Vorlesung

Systemprogrammierung I

Wintersemester 1999/2000 (10320)

A Organisatorisches

- Dozent
  - Dr.-Ing. Franz J. Hauck, IMMD 4 (Lehrstuhl für Betriebssysteme)
  - hauck@informatik.uni-erlangen.de
- Vorlesung und Übungen
  - für Studierende der Fachrichtung Informatik im 3. Semester
  - für Studierende der Fachrichtung Wirtschaftsinformatik ab 5. Semester
  - für Studierende der Fachrichtung Informatik Lehramt
  - für Studierende der Fachrichtung Mathematik mit Schwerpunkt Informatik im 3. Semester (Bachelor)
  - für Studierende der Fachrichtung Computational Engineering im 3. Semester (Bachelor)
- Anrechenbare Semesterwochenstunden:
  - 4 SWS Vorlesung, 4 SWS Übungen

A.1 Inhalt

- Vorlesung
  - Grundlagen der Betriebssysteme (eingeschränkt auf Monoprozessoren)
  - Konzepte moderner Betriebssysteme
  - Beispielhafte Betrachtung von UNIX, Windows, Windows NT
- Übungen
  - Umgang mit den in der Vorlesung vorgestellten Betriebssystemkonzepten
  - Betriebssystemschnittstelle des UNIX Betriebssystems
  - Umgang mit sogenannten „System Calls“
  - Praktische Arbeiten: Ausprogrammieren von Übungsaufgaben

A.2 Vorlesung

- Termine: ~~Mo. von 11.15 bis 12.45 im H7~~
  Mo. von 10.15 bis 11.45 im H8
  Do. von 16.15 bis 17.45 im H8
- Skript
  - zwei Alternativen:
    - Folien der Vorlesung werden im WWW zur Verfügung gestellt und können selbst ausgedruckt werden
    - Folien werden kopiert und vor der Vorlesung ausgegeben; Gutscheinverkauf, Kosten 10,00 DM
  - weitergehende Informationen zum Nachlesen findet man am Besten in der angegebenen Literatur

A.2 Vorlesung (2)

- URL zur Vorlesung
  - http://www4.informatik.uni-erlangen.de/Lehre/WS99/V_SP1/
  - hier findet man Termine, Folien zum Ausdrucken und Zusatzinformationen

- Literatur
  - A. Silberschatz; P. B. Galvin: Operating Systems Concepts. 4th Edition, Addison-Wesley, 1994.
  - A. S. Tanenbaum: Modern Operating Systems, Prentice Hall, Englewood Cliffs, NJ, 1992.
  - R. W. Stevens: Advanced Programming in the UNIX Environment. Addison-Wesley, 1992.

A.2 Vorlesung (3)

- Live-Übertragung nach Nürnberg findet nicht mehr statt!

- Rückmeldungen und Fragen
  - Geben Sie mir Rückmeldungen über den Stoff. Nur so kann eine gute Vorlesung entstehen.
  - Stellen Sie Fragen!
  - Machen Sie mich auf Fehler aufmerksam!
  - Nutzen Sie außerhalb der Vorlesung die Möglichkeit elektronische Post zu versenden: hauck@informatik.uni-erlangen.de !

A.3 Übungen

- Verantwortlich für die Übung
  - Dr.-Ing. Jürgen Kleinöder
  - Dr.-Ing. Frank Bellosa
  - Dipl.-Inf. Michael Golm
  - und studentische Hilfskräfte
- Aufteilung
  - Tafelübung – 2 SWS
  - Rechnerübung – 2 SWS

A.3 Übungen (2)

- Schein
  - wird auf die Lösung von Übungsaufgaben vergeben
  - Mindestpunktzahl zum Bestehen des Scheins nötig (Hälfte der Punkte)
  - Aufgaben sollten in Gruppen zu zwei Personen gelöst werden
  - Mitglieder einer Zweiergruppe müssen an der selben Tafelgruppe teilnehmen
- Anmeldung zur Übung und Einteilung in die Übungsgruppen
  - „login: span“ an allen CIP-Workstations der Informatik
  - Login-Freischaltung wird noch bekanntgegeben
  - Benötigte Eingaben: persönliche Daten, Matrikelnummer, Termin der gewünschten Tafelübungen

A.3 Übungen (3)

- Tafelübungen
  - ◆ Termine stehen noch nicht fest
  - ◆ Im Zweifelsfalls sind die Termine aus dem Anmeldeprogramms richtig
  - ◆ Besprechung von Übungsaufgaben
  - ◆ Klärung offener Fragen
  - ◆ Vermittlung ergänzender Informationen

A.3 Übungen (4)

- Rechnerübungen
  - ◆ Termine stehen noch nicht fest
  - ◆ Lösung der Übungsaufgaben
  - ◆ Raum 01.155 ist reserviert
    (Vorrang am Rechner für Übungsteilnehmer)
  - ◆ Übungsleiter steht für Fragen zur Verfügung
- Übungsbeginn ist Montag, 8. November 1998

B Einführung

B.1 Was sind Betriebssysteme?

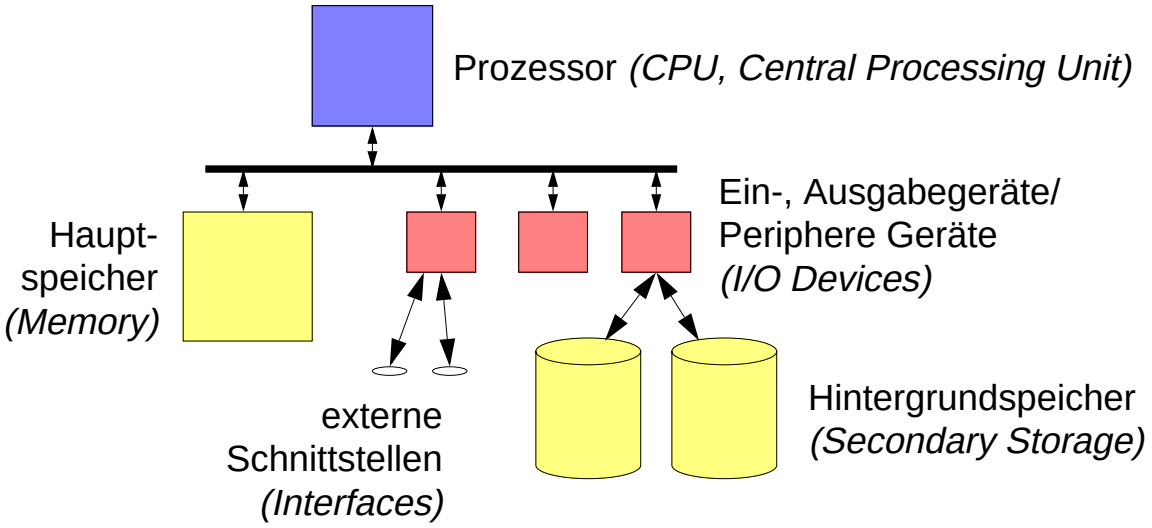
- DIN 44300
  - „...die Programme eines digitalen Rechensystems, die zusammen mit den Eigenschaften der Rechenanlage die **Basis der möglichen Betriebsarten** des digitalen Rechensystems bilden und die insbesondere die **Abwicklung von Programmen steuern und überwachen.**“
- Tanenbaum
  - „...eine Software-Schicht ..., 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].“

B.1 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

1 Verwaltung von Betriebsmitteln

- Betriebsmittel



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

2 Schnittstellen

- Betriebssystem soll Benutzervorstellungen auf die Maschinengegebenheiten abbilden
  - ◆ 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

2 Schnittstellen (2)

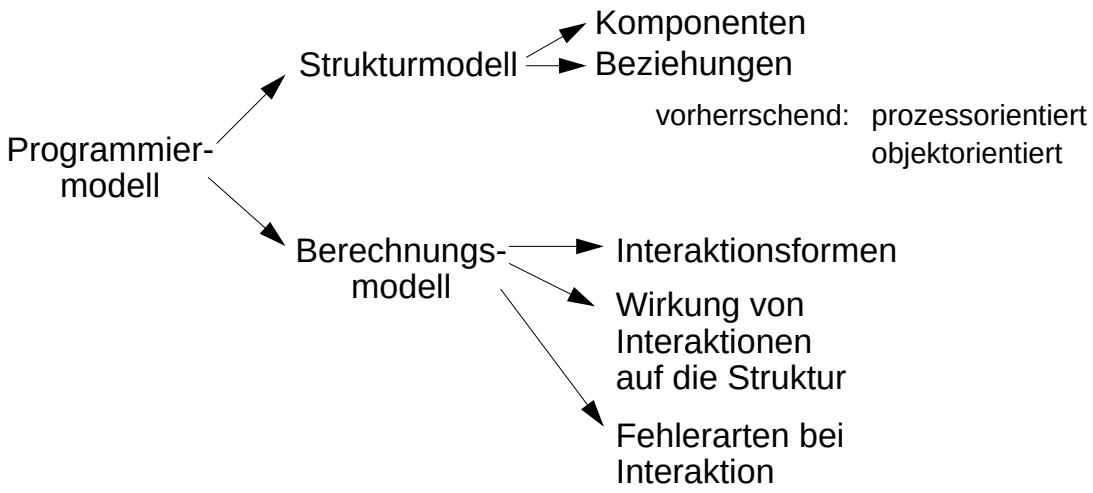
Administratoren:
Werkzeuge zur Benutzerverwaltung, langfristige Systemsteuerung

Programme:
„*Supervisor Calls* (SVC)“,
„*Application Programmer Interface* (API)“

Hardware:
Gerätetreiber

3 Programmiermodelle

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



3 Programmiermodelle (2)

- Beispiele für Strukturkomponenten
  - Dateien (Behälter zur langfristigen Speicherung von Daten)
  - Prozesse (in Ausführung befindliche Programme)
  - Klassen (Vorlagen zur Bildung von Instanzen)
  - Instanzen
  - Prozeduren
  - Sockets (Kommunikationsendpunkte, „Kommunikationssteckdosen“)
  - Pipes (Nachrichtenkanäle)
- Beispiele für Beziehungen
  - A kann B referenzieren, beauftragen, aufrufen, modifizieren
  - Pipe P verbindet A und B

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

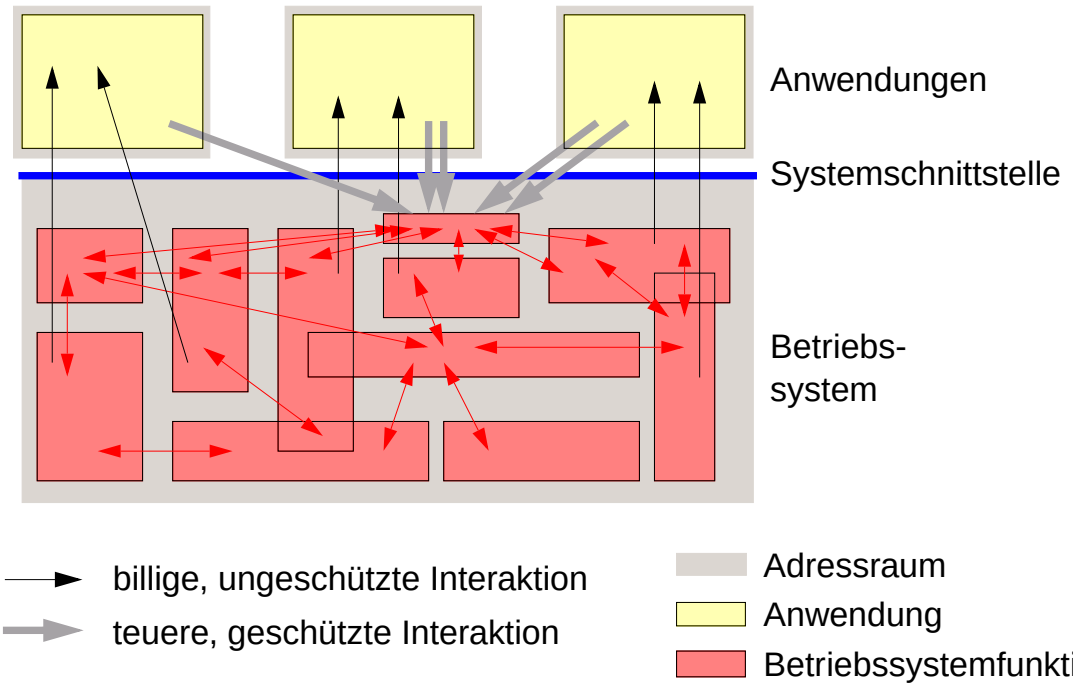
4 Ablaufmodelle

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

B.2 Betriebssystemarchitekturen

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

1 Monolithische Systeme



1 Monolithische Systeme (2)

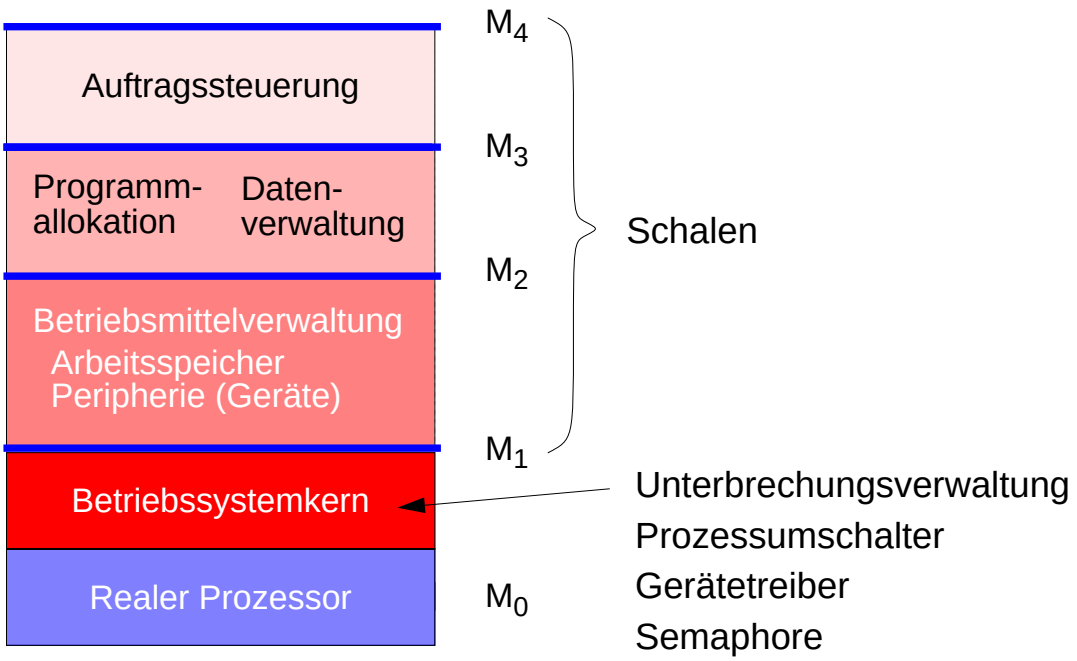
★ Vorteile

- ◆ Effiziente Kommunikation, effizienter Datenzugriff innerhalb des Kerns
- ◆ Privilegierte Befehle jederzeit ausführbar

▲ Nachteile

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

2 Geschichtete Systeme



2 Geschichtete Systeme (2)

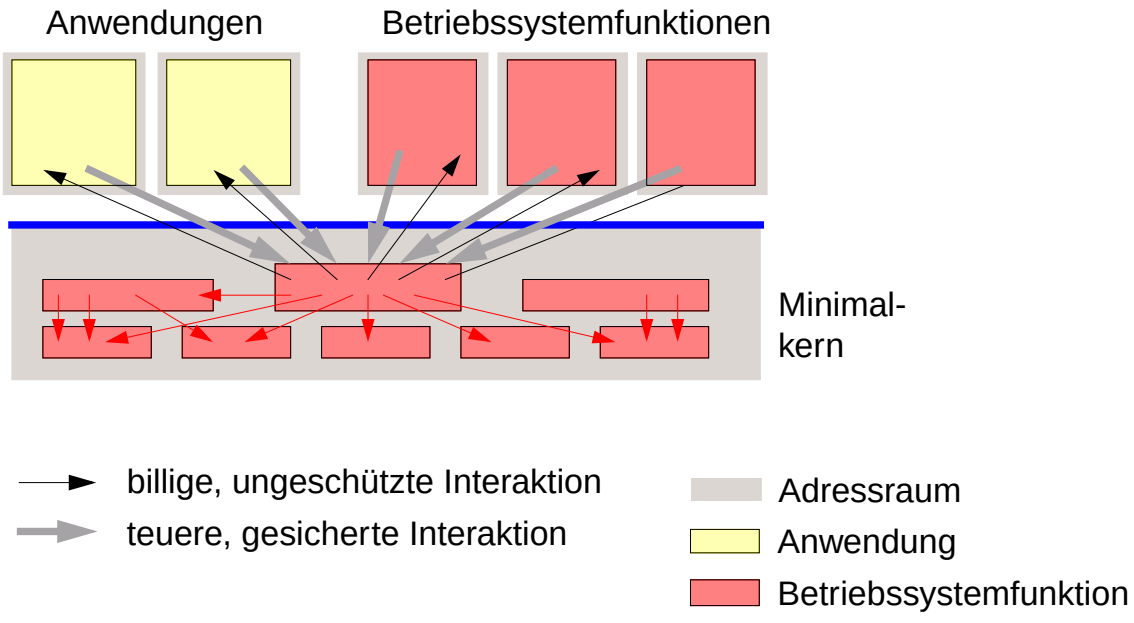
★ Vorteile

- ◆ Schutz zwischen verschiedenen BS-Teilen
- ◆ Interne Strukturierung

▲ Nachteile

- ◆ Mehrfacher Schutzraumwechsel ist teuer
- ◆ Unflexibler und nur einseitiger Schutz (von unten nach oben)

3 Minimalkerne



3 Minimalkerne (2)

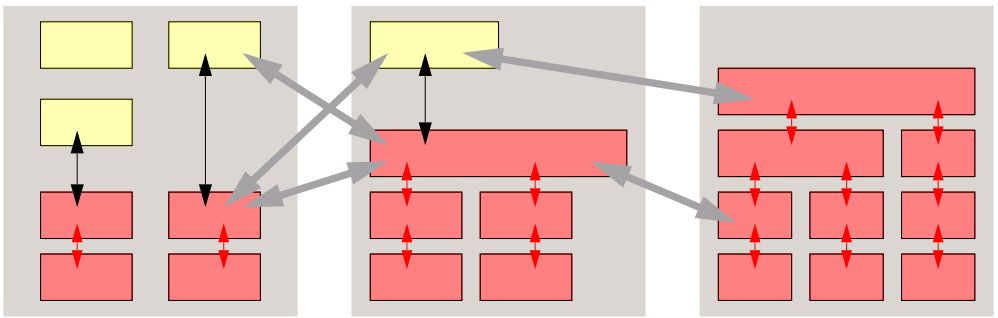
★ Vorteile

- ◆ Gute Modularisierung
- ◆ Schutz der Komponenten voreinander

▲ Nachteil

- ◆ Kommunikation zwischen Modulen ist teuer

4 Objektbasierte, offene Systeme



billige, durch Objektkapselung geschützte Interaktion

teuere, durch Adressraumgrenze geschützte Interaktion

Adressraum
Anwendungsobjekte
Betriebssystemobjekte

■ Sicherung der Modulgrenzen durch Programmiermodell und Software

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

4 Objektbasierte, offene Systeme (2)

★ Vorteile

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

▲ Nachteile

- ◆ Komplexes Sicherheitsmodell

B.3 Geschichtliche Entwicklung

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, Compiler, Binder und Lader, Programmbibliotheken

1960

2 1960–1965

1960

- Autonome periphere Geräte → Überlappung von Programmbearbeitung und Datentransport zw. Arbeitsspeicher und peripheren Geräten möglich
  - Wechselpufferbetrieb (abwechselndes Nutzen zweier Puffer)
  - 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

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
- Interaktiver Betrieb *(Interactive processing, Dialog mode)*

THE

- Mehrbenutzerbetrieb, Teilnehmersysteme *(Time sharing)*

MULTICS

- Problem: Kapselung von Prozessen und Dateien → geschützte Adressraum, Zugriffsschutz auf Dateien
- Dijkstra: Programmsysteme als Menge kooperierender Prozesse (heute *Client-Server*)

UNIX

- Problem: Prozessinteraktion bei gekapselten Prozessen → Nachrichtensysteme zur Kommunikation, gemeinsamer Speicher zur Kooperation

1970

4 1970–1975

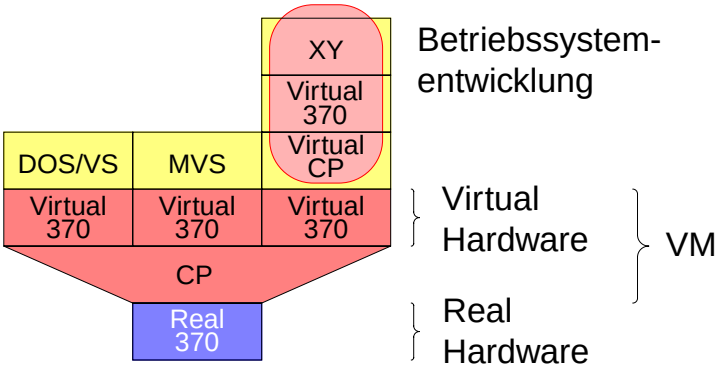
1970

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

- Komplexe Dateisysteme

5 1975–1985

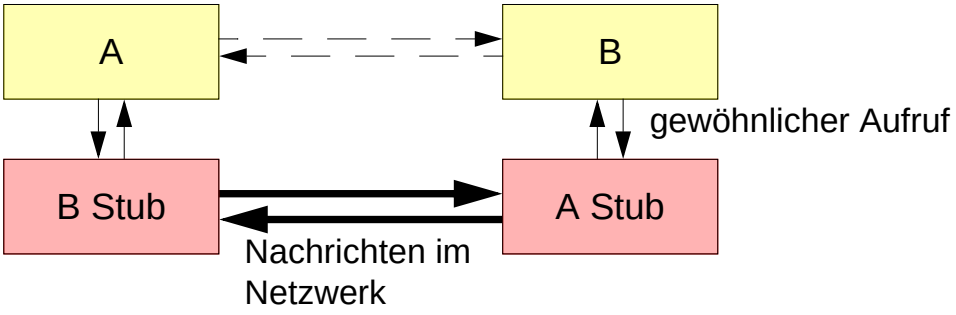
1975

- Vernetzung
- Protokolle (z.B. TCP/IP)
- Verteilte Systeme

- Newcastle Connection

LOCUS

- Fernaufruf *(Remote procedure call, RPC)*



1980

MS-DOS

NC

EDEN

1985

6 1985–1999

1985

OS/2

- Kryptographie
- Authentifizierung und Authentisierung
- Objektorientierte Systeme

Mach 3.0

- Parallele Systeme
- Mikrokerne

1990

Windows

Spring

Win NT

- Objektorientierte Mikrokerne
- Internet, Multimedia

1995

7 1995–1999

1995

SPIN

Exokernel

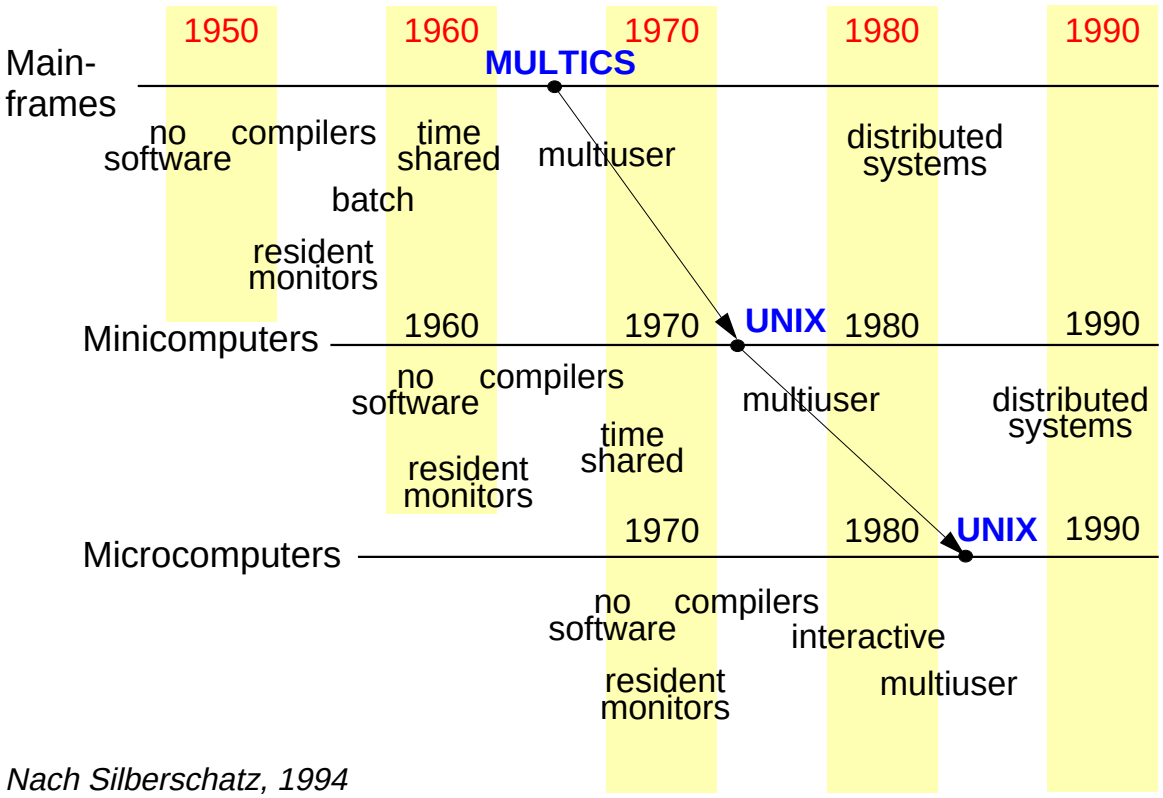
L4

Linux

- Echtzeitscheduling in Standardbetriebssystemen
- 64bit-Adressierung

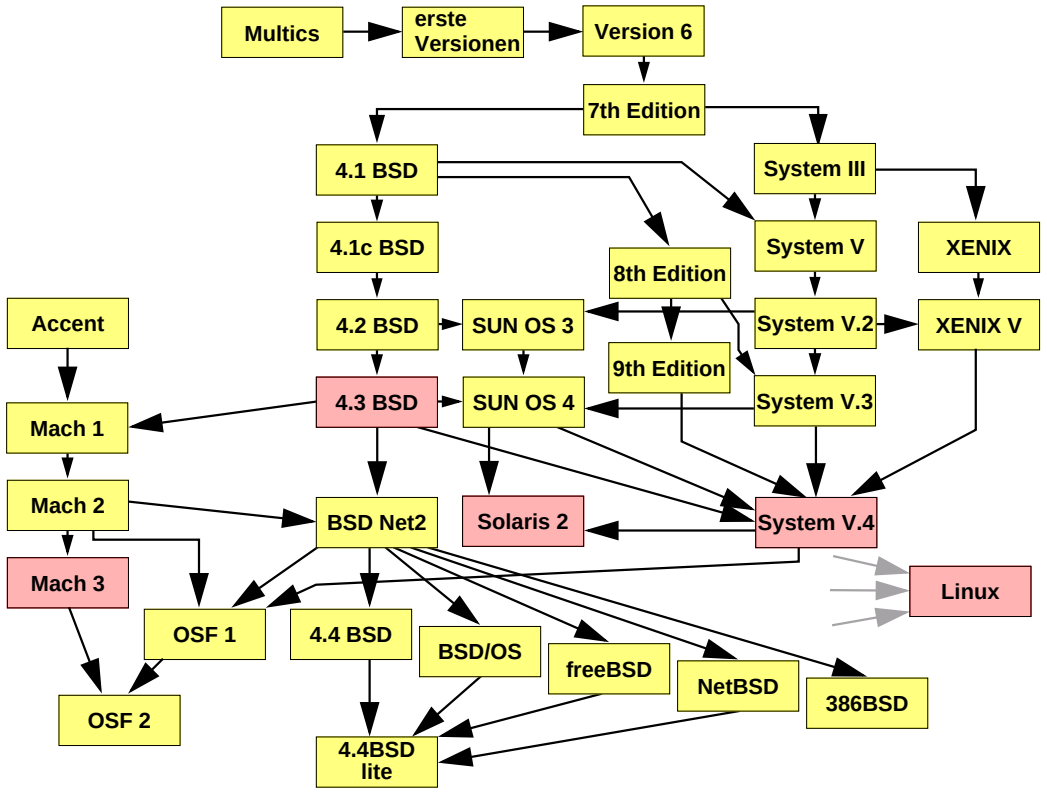
1999

8 Migration von Konzepten



Nach Silberschatz, 1994

9 UNIX Entwicklung



B.4 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

1 Phänomene der Speicherverwaltung

- Beispiel: Initialisierung von großen Matrizen
  - ◆ Variante 1:

```
#define DIM 6000

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

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

    exit(0);
}
```

1 Phänomene der Speicherverwaltung (2)

- Beispiel: Initialisierung von großen Matrizen
  - Variante 2:

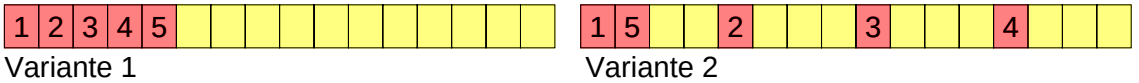
```
#define DIM 6000

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

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

    exit(0);
}
```

  - Schleifen sind vertauscht

1 Phänomene der Speicherverwaltung (3)

- Messergebnisse
  - Variante 1:
    User time= 3,69 sec; System time= 1,43 sec; Gesamtzeit= 22,03 sec
  - Variante 2:
    User time= 21,86 sec; System time= 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 Variante 2

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

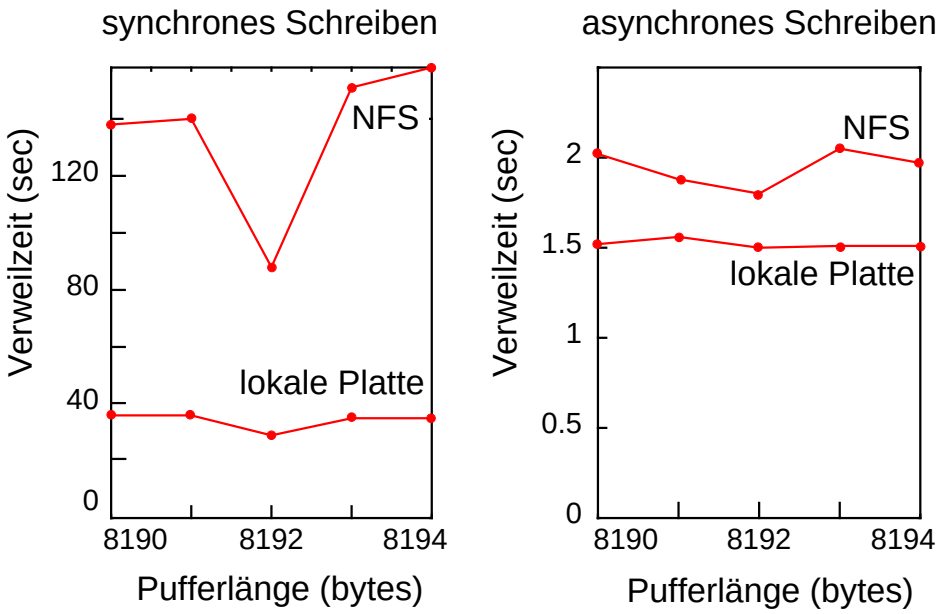
2 Phänomene des Dateisystems

- Beispiel: Sequentielles Schreiben mit unterschiedlicher Pufferlänge

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#define BUFLEN 8191

int main()
{
    static char buffer[BUFLEN];
    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, BUFLEN );

    exit(0);
}
```

2 Phänomene des Dateisystems (2)

- Messergebnisse



2 Phänomene des Dateisystems (3)

- Ursachen
  - Synchrones Schreiben erfordert sofortiges Rausschreiben der Daten auf Platte (nötig beispielsweise, wenn hohe Fehlertoleranz gefordert wird – Platte ist immer auf dem neuesten Stand)
  - 8192 ist ein Vielfaches der Blockgröße der Plattenblocks
  - kleine Abweichungen von der Blockgröße erfordern zusätzliche Blocktransfers

B.5 Überblick über die Vorlesung

★ Inhaltsübersicht