

# B Einführung

---

## Systemprogrammierung I

© 1997-2003, F. J. Hauck, W. Schröder-Preikschat, Inf 4, FAU Erlangen-Nürnberg[B-Intro.fm, 2003-10-22 20.25]  
Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

B - 1

## 1 Warum Systemprogrammierung I?

---

- Rasche Einarbeitung in spezielle Systeme
  - ◆ MVS, BS2000, VM, Solaris, Unix, Windows NT, Windows 95/98, MS/DOS
- Strukturierung komplexer Programmsysteme
  - ◆ Unterteilung in interagierende Komponenten
- Konzeption und Implementierung spezialisierter Systeme
  - ◆ Eingebettete Systeme (*Embedded Systems*)
  - ◆ Automatisierungssysteme
- Erstellung fehlertoleranter Systeme
- Verständnis für Abläufe im Betriebssystem
  - ◆ ökonomische Nutzung der Hardware
  - ◆ Laufzeitoptimierung anspruchsvoller Anwendungen

## Systemprogrammierung I

© 1997-2003, F. J. Hauck, W. Schröder-Preikschat, Inf 4, FAU Erlangen-Nürnberg[B-Intro.fm, 2003-10-22 20.25]  
Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

B - 2

## 1.1 Phänomene der Speicherverwaltung

### ■ Beispiel: Initialisierung von großen Matrizen

#### ◆ Variante 1:

```
#define DIM 6000

int main()
{
    long i, j;
    static long matrix[DIM][DIM];

    for( i = 0; i < DIM; i++ )
        for( j = 0; j < DIM; j++ )
            matrix[i][j] = 1;

    return 0;
}
```

## 1.1 Phänomene der Speicherverwaltung (2)

### ■ Beispiel: Initialisierung von großen Matrizen

#### ◆ Variante 2:

```
#define DIM 6000

int main()
{
    long i, j;
    static long matrix[DIM][DIM];

    for( j = 0; j < DIM; j++ )
        for( i = 0; i < DIM; i++ )
            matrix[i][j] = 1;

    return 0;
}
```

#### ◆ Schleifen sind vertauscht

## 1.1 Phänomene der Speicherverwaltung (3)

- Messergebnisse (Sun UltraSparc 1, 128M Hauptspeicher)
    - ◆ Variante 1:  
Benutzerzeit= 3,69 sec; Systemzeit= 1,43 sec; Gesamtzeit= 22,03 sec
    - ◆ Variante 2:  
Benutzerzeit= 21,86 sec; Systemzeit= 2,33 sec; Gesamtzeit= 86,39 sec
  - Ursachen
    - ◆ Variante 1 geht sequentiell durch den Speicher
    - ◆ Variante 2 greift versetzt ständig auf den gesamten Speicherbereich zu
- Beispiel: `matrix[4][4]` und die ersten fünf Zugriffe

Variante 1 

1	2	3	4	5															
---	---	---	---	---	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Variante 2 

1	5			2				3						4					
---	---	--	--	---	--	--	--	---	--	--	--	--	--	---	--	--	--	--	--

## 1.1 Phänomene der Speicherverwaltung (4)

- Ursachen
  - ◆ logischer Adressraum
    - benutzte Adressen sind nicht die physikalischen Adressen
    - Abbildung wird durch Hardware auf Seitenbasis vorgenommen (Seitenadressierung)
    - Variante 2 hat weniger Lokalität, d.h. benötigt häufig wechselnde Abbildungen
  - ◆ virtueller Speicher
    - möglicher Adressraum ist größer als physikalischer Speicher
    - auf Seitenbasis werden Teile des benötigten Speichers ein- und ausgelagert
    - bei Variante 2 muss viel mehr Speicher ein- und ausgelagert werden

## 1.2 Phänomene des Dateisystems

- Beispiel: sequentielles Schreiben mit unterschiedlicher Pufferlänge

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#define BUFLLEN 8191 /* 8192, 8193 */

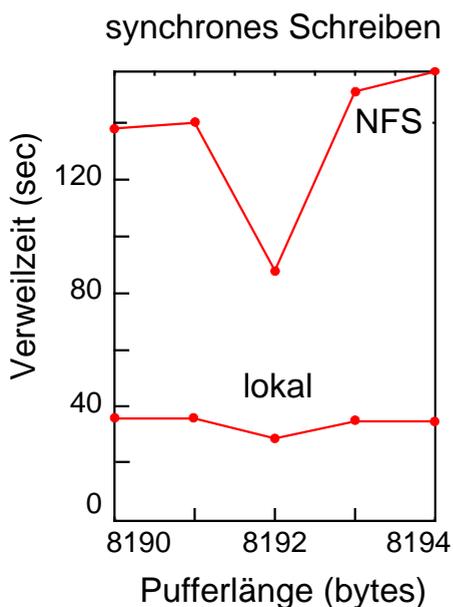
int main()
{
    static char buffer[BUFLLEN];
    int i, fd= open( "filename",
                    O_CREAT|O_TRUNC|O_WRONLY|O_SYNC,
                    S_IRUSR|S_IWUSR );

    for( i= 0; i < 1000; i++ )
        write( fd, buffer, BUFLLEN );

    return 0;
}
```

## 1.2 Phänomene des Dateisystems (2)

- Messergebnisse



## 1.2 Phänomene des Dateisystems (3)

### ■ Ursachen

- ◆ synchrones Schreiben erfordert sofortiges Rausschreiben der Daten auf die Platte
  - nötig beispielsweise, wenn hohe Fehlertoleranz gefordert wird
  - Daten auf der Platte sind immer auf dem neuesten Stand
- ◆ bereits kleine Abweichungen von der Blockgröße erfordern zusätzliche Blocktransfers
  - 8192 ist (hier) ein Vielfaches der Blockgröße der Plattenblöcke
  - nur Puffer mit diesem Vielfachen an Bytes können mit einem Ruck geschrieben werden

## 2 Überblick über die Vorlesung

### ★ Inhaltsübersicht

- A. Organisation
- B. Einführung
- C. Dateisysteme
- D. Prozesse und Nebenläufigkeit
- E. Speicherverwaltung
- F. Implementierung von Dateien
- G. Ein-, Ausgabe
- H. Verklemmungen
- I. Datensicherheit und Zugriffsschutz

### 3 Was sind Betriebssysteme?

#### ■ DIN 44300

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

#### ■ Tanenbaum

- ◆ „...eine Software-Schicht ..., die alle Teile des Systems verwaltet und dem Benutzer eine Schnittstelle oder eine *virtuelle Maschine* anbietet, die einfacher zu verstehen und zu programmieren ist [als die nackte Hardware].“

### 3 Was sind Betriebssysteme? (2)

#### ■ Silberschatz/Galvin

- ◆ „... ein Programm, das als Vermittler zwischen Rechnernutzer und Rechner-Hardware fungiert. Der Sinn des Betriebssystems ist eine Umgebung bereitzustellen, in der Benutzer bequem und effizient Programme ausführen können.“

#### ■ Brinch Hansen

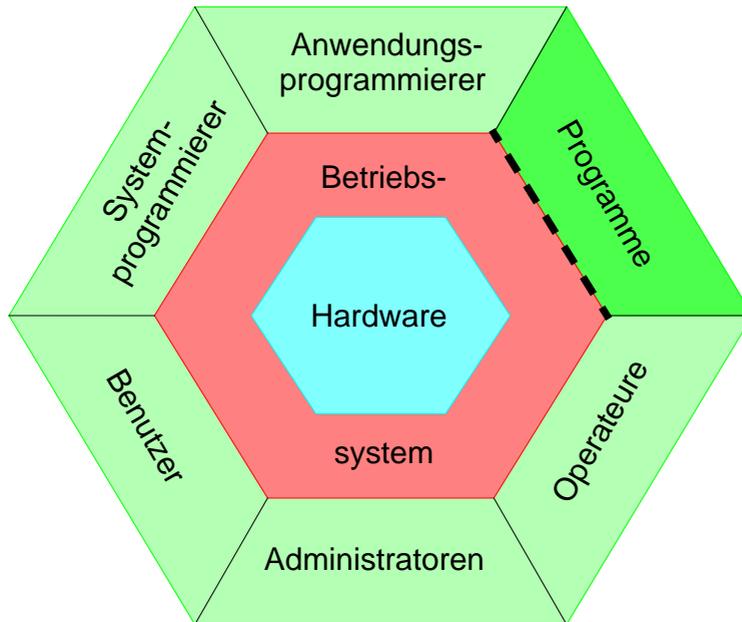
- ◆ „... der Zweck eines Betriebssystems [liegt] in der Verteilung von Betriebsmitteln auf sich bewerbende Benutzer.“

#### ★ Zusammenfassung

- ◆ Software zur Betriebsmittelverwaltung
- ◆ Bereitstellung von Grundkonzepten zur statischen und dynamischen Strukturierung von Programmsystemen

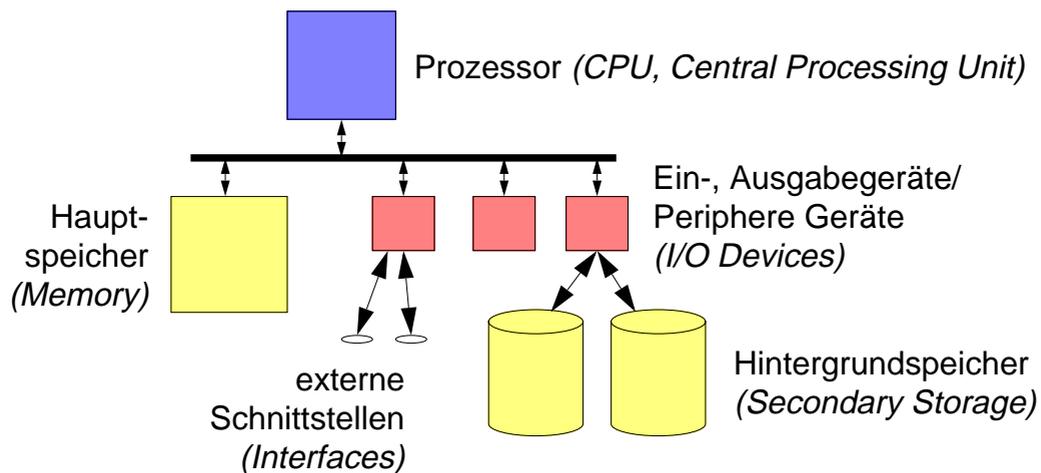
### 3 Was sind Betriebssysteme? (3)

#### ■ strukturelle Einordnung



### 3.1 Verwaltung von Betriebsmitteln

#### ■ Hardware-Betriebsmittel eines Rechners



## 3.1 Verwaltung von Betriebsmitteln (2)

- resultierende Aufgaben
  - ◆ Multiplexen von Betriebsmitteln für mehrere Benutzer bzw. Anwendungen
  - ◆ Schaffung von Schutzumgebungen
  
- Ermöglichen einer koordinierten gemeinsamen Nutzung von Betriebsmitteln, klassifizierbar in
  - ◆ aktive, zeitlich aufteilbare (Prozessor)
  - ◆ passive, nur exklusiv nutzbare (periphere Geräte, z.B. Drucker u.Ä.)
  - ◆ passive, räumlich aufteilbare (Speicher, Plattenspeicher u.Ä.)
  
- Unterstützung bei der Fehlererholung

## 3.2 Schnittstellen

- Betriebssystem soll helfen, Benutzervorstellungen auf die Maschinengegebenheiten abzubilden
  - ◆ Bereitstellung geeigneter Abstraktionen und Schnittstellen für
    - Benutzer:**  
Dialogbetrieb, graphische Benutzeroberflächen
    - Anwendungsprogrammierer:**  
Programmiersprachen, Modularisierungshilfen, Interaktionsmodelle (Programmiermodell)
    - Systemprogrammierer:**  
Werkzeuge zur Wartung und Pflege
    - Operateure:**  
Werkzeuge zur Gerätebedienung und Anpassung von Systemstrategien

## 3.2 Schnittstellen (2)

### Administratoren:

Werkzeuge zur Benutzerverwaltung, langfristige Systemsteuerung

### Programme:

„*Supervisor Calls (SVC)*“,

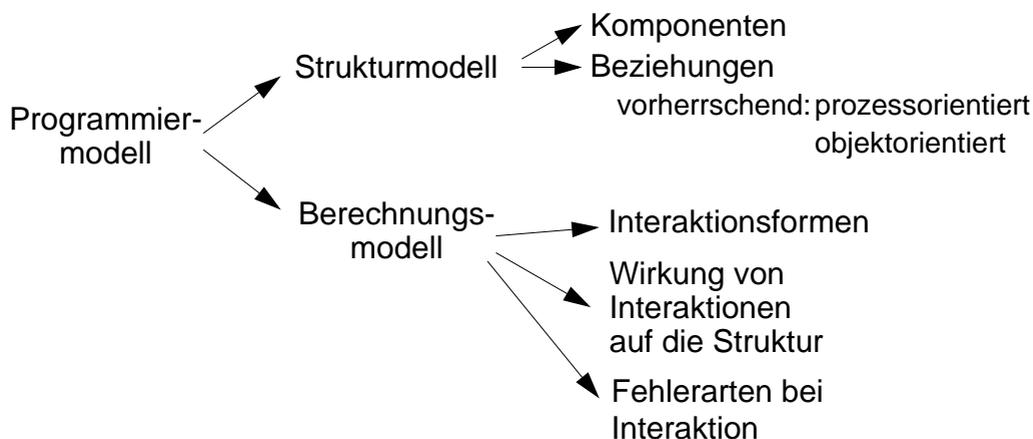
„*Application Programmer Interface (API)*“

### Hardware:

Gerätetreiber

## 3.3 Programmiermodelle

- Betriebssystem realisiert ein Programmiermodell
  - ◆ keine Notwendigkeit genauer Kenntnisse über Hardwareeigenschaften und spezielle Systemsoftwarekomponenten
  - ◆ Schaffung einer begrifflichen Basis zur Strukturierung von Programmsystemen und ihrer Ablaufsteuerung



## 3.3 Programmiermodelle (2)

- Beispiele für Strukturkomponenten
  - ◆ Klassen (Vorlagen zur Bildung von Instanzen)
  - ◆ Instanzen/Objekte
  - ◆ Prozeduren
  - ◆ **Dateien** (Behälter zur langfristigen Speicherung von Daten)
  - ◆ **Prozesse** (in Ausführung befindliche Programme)
    - besser: *Prozessinkarnationen*
  - ◆ **“Buchsen”** (Kommunikationsendpunkte; *sockets*)
  - ◆ **“Röhren”** (Nachrichtenkanäle; *pipes*)
- Beispiele für Beziehungen
  - ◆ A kann B referenzieren, beauftragen, aufrufen, modifizieren
  - ◆ Röhre R verbindet A und B

## 3.3 Programmiermodelle (3)

- Beispiele für Interaktionsformen
  - ◆ Prozedur-(Methoden-)Aufruf
  - ◆ Nachrichtenaustausch
  - ◆ Gemeinsame Speichernutzung
- Wirkung von Interaktionen auf die Struktur
  - ◆ Erzeugung und Tilgung von Prozessen
  - ◆ Instanziierung von Objekten
- Fehlerarten bei Interaktion
  - ◆ Verlust, Wiederholung oder Verspätung von Nachrichten
  - ◆ Abbruch aufgerufener Methoden, Ausnahmebehandlung

## 3.4 Ablaufmodelle

---

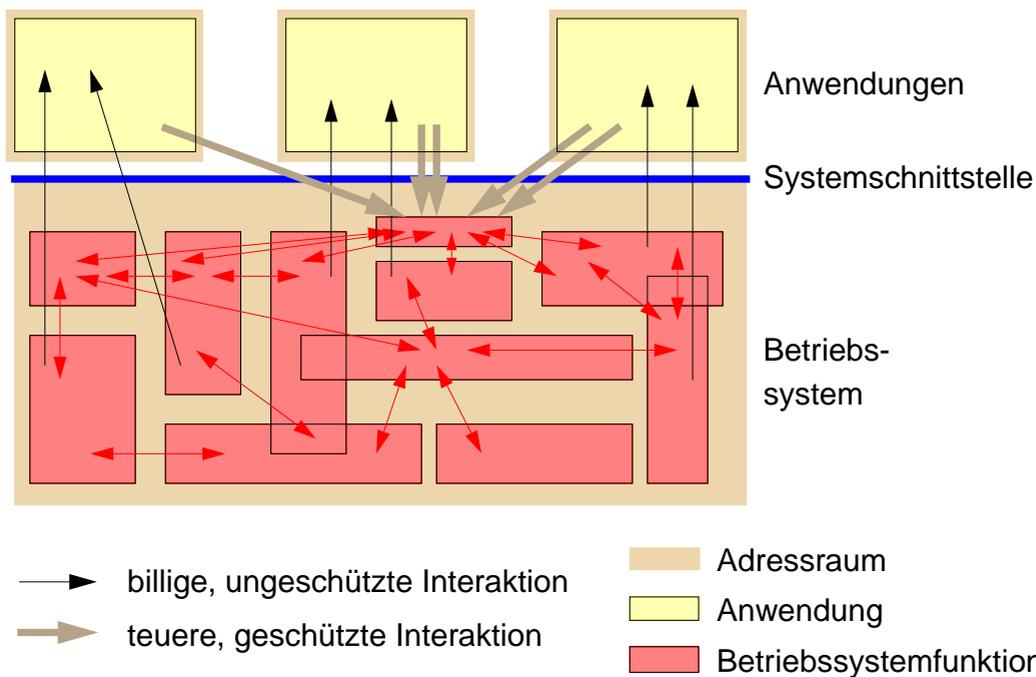
- Betriebssystem realisiert eine Ablaufumgebung
- Bereitstellung von Hilfsmitteln zur Bearbeitung von Benutzerprogrammen und zur Steuerung ihrer Abläufe
  - ◆ Laden, Starten (ggf. auch Stoppen und Fortführen) von Programmen
  - ◆ Überwachung des Programmablaufs
  - ◆ Beenden und Eliminieren von Programmen
  - ◆ Abrechnung (*Accounting*)

## 4 Betriebssystemarchitekturen

---

- Umfang zehntausende bis mehrere Millionen Befehlszeilen
  - ◆ Strukturierung hilfreich
- Verschiedene Strukturkonzepte
  - ◆ monolithische Systeme
  - ◆ geschichtete Systeme
  - ◆ Minimalkerne
  - ◆ offene objektorientierte Systeme

## 4.1 Monolithische Systeme



### Systemprogrammierung I

© 1997-2003, F. J. Hauck, W. Schröder-Preikschat, Inf 4, FAU Erlangen-Nürnberg[B-Intro.fm, 2003-10-22 20.25]  
Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

B - 23

## 4.1 Monolithische Systeme (2)

### ★ Vorteile

- ◆ effiziente Kommunikation und effizienter Datenzugriff innerhalb des Kerns
- ◆ privilegierte Befehle jederzeit ausführbar

### ▲ Nachteile

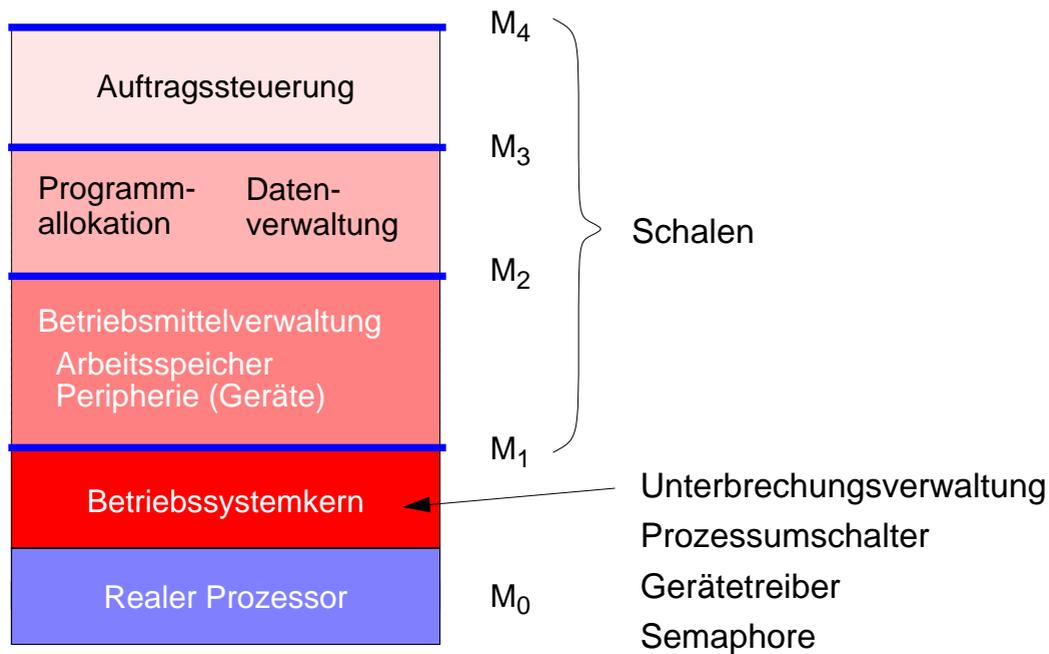
- ◆ keine interne Strukturierung (änderungsunfreundlich, fehleranfällig)
- ◆ kein Schutz zwischen Kernkomponenten (Problem: zugekaufte Treiber)

### Systemprogrammierung I

© 1997-2003, F. J. Hauck, W. Schröder-Preikschat, Inf 4, FAU Erlangen-Nürnberg[B-Intro.fm, 2003-10-22 20.25]  
Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

B - 24

## 4.2 Geschichtete Systeme



### Systemprogrammierung I

© 1997-2003, F. J. Hauck, W. Schröder-Preikschat, Inf 4, FAU Erlangen-Nürnberg[B-Intro.fm, 2003-10-22 20.25]  
Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

B – 25

## 4.2 Geschichtete Systeme (2)

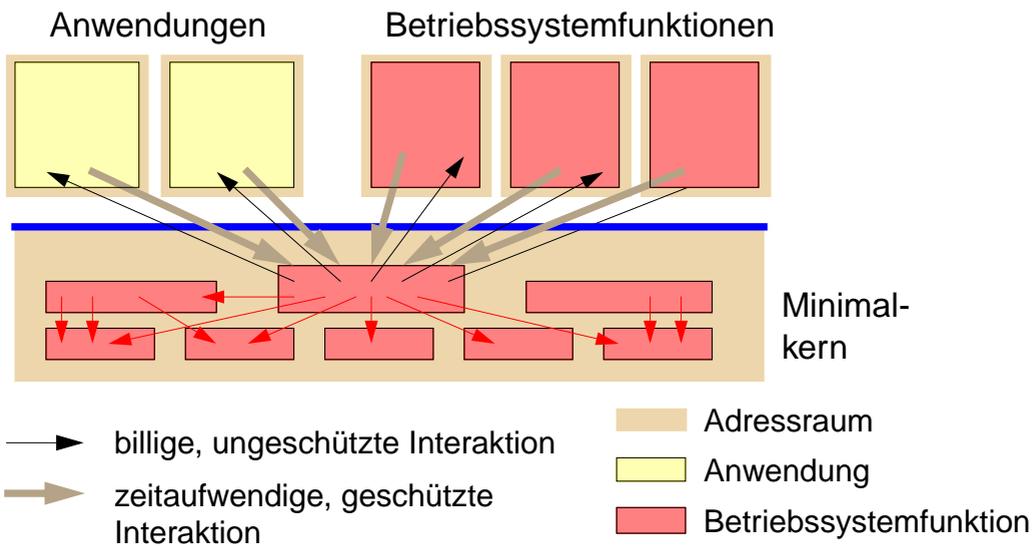
- ★ Vorteile
  - ◆ Schutz zwischen verschiedenen BS-Teilen
  - ◆ interne Strukturierung
- ▲ Nachteile
  - ◆ mehrfacher Schutzraumwechsel ist zeitaufwendig
  - ◆ unflexibler und nur einseitiger Schutz (von unten nach oben)

### Systemprogrammierung I

© 1997-2003, F. J. Hauck, W. Schröder-Preikschat, Inf 4, FAU Erlangen-Nürnberg[B-Intro.fm, 2003-10-22 20.25]  
Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

B – 26

## 4.3 Minimalkerne



### ★ Vorteile

- ◆ gute Modularisierung
- ◆ Schutz der Komponenten voreinander

### ▲ Nachteil

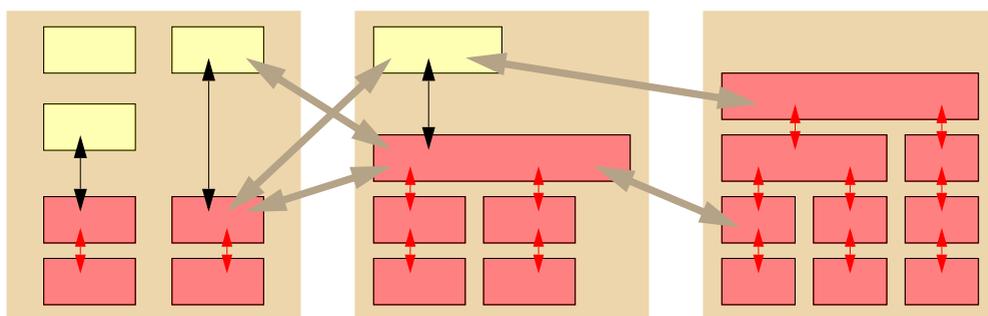
- ◆ Kommunikation zwischen Modulen ist zeitaufwendig

### Systemprogrammierung I

© 1997-2003, F. J. Hauck, W. Schröder-Preikschat, Inf 4, FAU Erlangen-Nürnberg[B-Intro.fm, 2003-10-22 20.25]  
 Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

B - 27

## 4.4 Objektbasierte, offene Systeme



- ▶ schnelle, durch Objektkapselung geschützte Interaktion  
 —▶ langsame, durch Adressraumgrenze geschützte Interaktion
- Adressraum  
 Anwendungsobjekte  
 Betriebssystemobjekte

### ■ Sicherung der Modulgrenzen durch Programmiermodell und Software

- ◆ z.B. Objektorientierung und Byte-Code-Verifier in Java

### Systemprogrammierung I

© 1997-2003, F. J. Hauck, W. Schröder-Preikschat, Inf 4, FAU Erlangen-Nürnberg[B-Intro.fm, 2003-10-22 20.25]  
 Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

B - 28

## 4.4 Objektbasierte, offene Systeme (2)

### ★ Vorteile

- ◆ Schutz auf mehreren Ebenen (Sprache, Code-Prüfung, Adressraum)
- ◆ Modularisierung und Effizienz möglich

### ▲ Nachteile

- ◆ komplexes Sicherheitsmodell

## 5 Geschichtliche Entwicklung

### 5.1 1950–1960

1950

- ◆ einströmige Stapelsysteme  
(*Single-stream batch processing systems*)  
Aufträge zusammen mit allen Daten werden übergeben und sequentiell bearbeitet

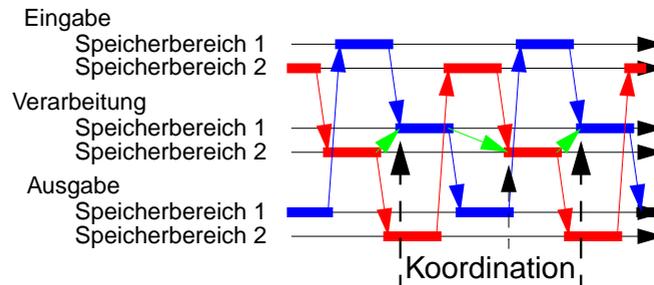
- ◆ Steuerung durch Auftragsabwickler  
(*Resident monitor, Job monitor*)  
Hilfsmittel: Assembler, Übersetzer (*Compiler*), Binder und Lader; Programmbibliotheken

1960

## 5.2 1960–1965

1960

- ◆ autonome periphere Geräte
  - eine Überlappung von Programmbearbeitung und Datentransport zwischen Arbeitsspeicher und peripheren Geräten möglich
  - Wechselpufferbetrieb (abwechselndes Nutzen zweier Puffer)



1965

### Systemprogrammierung I

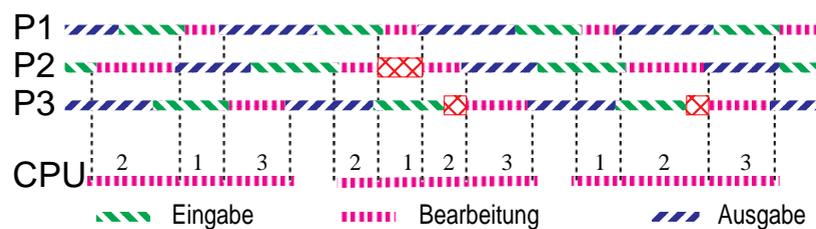
© 1997-2003, F. J. Hauck, W. Schröder-Preikschat, Inf 4, FAU Erlangen-Nürnberg[B-Intro.fm, 2003-10-22 20.25]  
Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

B – 31

## 5.2 1960-1965 (2)

1960

- ◆ autonome periphere Geräte (Forts.)
  - Mehrprogrammbetrieb (*Multiprogramming*)



- Spooling (*Simultaneous peripheral operation on-line*)
- ◆ mehrere Programme müssen gleichzeitig im Speicher sein → Auslagern von Programmen auf Sekundärspeicher
- ◆ Programme müssen während des Ablaufs verlagerbar sein (*Relocation problem*)
- ◆ Echtzeitdatenverarbeitung (*Real-time processing*), d.h. enge Bindung von Ein- und Ausgaben an die physikalische Zeit

1965

### Systemprogrammierung I

© 1997-2003, F. J. Hauck, W. Schröder-Preikschat, Inf 4, FAU Erlangen-Nürnberg[B-Intro.fm, 2003-10-22 20.25]  
Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

B – 32

## 5.3 1965–1970

1965

OS/360

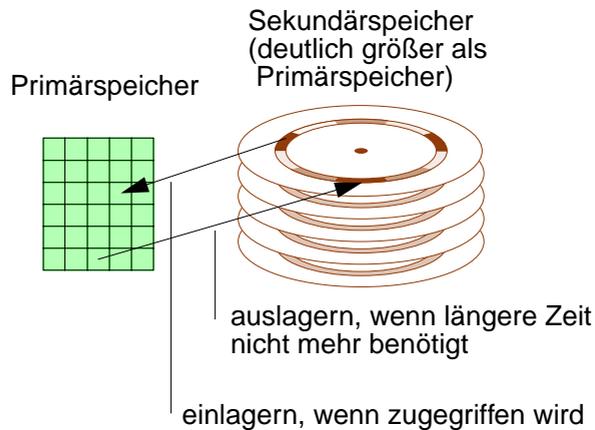
- ◆ Umsetzung von Programmadressen in Speicherorte zur Laufzeit: Segmentierung, Seitenadressierung (*Paging*)
- ◆ virtueller Adressraum: Seitentausch (*Paging*)  
Seiten werden je nach Zugriff ein- und ausgelagert

THE

MULTICS

UNIX

1970



## 5.3 1965-1970 (2)

1965

OS/360

- ◆ interaktiver Betrieb (*Interactive processing, Dialog mode*)
- ◆ Mehrbenutzerbetrieb, Teilnehmersysteme (*Time sharing*)

THE

MULTICS

UNIX

1970

- ◆ Problem: Kapselung von Prozessen und Dateien → geschützter Adressraum, Zugriffsschutz auf Dateien
- ◆ Dijkstra: Programmsysteme als Menge kooperierender Prozesse (heute *Client-Server*)
- ◆ Problem: Prozessinteraktion bei gekapselten Prozessen → Nachrichtensysteme zur Kommunikation, gemeinsamer Speicher zur Kooperation

## 5.4 1970–1975

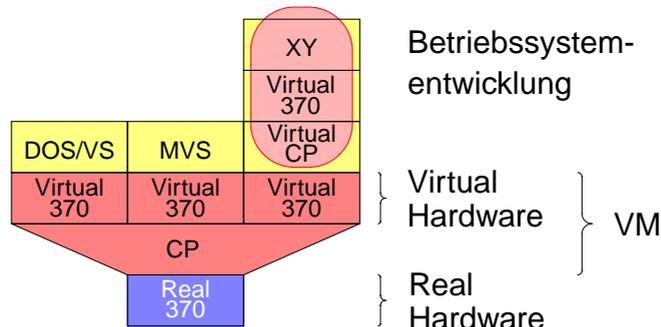
1970

VM

Hydra

MVS

- ◆ Modularisierung:  
Datenkapselung, Manipulation durch Funktionen (nach Parnas)
- ◆ virtuelle Maschinen: Koexistenz verschiedener Betriebssysteme im gleichen Rechner



- ◆ symmetrische Multiprozessoren: HYDRA
  - Zugangskontrolle zu Instanzen durch *Capabilities*
  - Trennung von Strategie und Mechanismus

1975

### Systemprogrammierung I

© 1997-2003, F. J. Hauck, W. Schröder-Preikschat, Inf 4, FAU Erlangen-Nürnberg[B-Intro.fm, 2003-10-22 20.25]  
Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

B – 35

## 5.5 1975–1985

1975

LOCUS

1980

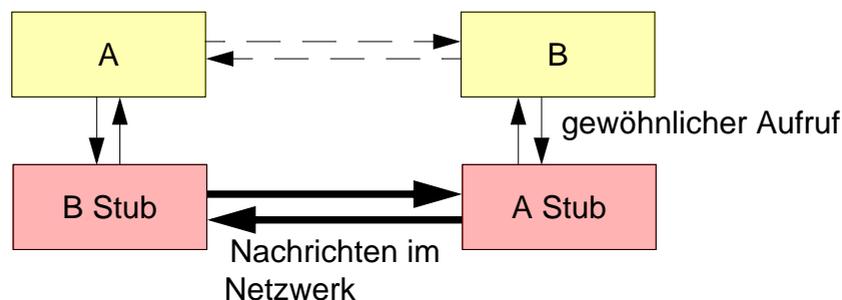
MS-DOS

NC

EDEN

1985

- ◆ Vernetzung, Protokolle (z.B. TCP/IP)
- ◆ Verteilte Systeme
- ◆ Newcastle Connection
- ◆ Fernaufruf (*Remote procedure call, RPC*)

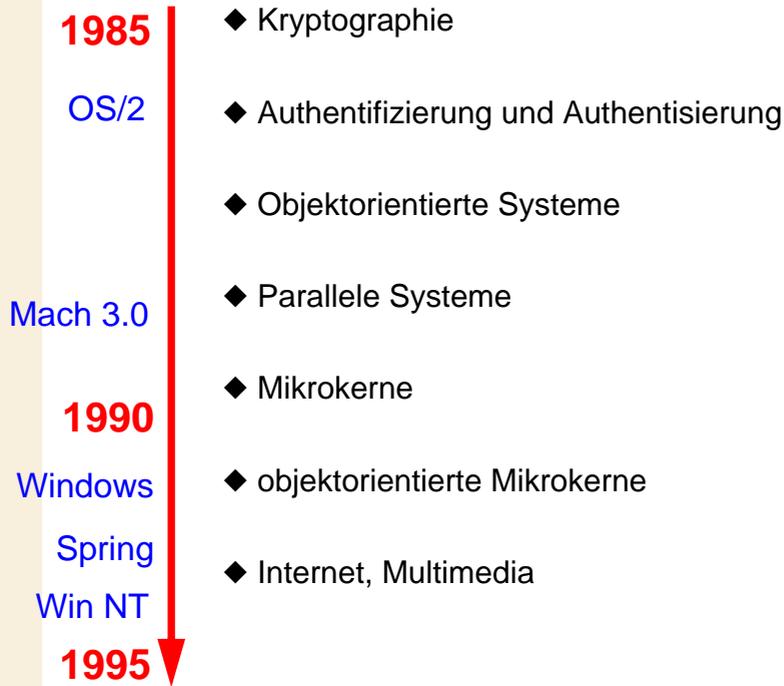


### Systemprogrammierung I

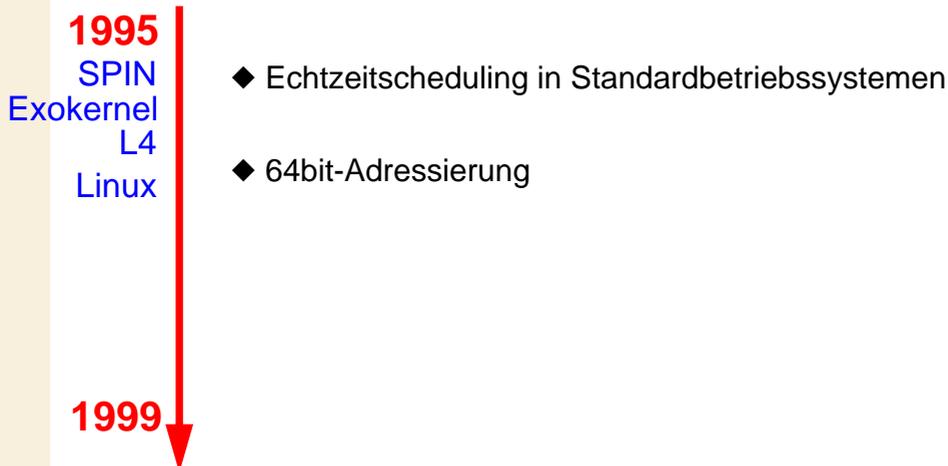
© 1997-2003, F. J. Hauck, W. Schröder-Preikschat, Inf 4, FAU Erlangen-Nürnberg[B-Intro.fm, 2003-10-22 20.25]  
Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

B – 36

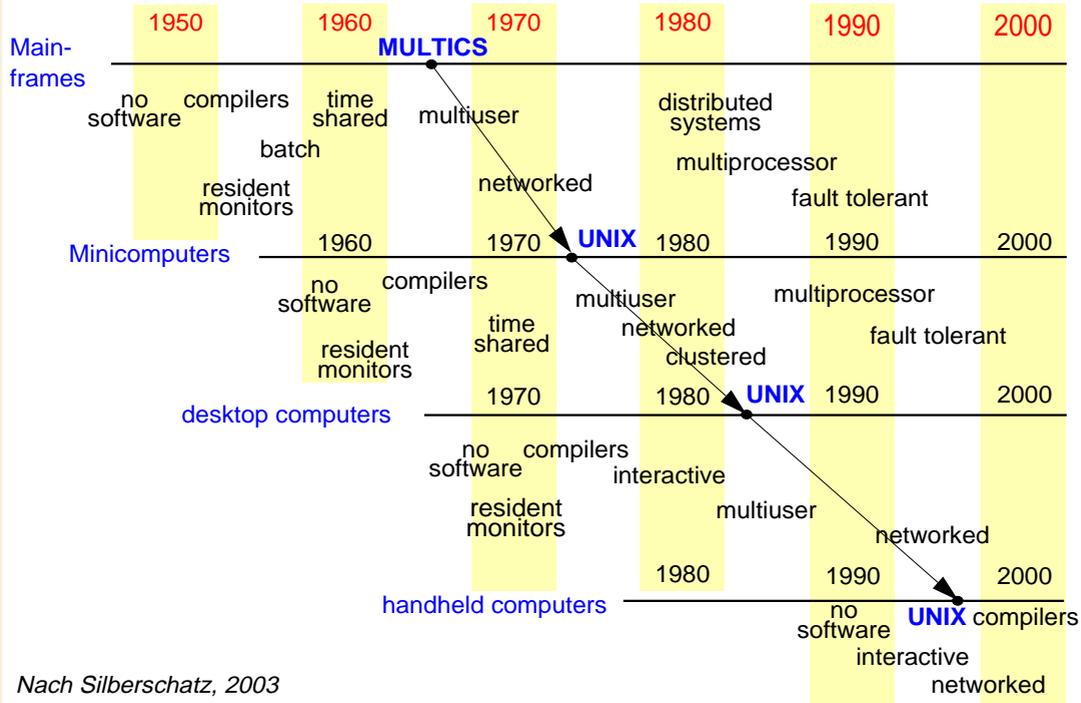
## 5.6 1985–1999



## 5.7 1995–1999



# 5.8 Migration von Konzepten



# 5.9 UNIX Entwicklung

