Vorlesung

Systemprogrammierung I

Wintersemester 1997/98 (10320)

A Organisatorisches

- Dozent
  - Dr.-Ing. Franz J. Hauck, IMMD IV (Lehrstuhl für Betriebssysteme)
  - hauck@informatik.uni-erlangen.de

A.1 Vorlesung

- „Systemprogrammierung I“
  - Grundlagen der Betriebssysteme (eingeschränkt auf Monoprozessoren)
  - für Studierende der Fachrichtung Informatik im 3. Semester und einige weitere
- Termine: Mi. und Do. von 16.15 bis 17.45 im H4 (4 SWS)

A.1 Vorlesung (2)

- 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 in den ersten Übungen; Kosten 10,00 DM
  - weitergehende Informationen zum Nachlesen findet man am besten in der angegebenen Literatur
- URL zur Vorlesung
  - http://www4.informatik.uni-erlangen.de/Lehre/WS97/V_SP1/
  - hier findet man Termine, Folien zum Ausdrucken und evtl. Zusatzinformationen

A.1 Vorlesung (3)

- 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. Hieronymus: UNIX - Systemarchitektur und Programmierung. Vieweg, 1993.

A.1 Vorlesung (4)

- Live-Übertragung nach Nürnberg
- Rückmeldungen und Fragen
  - Gute Vorlesung kann nicht entstehen, wenn nur Informationen vom Dozent zu den Hörern fließen!
  - Machen Sie mich auf Fehler aufmerksam!
  - Stellen Sie Fragen!
  - Nutzen Sie außerhalb der Vorlesung die Möglichkeit elektronische Post zu versenden: hauck@informatik.uni-erlangen.de !

A.2 Übungen

- Verantwortlich für die Übung
  - Dr.-Ing. Jürgen Kleinöder
  - Dipl.-Inf. Martin Steckermeier
  - Dipl.-Inf. Michael Golm
  - und studentische Hilfskräfte
- Inhalt
  - Betriebssystemschnittstelle des UNIX Betriebssystems
  - Umgang mit sogenannten „System calls“
  - Umgang mit den in der Vorlesung vorgestellten Betriebssystemkonzepten

A.2 Übungen (2)

- Schein
  - wird auf die Lösung von Übungsaufgaben vergeben
  - Mindestpunktzahl jeweils für die Aufgaben vor und nach Weihnachten erforderlich
  - Aufgaben sollten in Gruppen zu zwei Personen gelöst werden
  - Mitglieder einer Zweiergruppe müssen an der selben Tafelgruppe teilnehmen
- Aufteilung
  - Tafelübung – 2 SWS
  - Rechnerübung – 2 SWS
- Anmeldung zur Übung und Einteilung in die Übungsgruppen
  - „login: span“ an allen CIP-Workstations der Informatik

A.2 Übungen (3)

- Tafelübungen
  - Mo. 10:00 - 12:00 Raum 00.151
  - Mo. 12:00 - 14:00 Raum 2.038
  - Mo. 14:00 - 16:00 Raum 0.031
  - Mo. 16:00 - 18:00 Raum 00.151
  - Di. 10:00 - 12:00 Raum 0.031
  - Di. 12:00 - 14:00 Raum 0.031
  - Besprechung von Übungsaufgaben
  - Klärung offener Fragen
  - Vermittlung ergänzender Informationen

A.2 Übungen (4)

- Rechnerübungen
  - ~~Mo. 10:00 - 12:00 Raum 01.155~~ Mi. 08:00 - 10:00 Raum 01.155
  - Mo. 16:00 - 18:00 Raum 01.155
  - Di. 16:00 - 18:00 Raum 01.155

- Lösung der Übungsaufgaben
- Raum 01.155 ist reserviert (Vorrang am Rechner für Übungsteilnehmer)
- Übungsleiter steht für Fragen zur Verfügung

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.**“

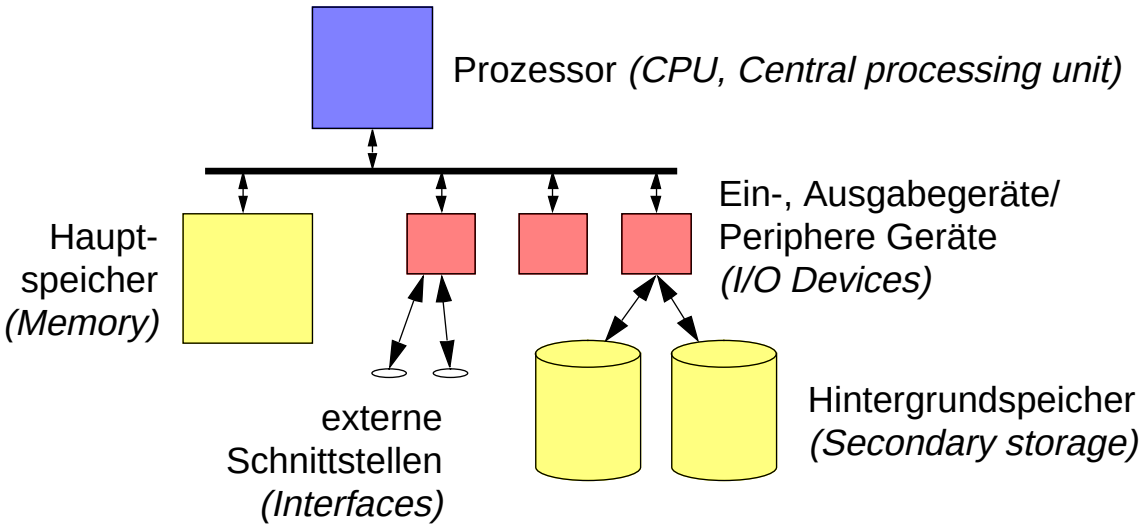
- Tanenbaum
  - „...eine Software-Schicht ..., die alle Teile des Systems verwaltet und dem Benutzer eine Schnittstelle oder eine *virtuelle Maschine* anbietet, die einfacher zu verstehen und zu programmieren ist [als die nackte Hardware].

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
- Ermöglichung 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

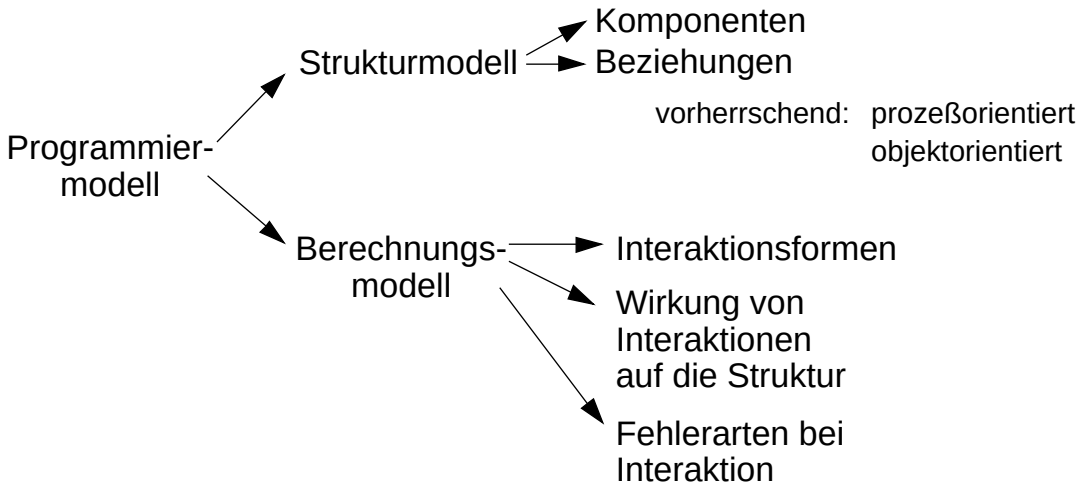
Administratoren: Werkzeuge zur Benutzerverwaltung, langfristige Systemsteuerung

Programme: „*Supervisor calls* (SVC)“

Hardware: Gerätetreiber

3 Programmiermodelle

- Realisierungen von Programmiermodellen
  - 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
  - ◆ Instantiierung
- Fehlerarten bei Interaktion
  - ◆ Verlust, Wiederholung oder Verspätung von Nachrichten
  - ◆ Abbruch aufgerufener Methoden

4 Ablaufmodelle

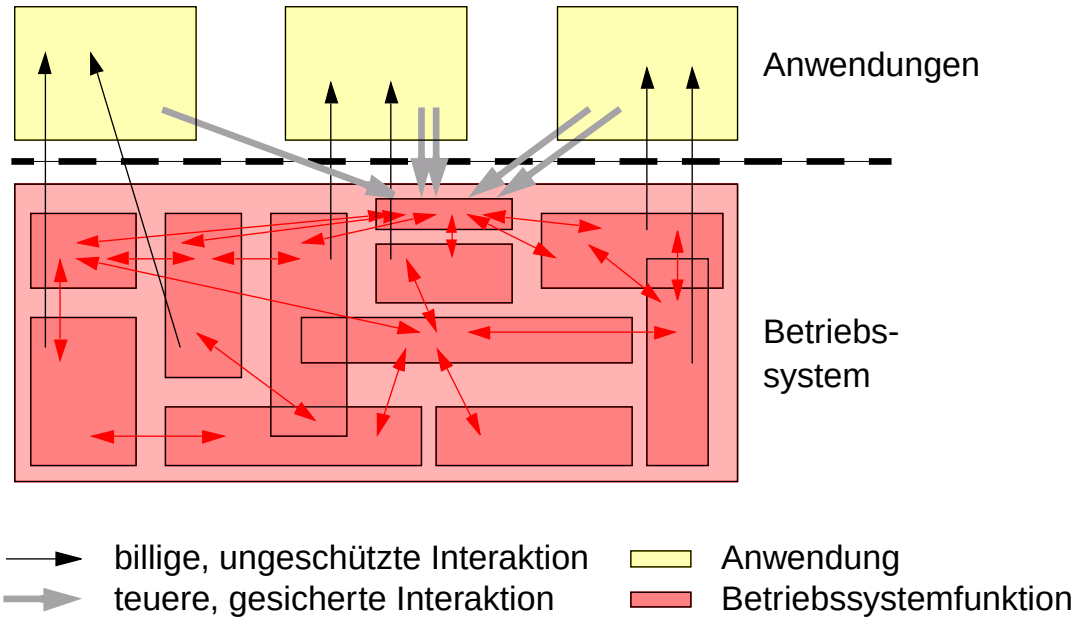
- Realisierungen von Ablaufumgebungen
- Bereitstellung von Hilfsmitteln zur Bearbeitung von Benutzerprogrammen und zur Steuerung ihrer Abläufe.
  - ◆ Laden und Starten von Programmen
  - ◆ Überwachung des Programmablaufs
  - ◆ Beenden und Eliminieren von Programmen
  - ◆ Abrechnung *(Accounting)*

5 Implementierung

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

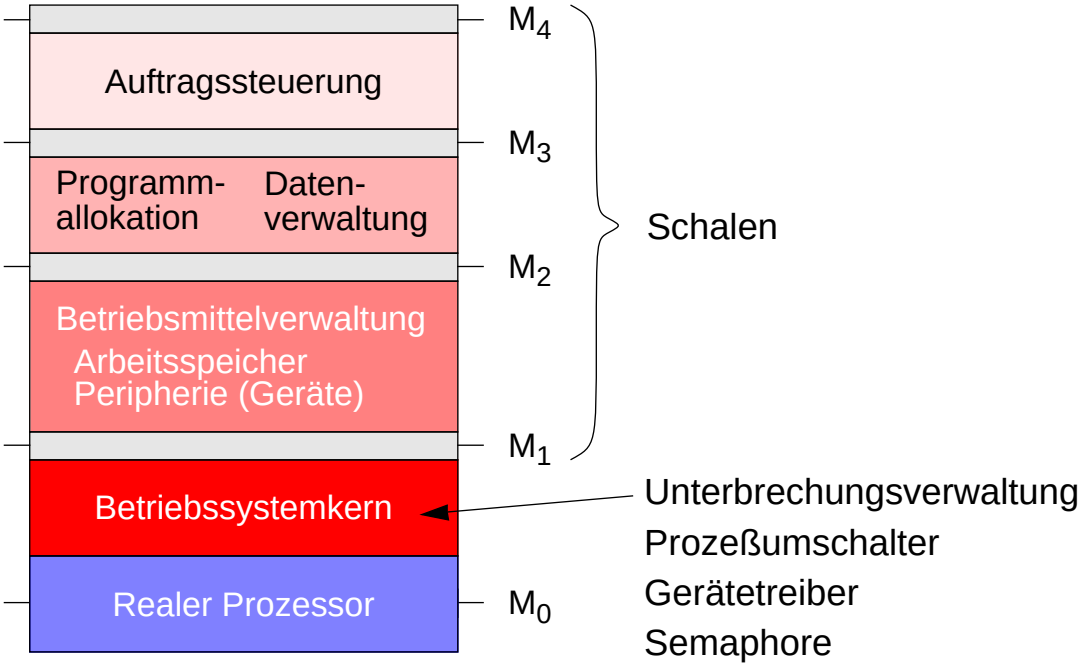
5 Implementierung (2)

- Monolithische Systeme



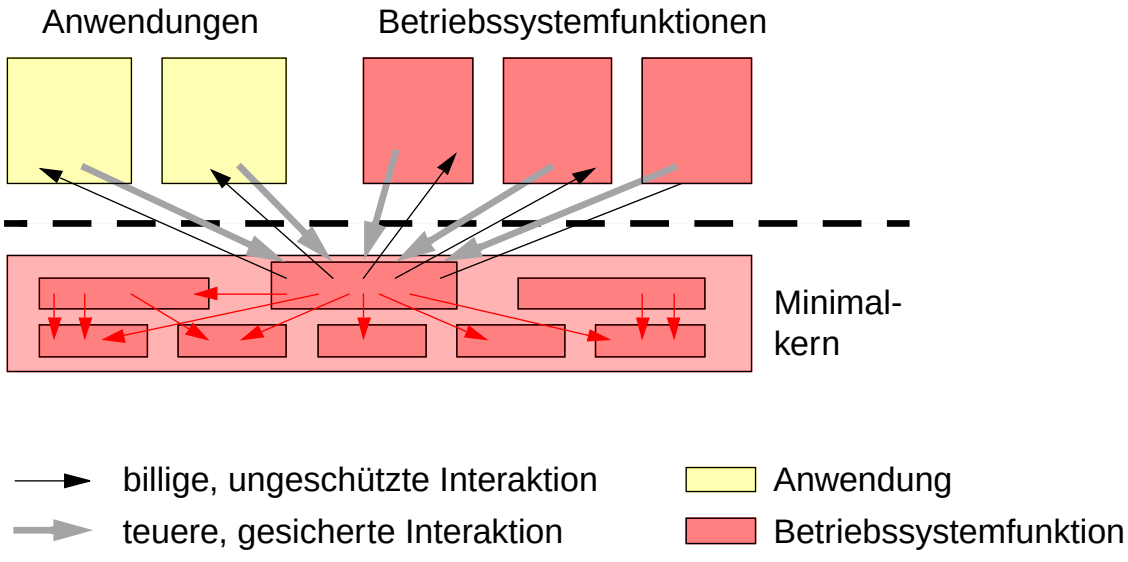
5 Implementierung (3)

- Geschichtete Systeme



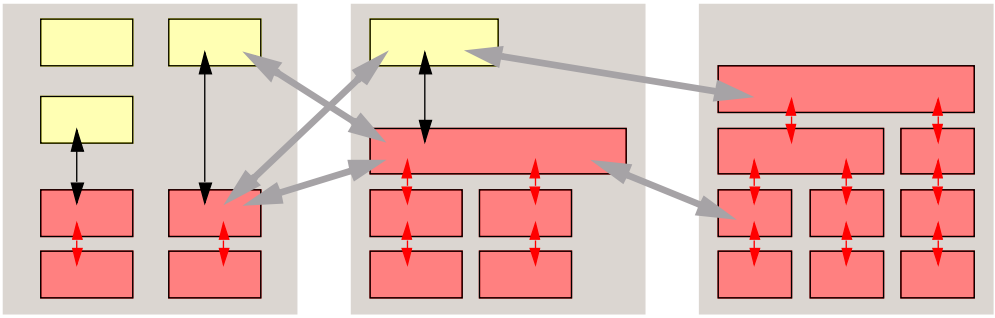
5 Implementierung (4)

- Minimalkerne



5 Implementierung (5)

- Objektbasierte, offene Systeme



billige, durch Objektkapselung geschützte Interaktion

teuere, durch Adreßraumgrenze geschützte Interaktion

Adreßraum

Anwendungsobjekte

Betriebssystemobjekte

B.2 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

- Umsetzung von Programmadressen in Speicherorte zur Laufzeit: Segmentierung, Seitenadressierung *(Paging)*

OS/360

- 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 Adreßräume, 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

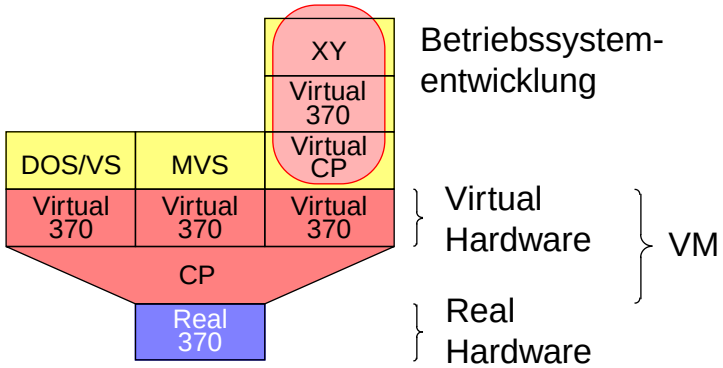
1970

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

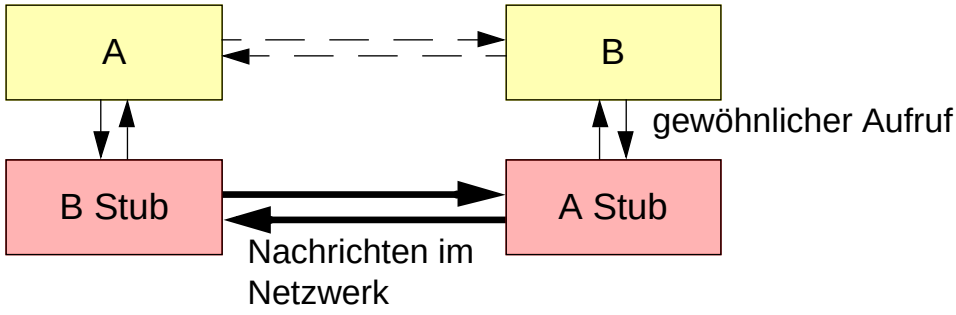
1975

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

- Newcastle Connection

LOCUS

- Fernaufruf *(Remote procedure call, RPC)*



1980

MS-DOS

NC

EDEN

1985

6 1985–1997

1985

OS/2

- Kryptographie
- Authentifizierung und Authentisierung
- Objektorientierte Systeme

MACH 3.0

- Parallele Systeme
- Mikrokerne

1990

WINDOWS

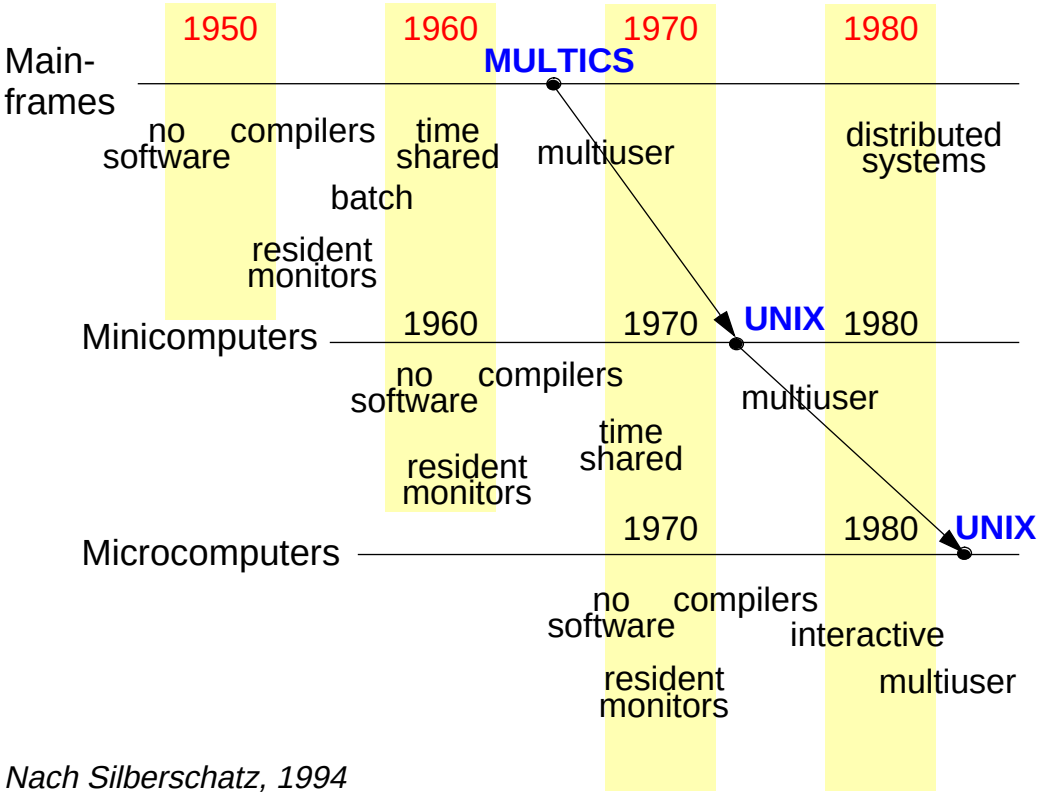
SPRING

WIN NT

- Objektorientierte Mikrokerne
- Internet, Multimedia

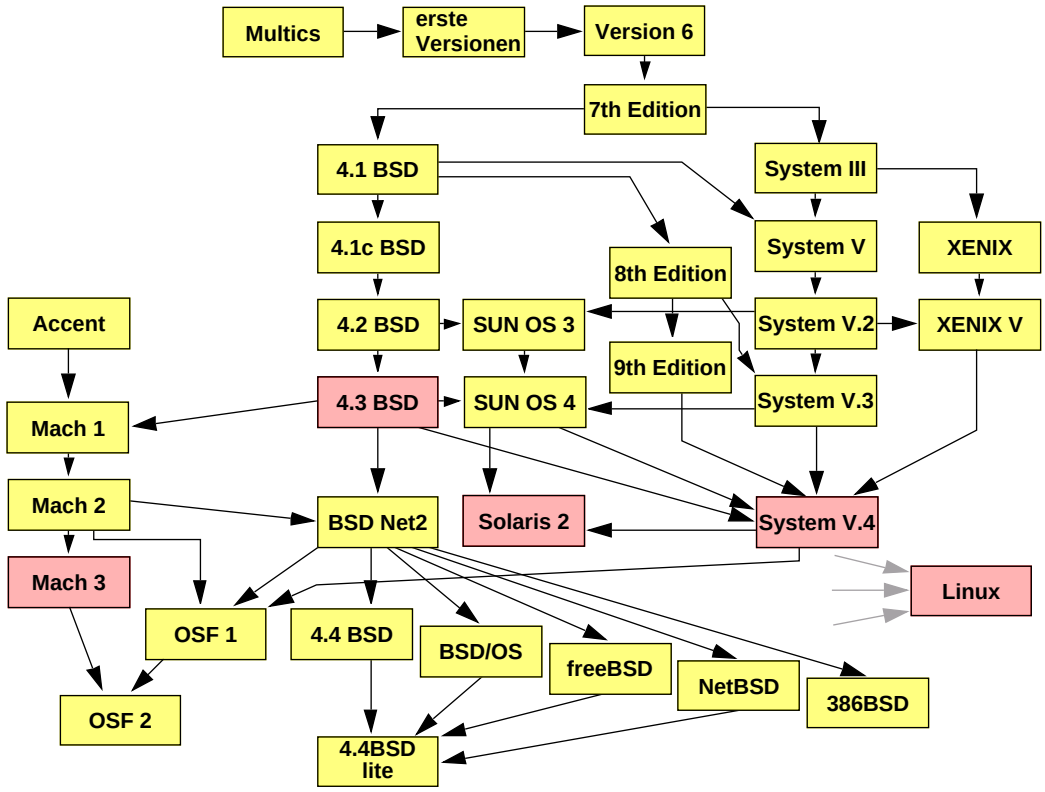
1997

7 Migration von Konzepten



Nach Silberschatz, 1994

8 UNIX Entwicklung



B.3 Warum Systemprogrammierung I?

- Rasche Einarbeitung in spezielle Systeme
  - MVS, BS2000, VM, Solaris, Unix, Windows NT, Windows 95, MS/DOS
- Strukturierung komplexer Programmsysteme
  - Unterteilung in interagierende Komponenten
- Konzeption und Implementierung spezialisierter 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

```
#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;

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

    exit(0);
}
```

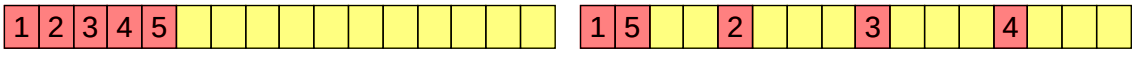
Variante 1

Variante 2

1 Phänomene der Speicherverwaltung (2)

- 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

  - Seitenadressierung: Variante 2 hat weniger Lokalität
  - Virtueller Speicher:
    bei Variante 2 muß viel mehr Speicher ein- und ausgelagert werden

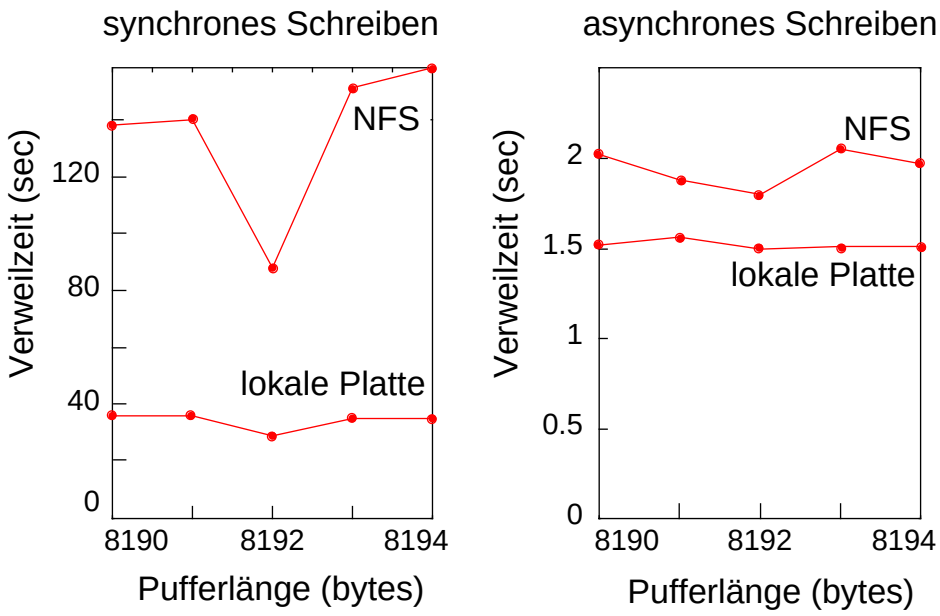
2 Phänomene des Dateisystems

- 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)

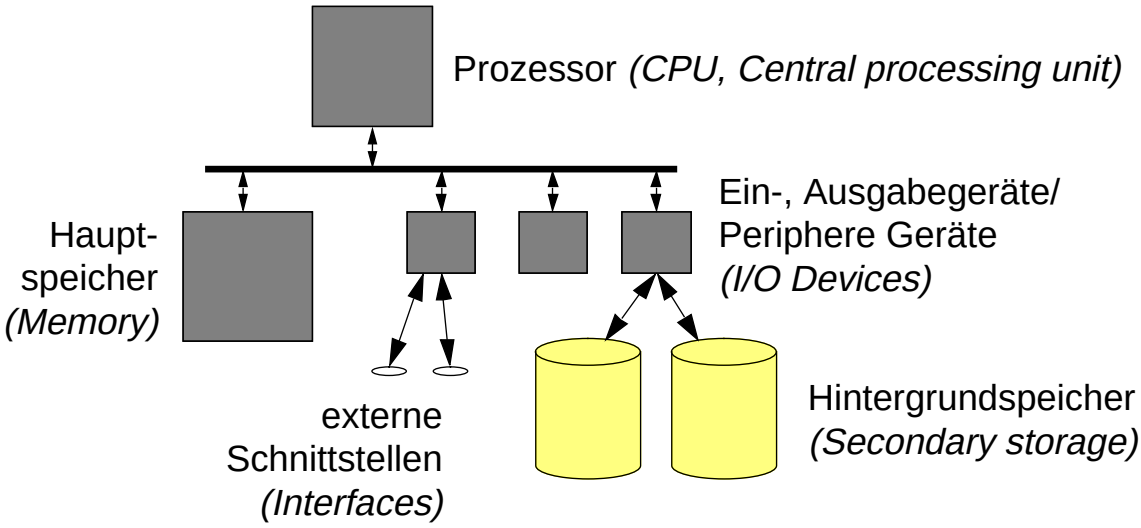
- Ursachen
  - Synchrones Schreiben erfordert sofortiges Rausschreiben der Daten auf Platte (nötig beispielsweise, wenn hohe Fehlertoleranz gefordert wird – Platte ist immer auf dem neuesten Stand)
  - 8192 ist ein Vielfaches der Blockgröße der Plattenblocks
  - kleine Abweichungen von der Blockgröße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

B.4 Überblick über die Vorlesung

- Inhaltsübersicht

C Dateisysteme

- Einordnung

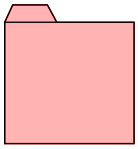


C Dateisysteme (2)

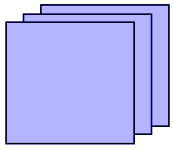
- Dateisysteme speichern Daten und Programme persistent in Dateien
  - ◆ Betriebssystemabstraktion zur Nutzung von Hintergrundspeichern (z.B. Platten, CD-ROM, Floppy Disk, Bandlaufwerke)
    - Benutzer muß sich nicht um die Ansteuerungen verschiedener Speichermedien kümmern
    - einheitliche Sicht auf den Sekundärspeicher
- Dateisysteme bestehen aus
  - ◆ Dateien *(Files)*
  - ◆ Katalogen / Verzeichnissen *(Directories)*
  - ◆ Partitionen *(Partitions)*

C Dateisysteme (3)

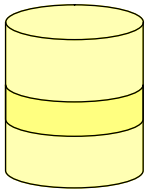
- Datei
  - ◆ speichert Daten oder Programme
- Katalog
  - ◆ erlaubt Benennung der Dateien
  - ◆ enthält Zusatzinformationen zu Dateien
- Partitionen
  - ◆ eine Menge von Katalogen und deren Dateien
  - ◆ Sie dienen zum physischen oder logischen Trennen von Dateimengen.



Katalog



Dateien



Partition

C.1 Dateien

- Kleinste Einheit, in der etwas auf den Hintergrundspeicher geschrieben werden kann.

1 Dateiattribute

- *Name* — Symbolischer Name, vom Benutzer les- und interpretierbar
  - ◆ z.B. **`AUTOEXEC.BAT`**
- *Typ* — Für Dateisysteme, die verschiedene Dateitypen unterscheiden
  - ◆ z.B. sequentielle Datei, satzorientierte Datei
- *Ortsinformation* — Wo werden die Daten physisch gespeichert?
  - ◆ Gerätenummer, Nummern der Plattenblocks

1 Dateiattribute (2)

- *Größe* — Länge der Datei in Größeneinheiten (z.B. Bytes, Blocks, Sätze)
  - ◆ steht in engem Zusammenhang mit der Ortsinformation
  - ◆ wird zum Prüfen der Dateigrenzen z.B. beim Lesen benötigt
- *Zeitstempel* — Zeit und Datum der Erstellung, letzten Modifikation etc.
  - ◆ unterstützt Backup, automatische Dateierzeugung, Benutzerüberwachung etc.
- *Rechte* — Zugriffsrechte bestimmen, wer lesen, schreiben etc. kann
  - ◆ z.B. nur für den Eigentümer schreibbar für alle anderen nur lesbar
- *Eigentümer* — Identifikation des Eigentümers
  - ◆ Eventuell eng mit den Rechten verknüpft
  - ◆ Zuordnung beim Accounting (Abrechnung des Plattenplatzes)

2 Operationen auf Dateien

- Erzeugen *(Create)*
  - ◆ Nötiger Speicherplatz wird angefordert
  - ◆ Katalogeintrag wird erstellt
  - ◆ Initiale Attribute werden gespeichert
- Schreiben *(Write)*
  - ◆ Identifikation der Datei
  - ◆ Daten werden auf Platte transferiert
  - ◆ Eventuelle Anpassung der Attribute, z.B. Länge
- Lesen *(Read)*
  - ◆ Identifikation der Datei
  - ◆ Daten werden von Platte gelesen