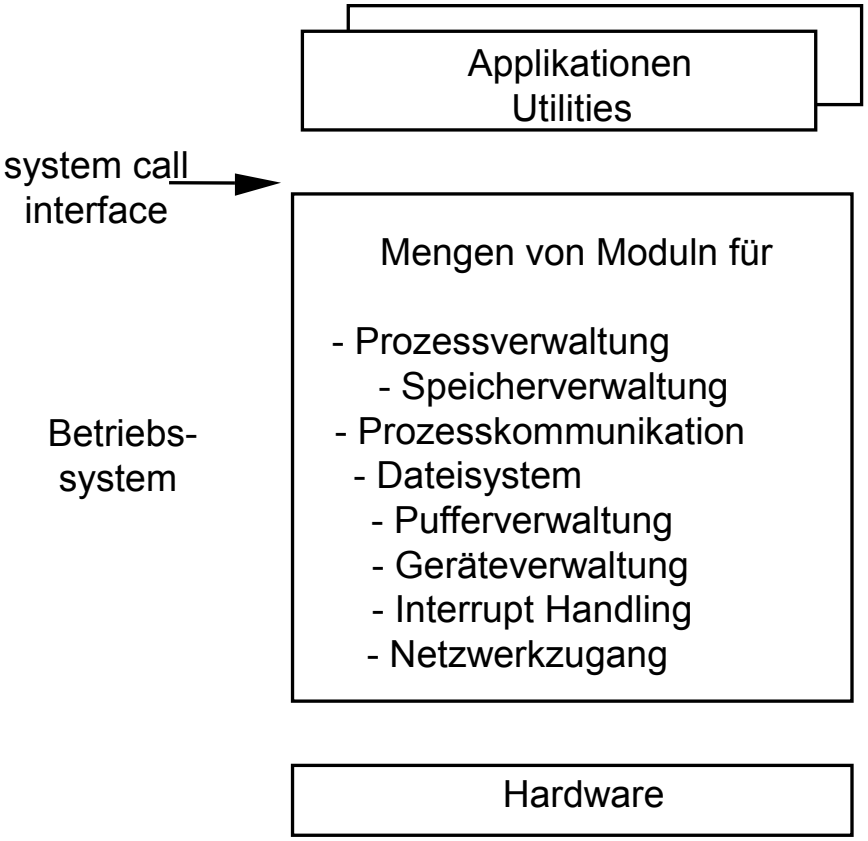
Kap. 2: Betriebssystem-Strukturen

2.1. Monolithische Systeme

Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim

Vorwiegende Struktur aller kommerziellen Betriebssysteme:
z.B. UNIX



Benutzermodus

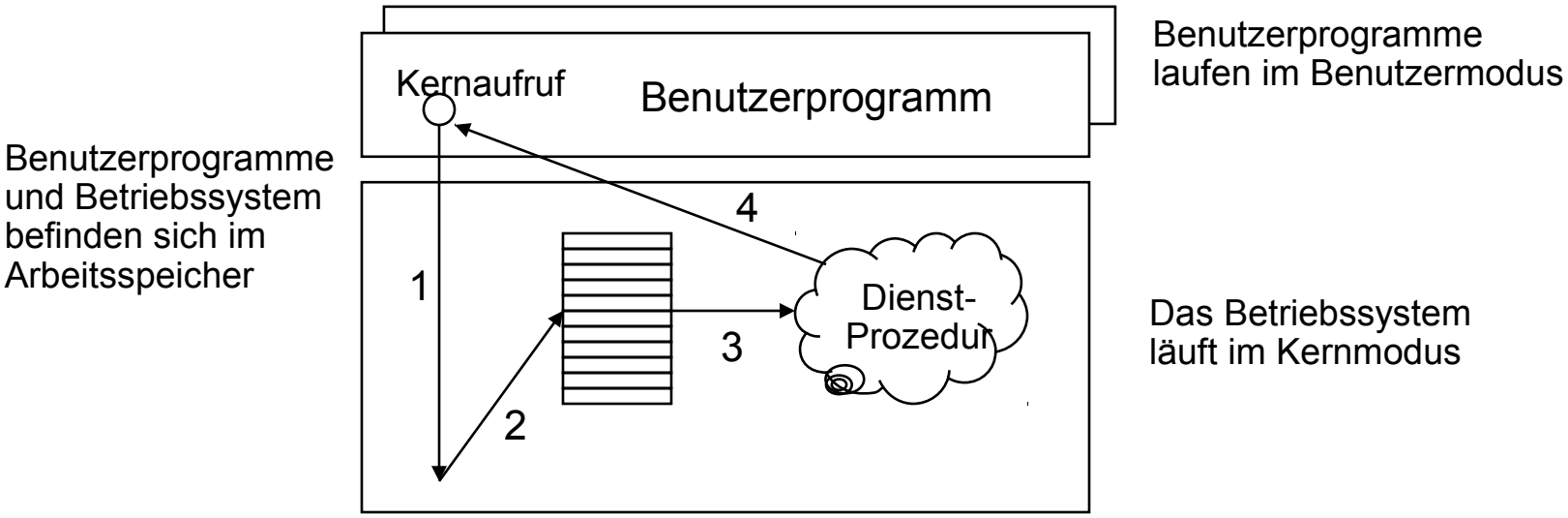
Trap-Befehl
=Kernaufruf
=Supervisor Call

Kernmodus
= Supervisor Mode

Durch Binder
zu einem "Kloß"
zusammengebunden
(bis auf dyn. ladbare
Kernmodule)

Durchführung eines Kernaufrufs

Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim



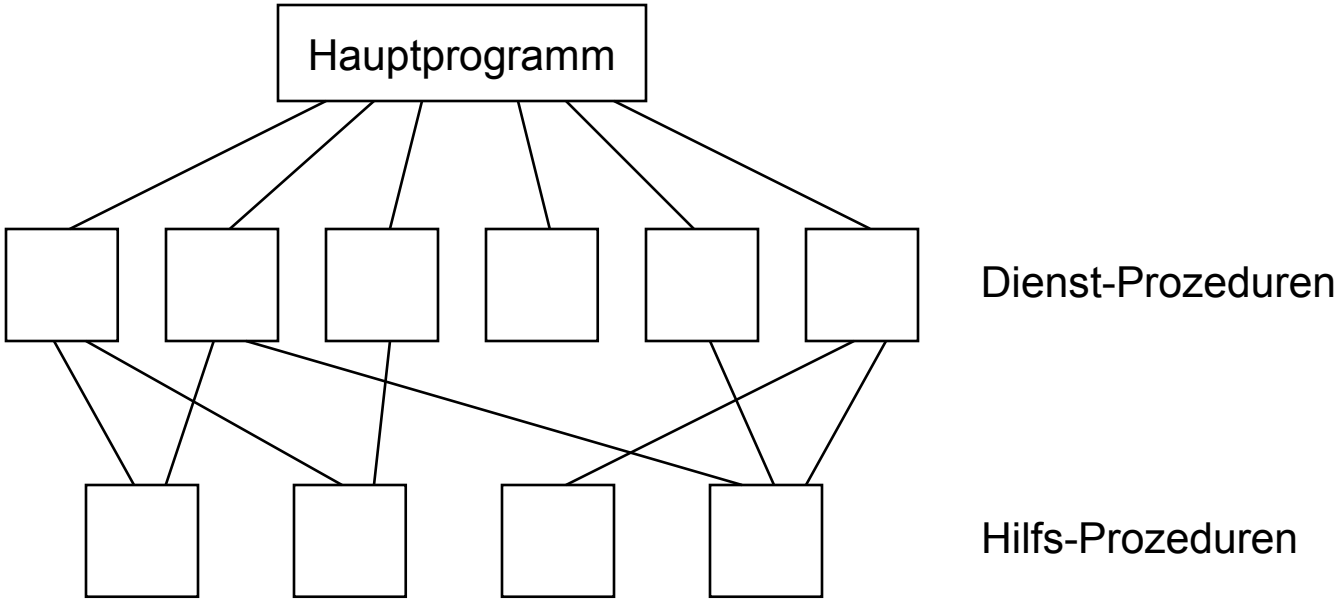
1: Benutzerprogramm springt über TRAP in den Kern und führt den Code selbst aus.
2: BS Code bestimmt die Nummer des angeforderten Dienstes.
3: BS Code lokalisiert Prozedur-Code für Systemaufruf und ruft sie auf.
4: Kontrolle wird an das Benutzerprogramm zurückgegeben.

Wichtig: Kern selbst ist passiv (Menge von Datenstrukturen und Prozeduren)

Einfaches Strukturmodell

Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim

Innere Struktur eines monolithischen BS:



Da der Betriebssystemkern passiv ist
und der Code aus einer Menge von Prozeduren besteht,
heißt ein solches Betriebssystem auch prozedurorientiert.

Beispiel: BSD-UNIX

Der UNIX Betriebssytemkern:

- monolithisch, aber portierbar
- Beispiel: 4.3BSD UNIX Kern (1987)
  - Lines of Code: 116 470 (nicht mehr !)
  - C - Anteil: 97.1 %
  - maschinenunabhängig: 41.5 %
  - maschinenabhängig: 58.5 %
    davon Gerätetreiber: 35.5 %
    Netzwerktreiber: 14.8 %

Neben dem Betriebssytemkern wird ein Großteil der UNIX-Systemfunktionalität durch sogenannte Dämon-Prozesse erbracht.

Vergleich Kernel: SLOC (ohne Leerzeilen , ohne Kommentarzeilen)
Linux 1.0.0 (1994) 176.250
Linux 2.2.0 (1999) 1.800.847
Linux 2.6.0 (2003) 5.929.913
Linux 3.2 (2012) 14.998.651
Windows Server 2003 (Gesamtsystem) ca. 50 Mio

(Vergleich zu sonstigen Codegrößen)

Project	No. of Files	eLOC
Linux Kernel 2.6.17	15,995	4,142,481
Firefox 1.5.0.2	10,970	2,172,520
MySQL 5.0.25	1973	894,768
PHP 5.1.6	1316	479,892
Apache Http 2.0.x	275	89,967

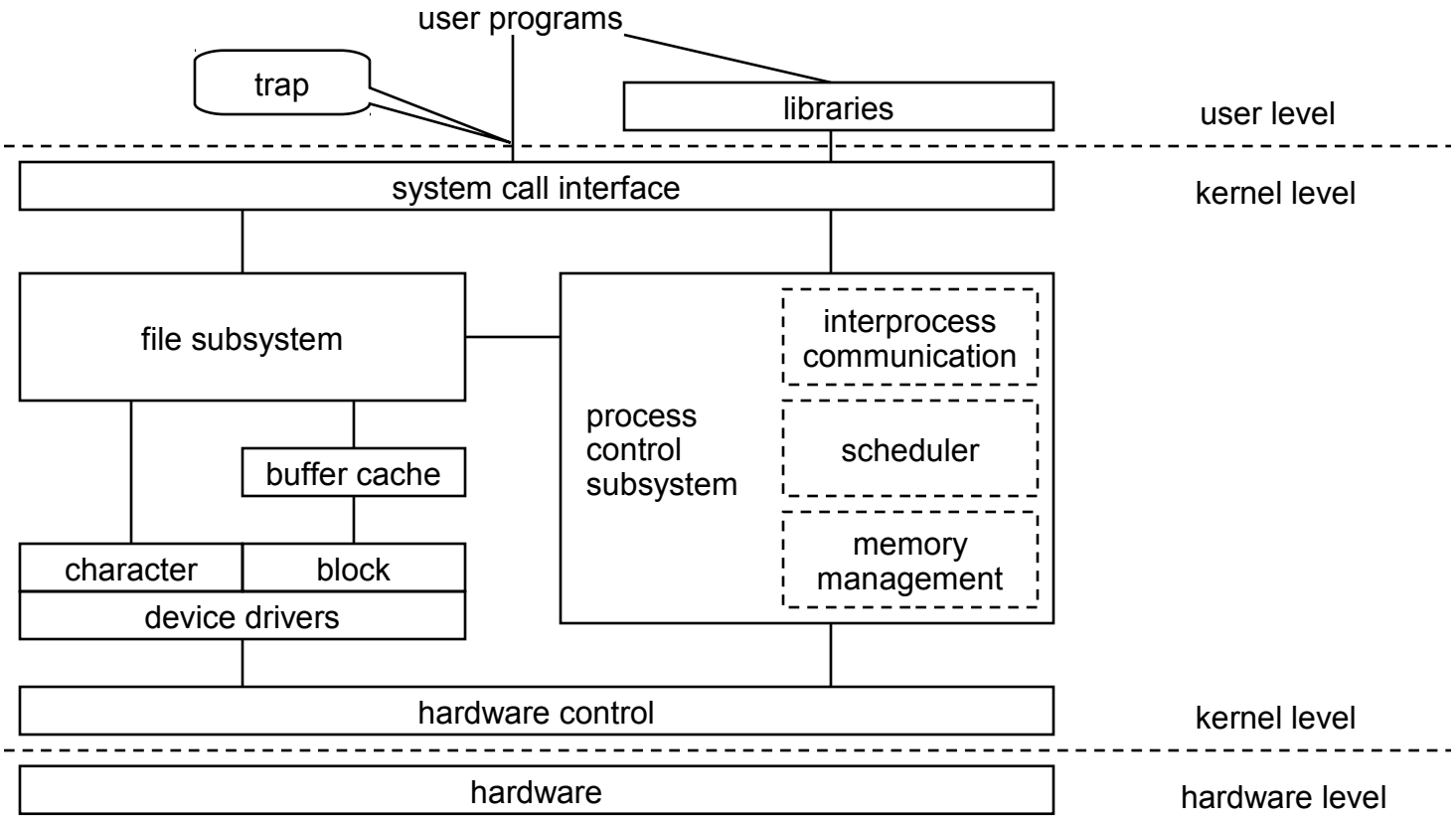
http://msquaredtechnologies.com/m2rsm/rsm_software_project_metrics.htm

The effective lines of code (eLOC) are measured using the following method:
1. Get the number of lines of code
2. Subtract whitespace lines
3. Subtract comment lines
4. Subtract the lines that contains only block constructs

Beispiel: UNIX (2)

Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim

Blockdiagramm des Systemkerns:



aus [Bach]: The Design of the UNIX Operating System

2.2. Geschichtete Systeme

Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim

Verallgemeinerung des monolithisches Ansatzes:

- BS als Hierarchie von Schichten (engl. layers).
- Jede Schicht abstrahiert von gewissen Restriktionen der darunterliegenden Schicht. Schicht benutzt Dienste der darunterliegenden Schicht.
- Erstes System: THE (Techn. Hochschule Eindhoven, Dijkstra, 1968, einfaches Stapelverarbeitungssystem in Pascal).

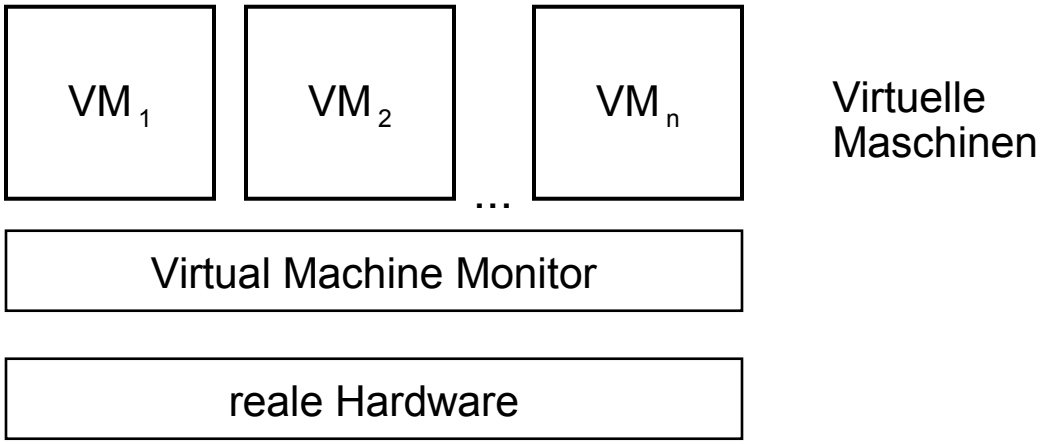
Schicht 5	Operateur
Schicht 4	Benutzerprogramme
Schicht 3	Ein- / Ausgabeverwaltung
Schicht 2	Operateur-zu-Prozess-Kommunikation
Schicht 1	Speicher- und Trommelverwaltung
Schicht 0	Prozessorvergabe und Multiprogramming

- Weitere Verallgemeinerung in MULTICS: "konzentrische (Schutz-) Ringe", verbunden mit nach innen zunehmender Privilegierung, kontrollierter Aufruf zwischen den Ebenen zur Laufzeit.

2.3. Virtuelle Maschinen

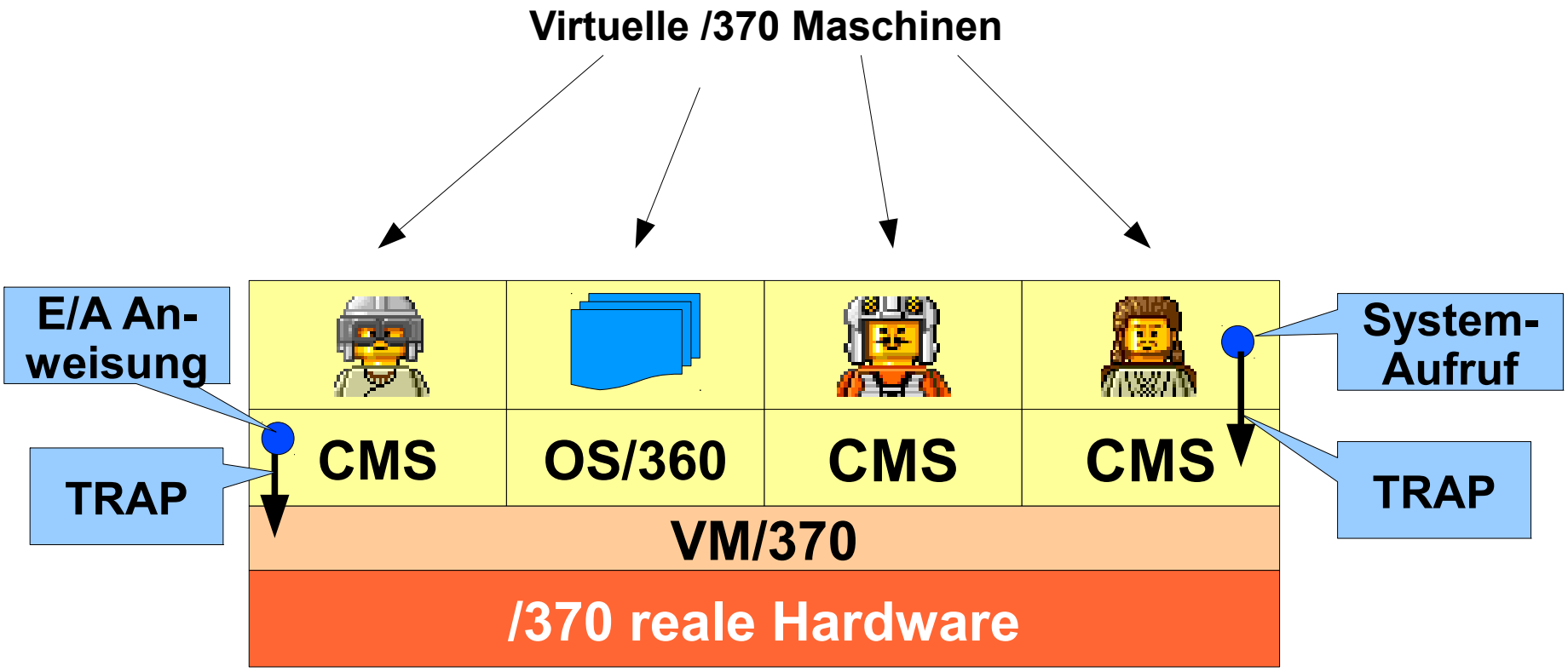
Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim

- Trennen der Funktionen Mehrprogrammbetrieb und erweiterte Maschine
- Virtualisierung durch "Virtual Machine Monitor":
  virtuelle Maschinen als mehr oder weniger identische Kopien der unterliegenden Hardware
- In jeder virtuellen Maschine: übliches Betriebssystem.



Beispiel: VM/370

Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim

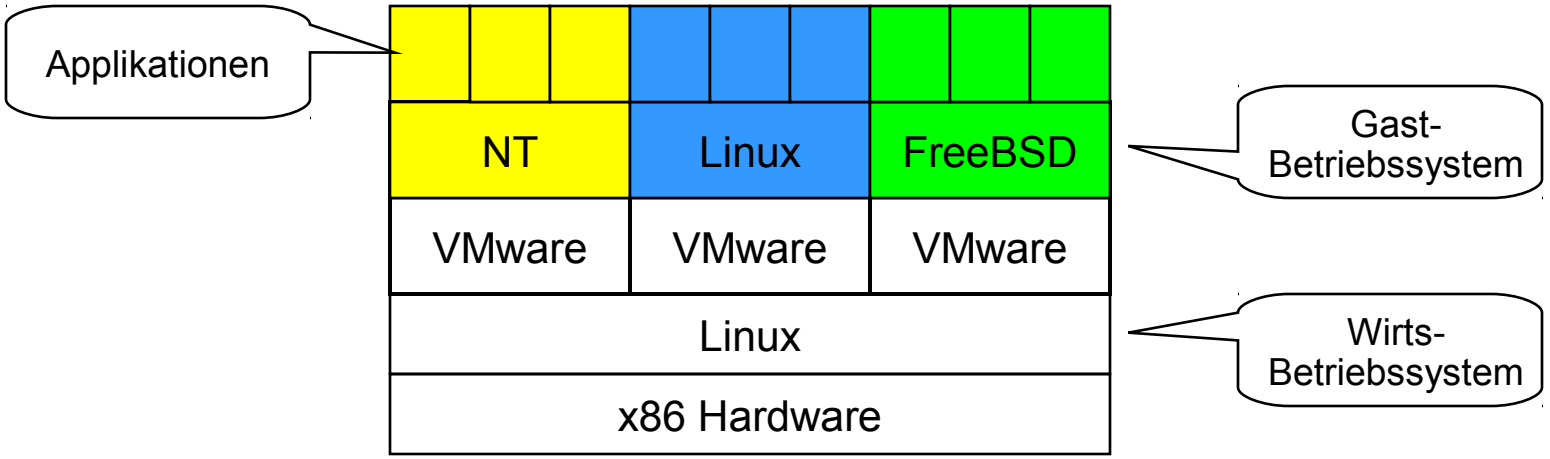


Beispiel: VM/370 (2)

- Das offizielle IBM-Produkt für Tmesharing-Betrieb der /360, TSS/360, kam zu spät, war zu groß und zu langsam.
- In der Zwischenzeit: IBM Scientific Center Cambridge, Mass. Eigenentwicklung, wurde als Produkt (ursprünglich CP/CMS) akzeptiert, erlangte als VM/370 weite Verbreitung.
- Unterste Ebene: virtuelle Maschinen als identische Kopien der unterliegenden Hardware mit Nachbildung von Benutzer/Supervisor-Modi, I/O, Unterbrechungen, (Simulation mehrerer /370 Rechner).
- Betriebssysteme in virtuellen Maschinen:
  z.B. ein Stapelverarbeitungssystem (OS/360) und eine Menge von Einbenutzer-Dialogsystemen (CMS, Conversational Monitor System) gleichzeitig möglich.
- Heute: z/VM: erlaubt z.B. 100 unabhängige Linux-Systeme auf einem IBM Mainframe.

Beispiel: VMware Workstation

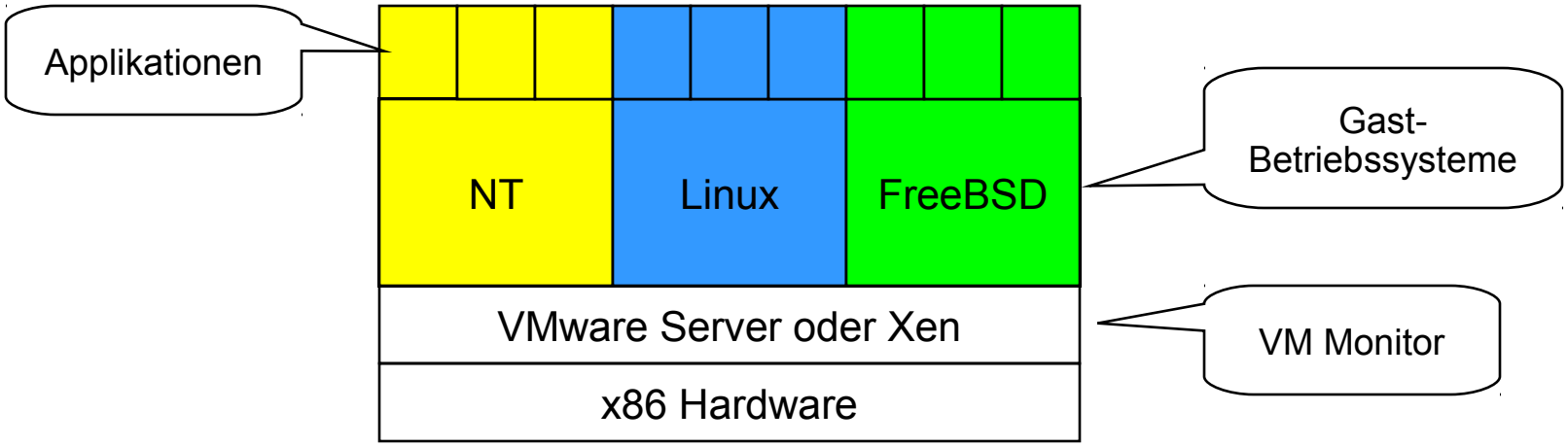
Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim



- erlaubt beliebige Betriebssysteme für x86-Architektur auf Linux oder Windows
- Jedes Gastbetriebssystem kann abstürzen, ohne den Rest zu beeinflussen

Beispiele: VMware Server, Xen

Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim



- Xen: Paravirtualisierung: Gastsysteme müssen angepasst werden (Quellcode Voraussetzung)
- VMware Server: klassischer VM Monitor

2.4. Client/Server-Struktur (µkern)

Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim

- Problem monolithischer Systeme: Kernelcode wird immer umfangreicher und komplexer, damit zwangsläufig auch fehlerträchtiger (z.B. Linux 2.6.x: ca. 5.7 Mio Zeilen)
- Aller Code, der im privilegierten Modus läuft, hat Zugriff auf alle Betriebsmittel und zählt damit immer zur „Trusted Code Base“.
- Nicht alle Anwendungen benötigen wirklich alle Dienste, die ein Kernel anbietet
- Art und Anzahl der Dienste werden aber durch den Kernel vorgegeben
- **Mikrokern**-Ansatz: Alle Funktionen, die für ihre Funktion nicht im privilegierten Modus arbeiten müssen, werden aus dem Kernel ausgelagert

Client/Server-Struktur (2)

Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim

Mikrokern-Ansatz:

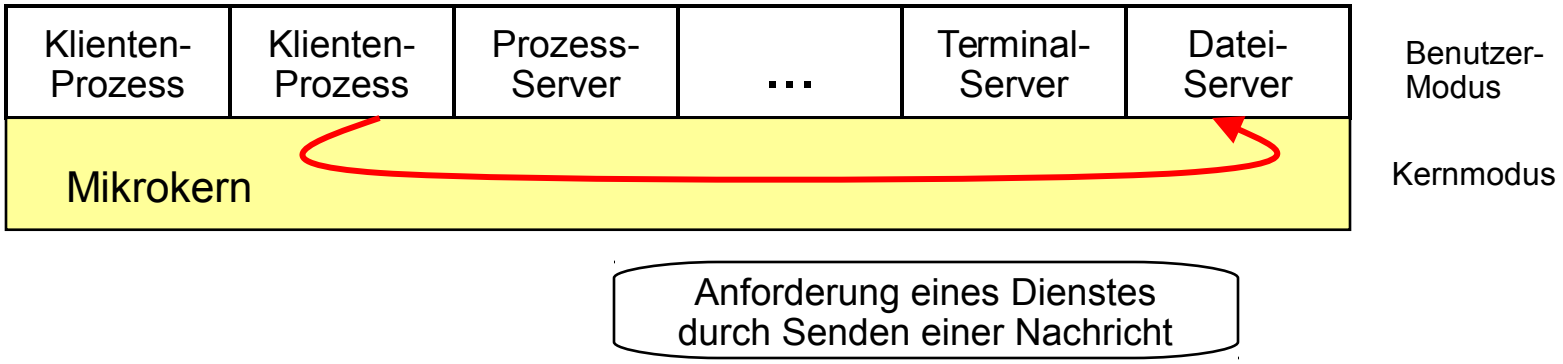
- Dienste wie Dateisystem, Netzwerkprotokolle, Speicherverwaltung, Prozesssteuerung, sogar Gerätetreiber müssen nicht zwangsläufig im Kernel angesiedelt sein.
- „Server“-Prozesse, die wie Anwenderprogramme ohne besondere Privilegien arbeiten, übernehmen diese Aufgaben
- Prinzip: Trennung von Strategien und Mechanismen (*separation of policy and mechanism*)
- Der verbleibende *Mikro*kern bietet nur noch Dienste zur Kommunikation der Server untereinander an.
- Er *sollte* daher wesentlich weniger komplex sein → kleinere, bzw. feingranularere „Trusted Code Base“

Client/Server-Struktur (3)

Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim

Ansatz für moderne Betriebssysteme:

- Auslagerung großer Teile der Funktionalität eines BS-Kerns in Benutzerprogramme (=lauffähig im Benutzermodus).
- Übrig bleibt minimaler BS-Kern, als Mikrokern bezeichnet (= "Infrastruktur").

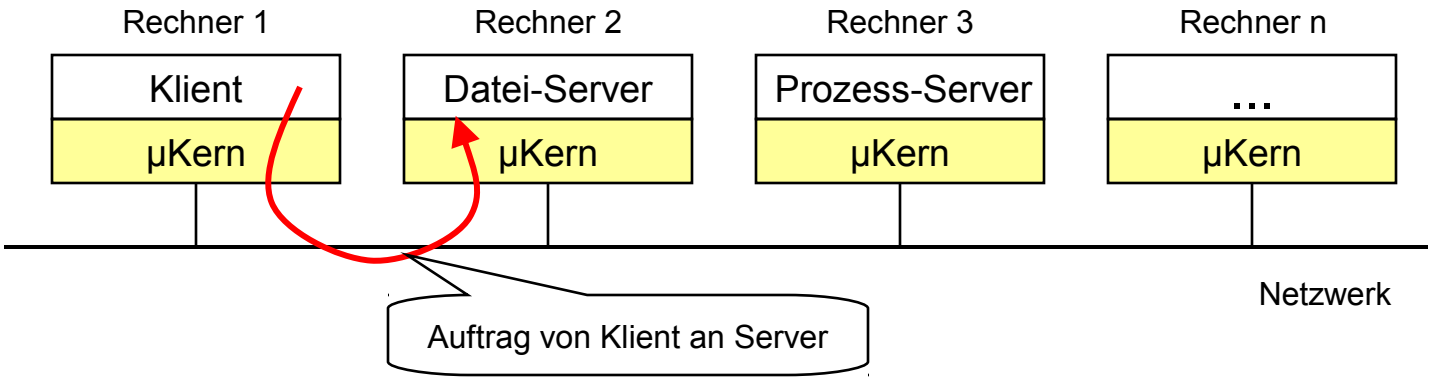


- Clients erhalten einen Dienst, indem sie Nachrichten an einen Serverprozess senden
- Mehrere Server können ihre Dienste in verschiedener, auf das jeweilige Ziel zugeschnittener Form anbieten
- Die Betriebssystemschnittstelle ist die Menge der Dienste, die ein Klient nutzt
- Jeder Klient kann „seine“ Schnittstelle selbst definieren
- Wie bei Virtualisierung sind mehrere BS-Schnittstellen in einem System möglich

Client/Server-Struktur (4)

Vorteile:

- Isolation einzelner "Systemteile" gegeneinander (Vermeidung von Fehlerausbreitung).
- Erweiterbarkeit, Anpassungsfähigkeit und flexible Konfigurierbarkeit insbesondere für Verteilte Systeme.

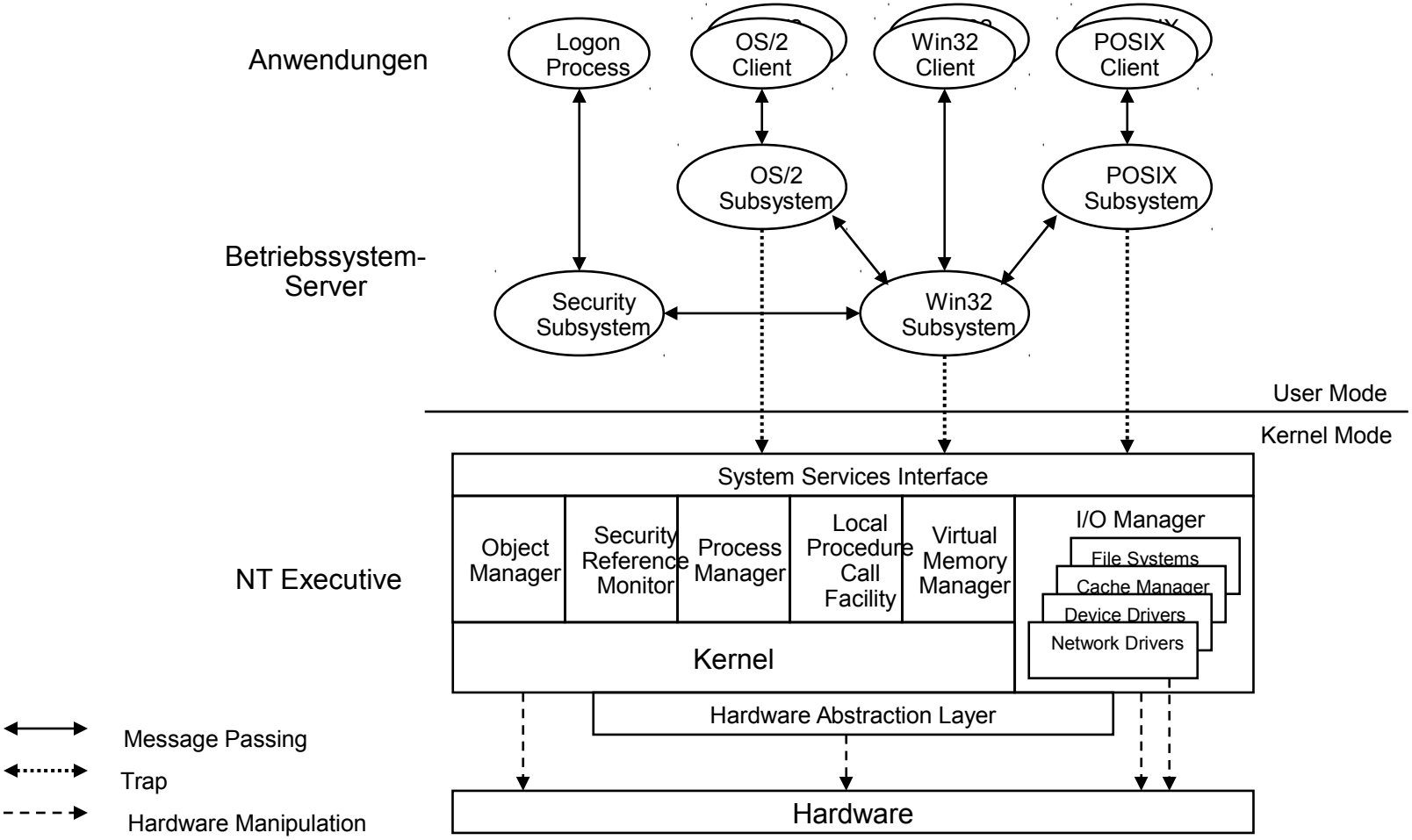


Beispiele: Mach, Chorus, Amoeba, Windows NT

Ein Betriebssystem, das auf einem über Nachrichten realisierten Beauftragungsprinzip beruht, heißt auch nachrichtenorientiert. Nachrichtenorientiertheit und Prozedurorientiertheit sind funktional gleichwertig, nachrichtenorientierte Systeme leiden aber häufig unter Ineffizienz.

Client/Server-Struktur (5)

Beispiel: Windows NT



2.5 Zusammenfassung

1. Grundverständnis einer Betriebssystemschnittstelle
2. Strukturierungsprinzipien von Betriebssystemen:
   - Monolithische Struktur
   - Hierarchie von Schichten
   - Virtuelle Maschinen
   - Client/Server-Struktur (Microkernel)