1 Einführung

1.1 Was sind Betriebssysteme?

1.1.1 Betriebsmittelverwalter

1.1.2 Ausführungsmodelle

1.2 Wie funktionieren Betriebssysteme?

1.3 Warum sollte man das wissen?

1 Einführung

1.1 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**"

- Nach Habermann ist

  "... ein Betriebssystem eine Menge von Programmen, die die Ausführung von Benutzerprogrammen und die Benutzung von Betriebsmitteln steuern."

- Nach Brinch Hansen besteht

  "... der Zweck eines Betriebssystems in der Verteilung von Betriebsmitteln auf sich bewerbende Benutzer."

- Weitergehende Auffassung
  - Betriebsmittelverwaltung

  und

  - Bereitstellung von Grundkonzepten zur statischen und dynamischen Strukturierung von Programmsystemen

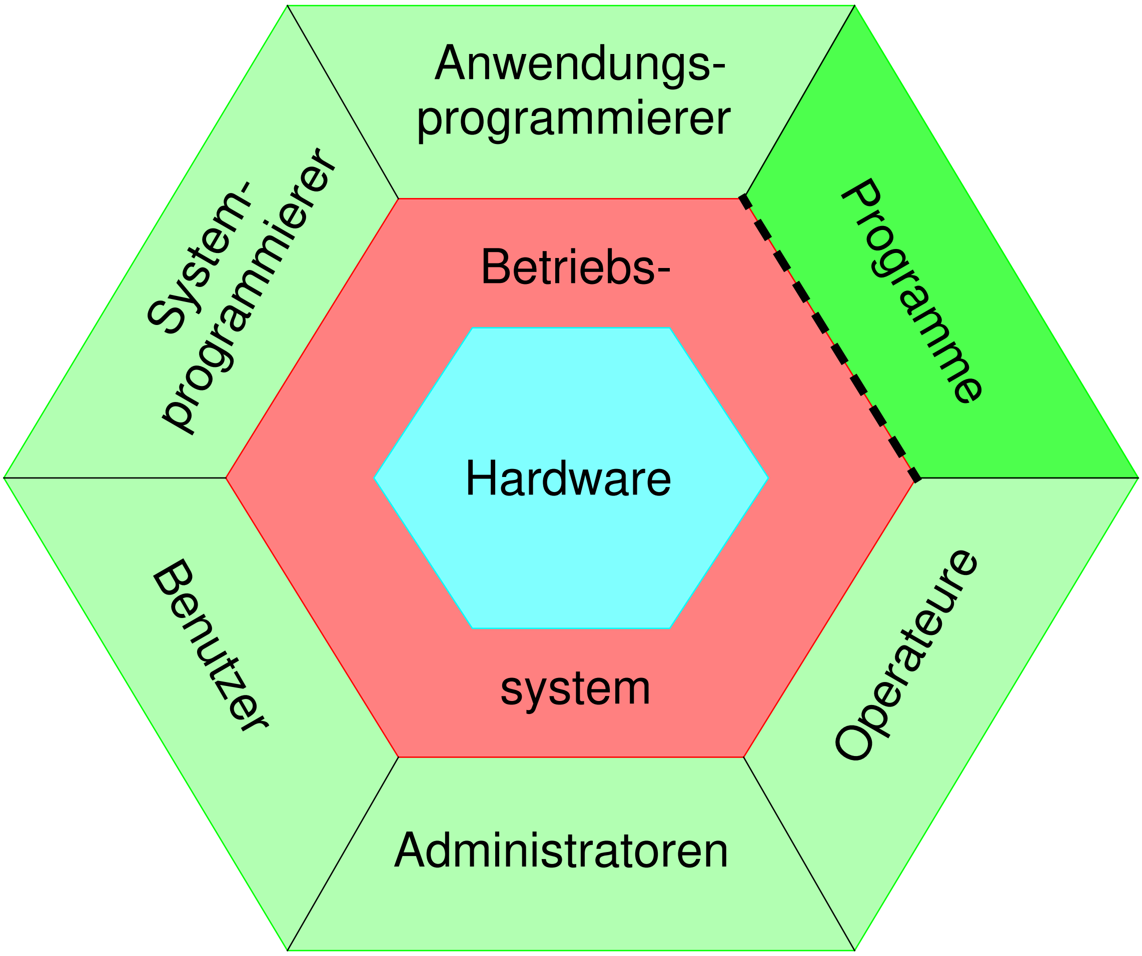
1.1.1 Betriebsmittelverwalter (*resource allocator*)

Die Betriebsmittel, die dem Rechner zur Verfügung stehen, müssen koordiniert und den jeweiligen Anforderungen entsprechend an mehrere, nacheinander oder gleichzeitig arbeitende Benutzer aufgeteilt werden.

- Beispiele:
  - Prozessor (*central processing unit, cpu*)
  - Arbeitsspeicher, Primärspeicher (*memory*)
  - Hintergrundspeicher, Sekundärspeicher (*secondary storage*)
  - Periphere Geräte und Anschlüsse (*peripheral devices, I/O devices*)
- Resultierende Aufgaben
  - Multiplexen der Betriebsmittel für mehrere Benutzer
  - Schaffung von Schutzumgebungen
- Ermöglichung einer koordinierten gemeinsamen Nutzung von Betriebsmitteln, klassifizierbar in
  - aktive, zeitlich aufteilbare (Prozessoren)
  - passive, nur exklusiv benutzbare (periphere Geräte wie Drucker u. ä.)
  - passive, räumlich aufteilbare (Speicher wie Plattenspeicher u. ä.)
- Unterstützung bei der Fehlererholung.

- Umfang: Einige zehntausend bis mehrere Millionen Befehlszeilen

- Strukturelle Einordnung



- Nähere Erläuterung anhand der historische Entwicklung (Die zeitliche Einordnung ist nur als ungefähre Zeitangabe für den Beginn der Verbreitung der Konzepte zu sehen)

1950

- Einströmige Stapelsysteme (*single-stream batch processing systems*): Aufträge werden zusammen mit allen benötigten Daten dem Rechensystem übergeben und sequentiell bearbeitet.

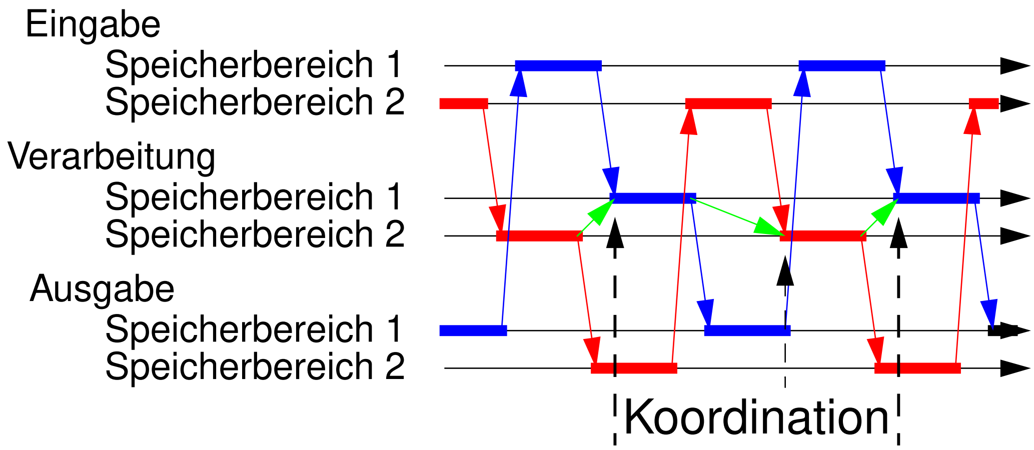
- Gesteuert durch einen Auftragsabwickler (*resident monitor, job monitor*).

  Hilfsmittel: Assembler, Compiler, Binder und Lader, Programmbibliotheken.

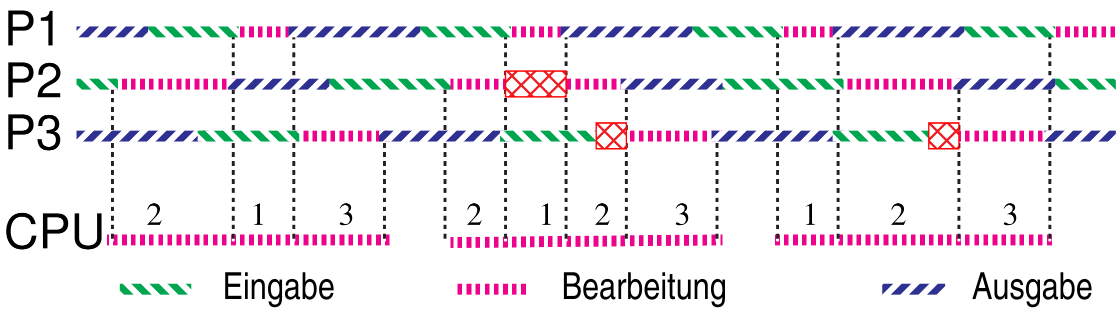
1960

1960

- Autonome periphere Geräte erlauben im Prinzip Überlappung von Programmbearbeitung mit Datentransport zwischen Arbeitsspeicher und peripheren Geräten. Nutzung durch
  - Wechselpufferbetrieb



  - Mehrprozeßbetrieb (*multiprogramming*).



  - SPOOLING (*simultaneous peripheral operation on-line*)

- Problem: gleichzeitige Speicherung mehrerer Programme

  Antwort: zeitweise Zwischenlagerung von Programmen auf Sekundärspeicher (*swapping*)

- Neues Problem: Verlagerbarkeit von Programmen während des Ablaufs (*relocation*)

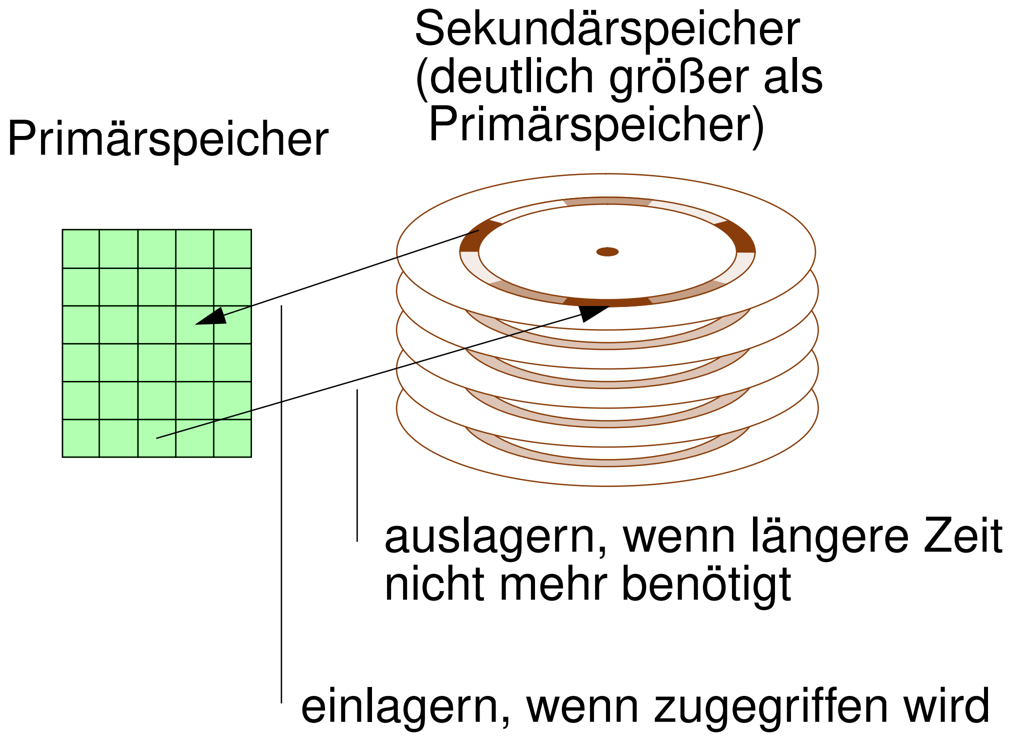
- Aufkommen der Echtzeitdatenverarbeitung (*real time processing*), d. h. enge Anbindbarkeit von Ein- und Ausgaben an die physikalische Zeit

1965

1965

OS/360

- Umsetzung von Programmadressen in Speicherorte zur Laufzeit: Seitenadressierung (*paging*), Segmentierung
- Virtueller Adreßraum: Seitentausch (*paging*)



- Interaktiver Betrieb (*interactive processing, dialog mode*)
- Mehrbenutzerbetrieb, Teilnehmersysteme (*time sharing*)

THE

- Problem: Kapselung von Prozessen und Dateien, geschützte Adreßräume, Datei-Zugriffsschutz

MULTICS

- Dijkstra: Programmsysteme als Menge kooperierender Prozesse (heute: *client* - *server*)
- Problem: Prozeßinteraktion bei gekapselten Prozessen

UNIX

- Lösung: Nachrichtensysteme zur Kommunikation, gemeinsamer Speicher zur Kooperation

1970

1970

VM

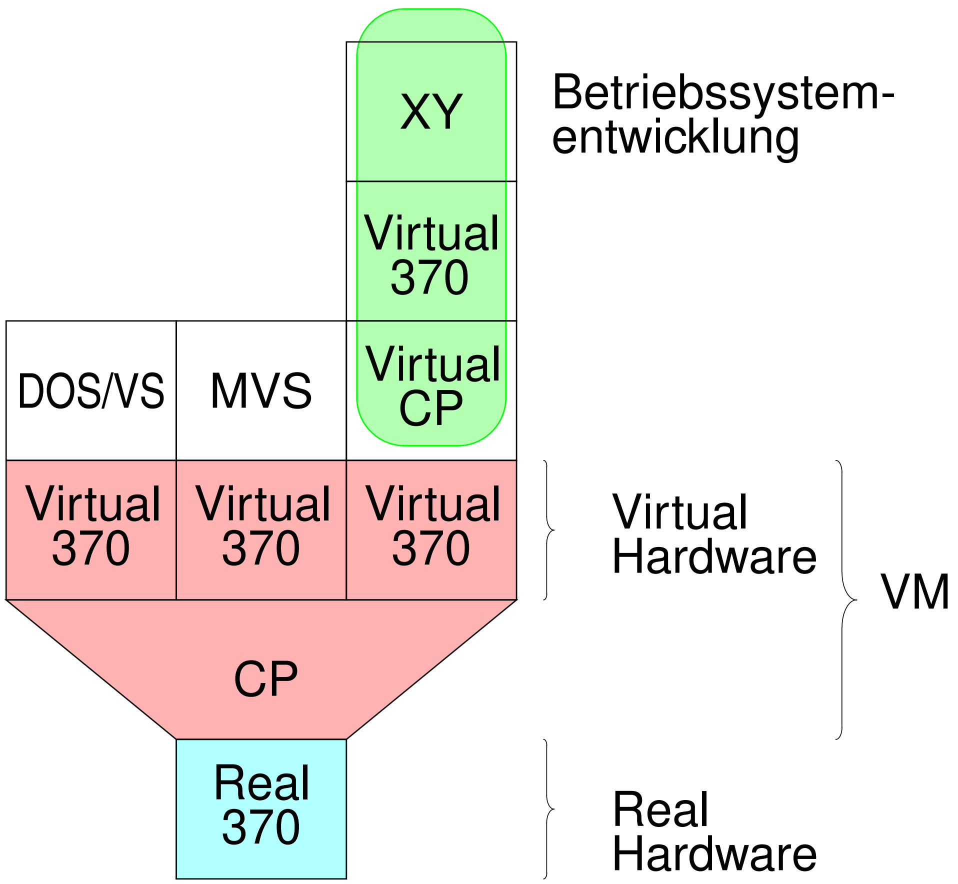
HYDRA

MVS

- Modularisierung: Daten gekapselt, Manipulation nur durch Funktionen (Parnas).

  In heutiger Terminologie Objekte oder Instanzen, die durch Methoden beeinflußbar sind.

- Virtuelle Maschinen
  Koexistenz verschiedener Betriebssysteme im gleichen Rechner.



- Symmetrische Multiprozessoren: Hydra
  - Kontrolle des Zugangs zu Instanzen durch Capabilities.
  - Trennung von Strategie und Mechanismus

1975

- Komplexe Dateisysteme

1975

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

- NewCastle Connection

- Fernaufruf (*remote procedure call, RPC*)

LOCUS

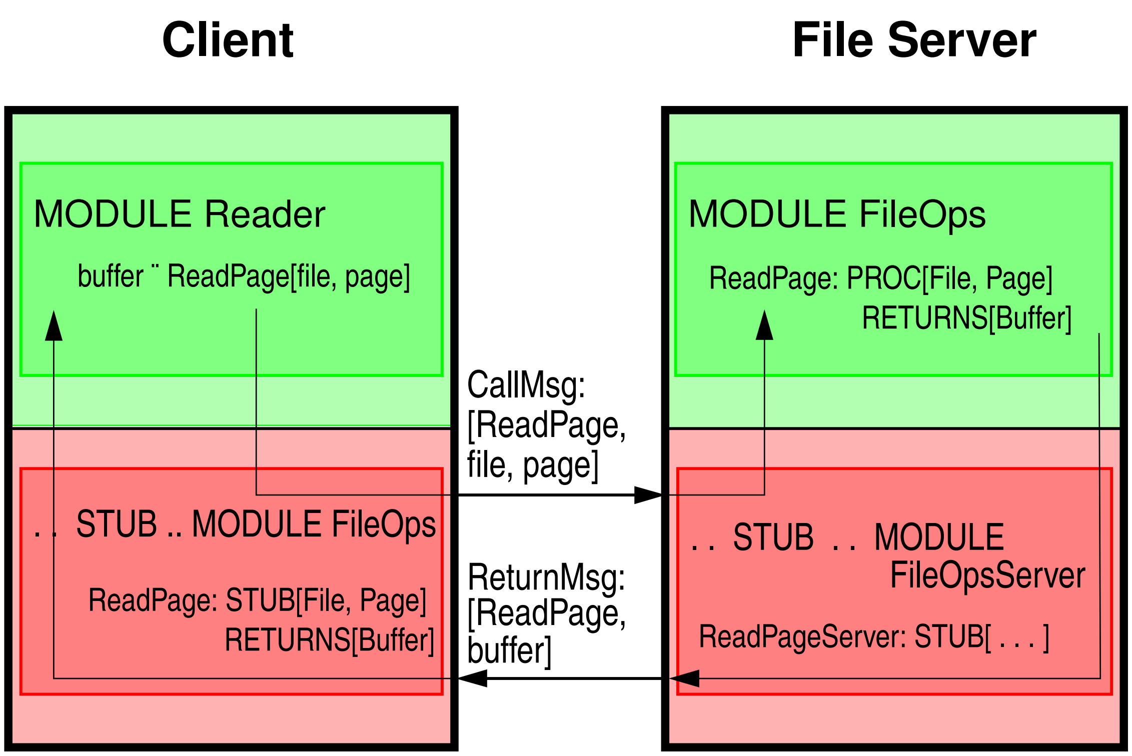
1980

MS-DOS

NC

EDEN

1985



- Kryptographie

1985

OS/2

- Authentifizierung und Authentisierung
- Objektorientierte Systeme

MACH 3.0

- Parallele Systeme

1990

WINDOWS 3.0

- Mikrokerne
- objektorientierte Mikrokerne: Spring
- Internet, Multimedia

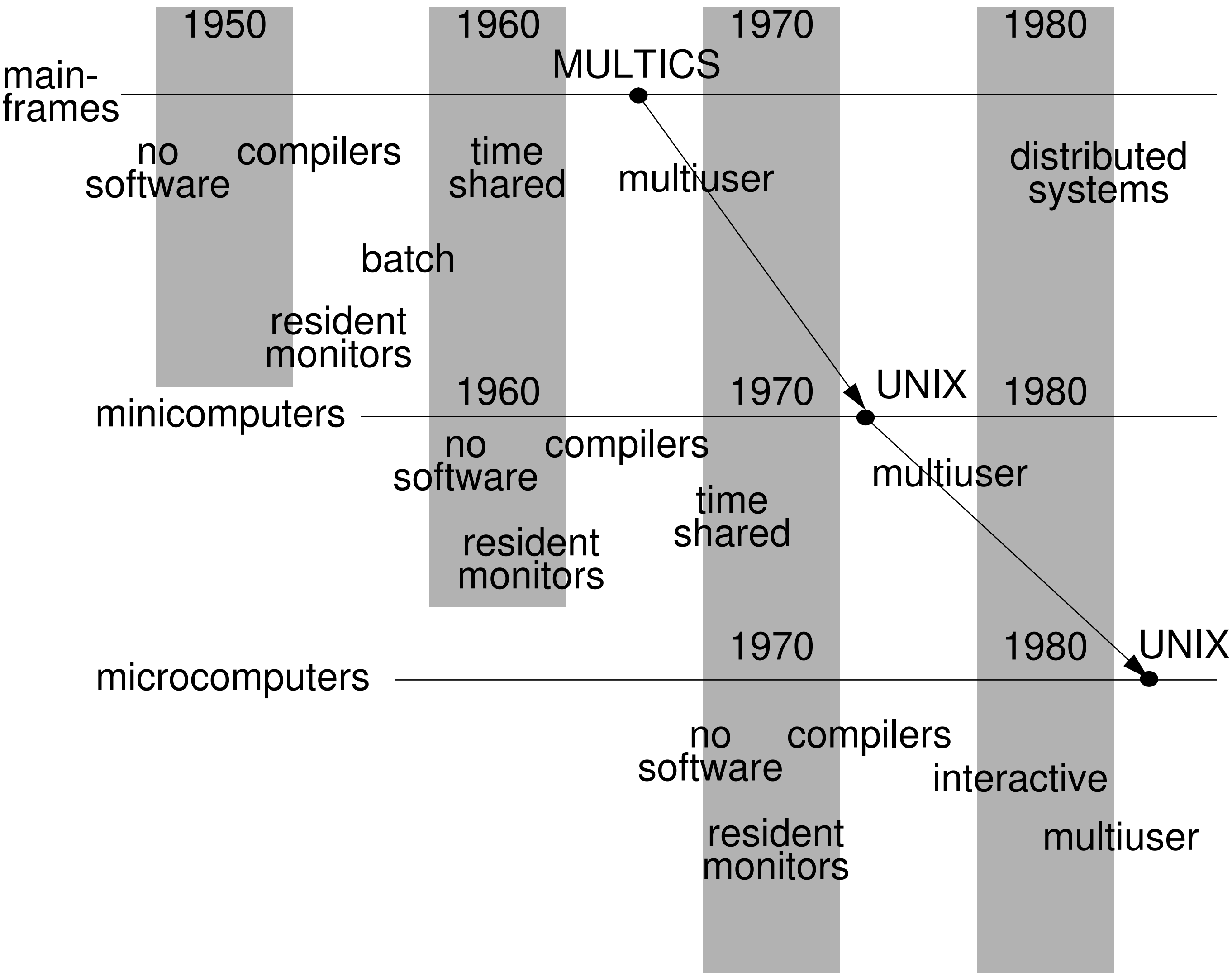
Bemerkungen zur Betriebssystementwicklung

In the world of operating systems, the wheels of progress turn slowly.

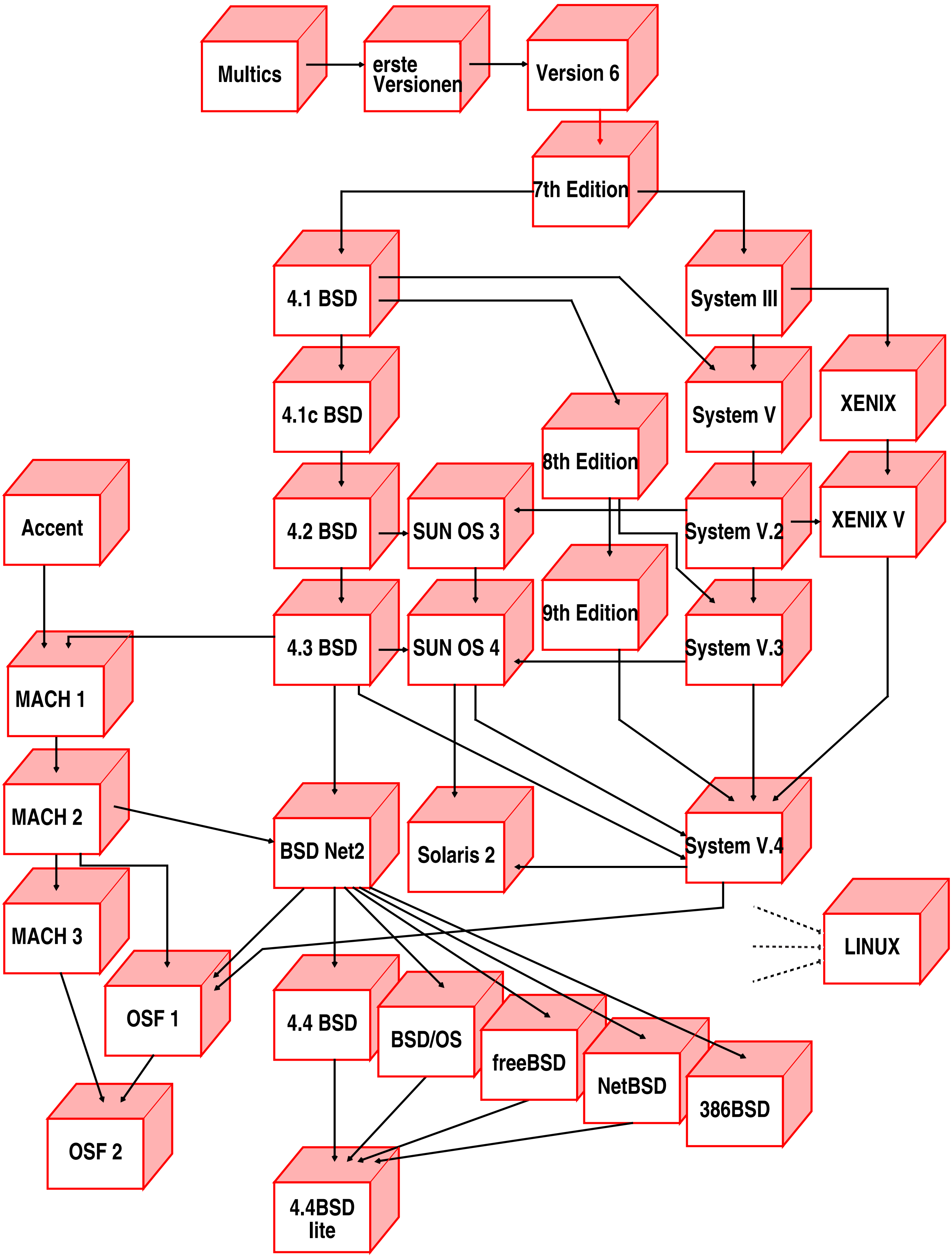
(Helen Custer: Inside Windows NT. Microsoft Press, 1993)

Migration of operating-system concepts and features

(Silberschatz, 1994)

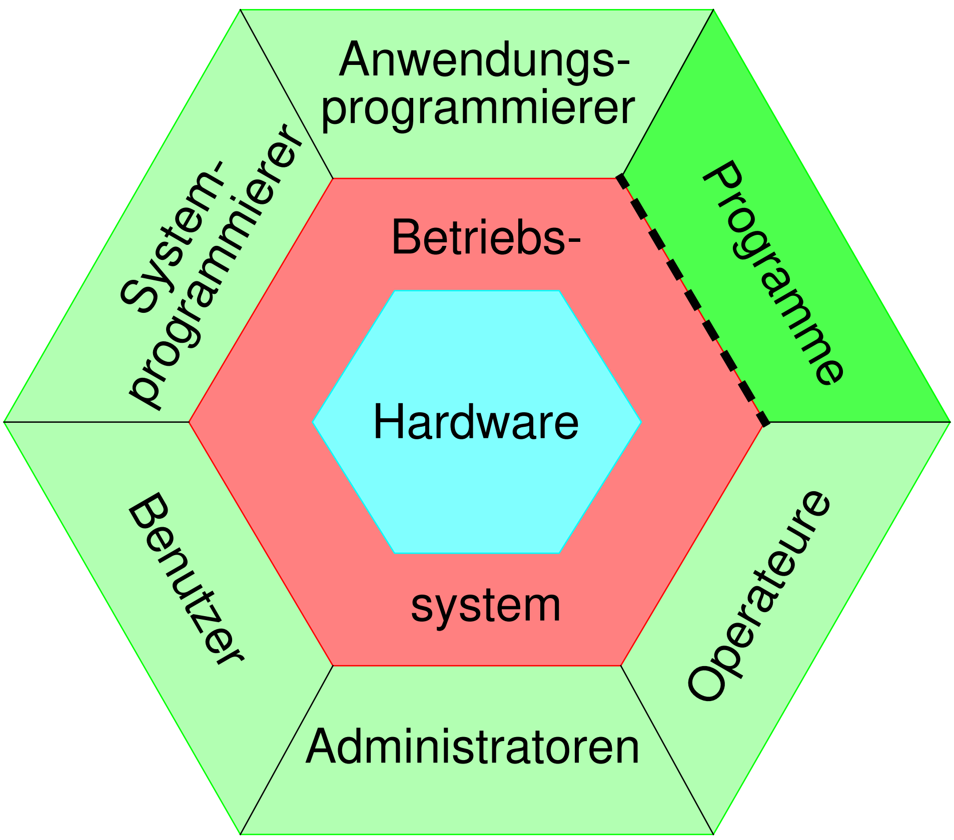


Die Entwicklung der Unix-Familie (Auswahl)

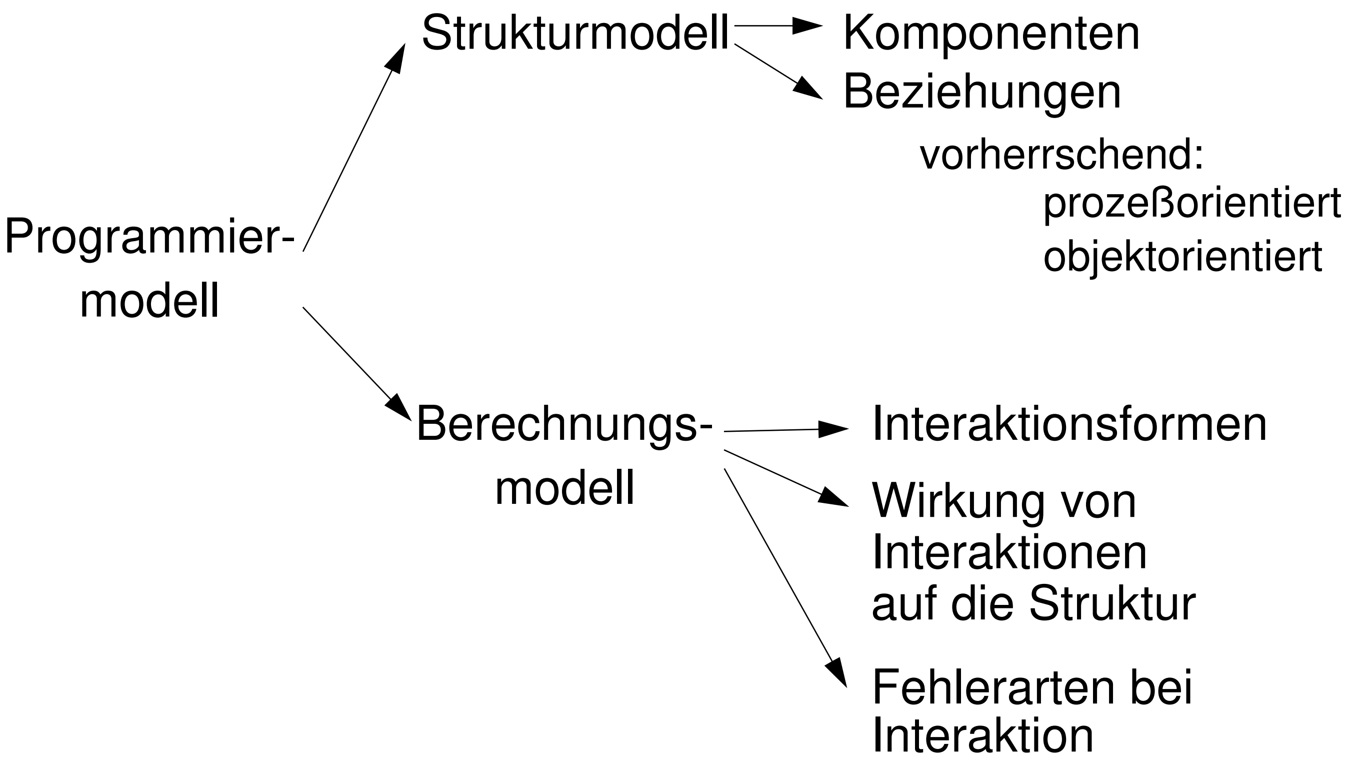


1.1.2 Ausführungsmodelle

- Abbildung der Benutzervorstellungen auf die Maschinengegebenheiten durch Bereitstellung geeigneter Abstraktionen und Schnittstellen zu:
  - Benutzern: Dialogbetrieb, graphische Benutzeroberflächen
  - Anwendungsprogrammierern: Programmiersprachen, Modularisierungshilfen
  - Systemprogrammierern: Wartung, Pflege
  - Operateuren: Gerätebedienung, Anpassung von Auswahlstrategien
  - Administratoren: Benutzerverwaltung, langfristige Systemsteuerung
  - Programmen: 'Supervisor-Calls (SVC)'
  - Hardware: Gerätetreiber



- Realisierungen von Programmiermodellen
- Befreiung von der Notwendigkeit, genaue Kenntnisse über Hardwareeigenschaften und spezielle Systemsoftwarekomponenten zu besitzen.
- Schaffung einer begrifflichen Basis zur Strukturierung von Programmsystemen und ihrer Ablaufsteuerung.

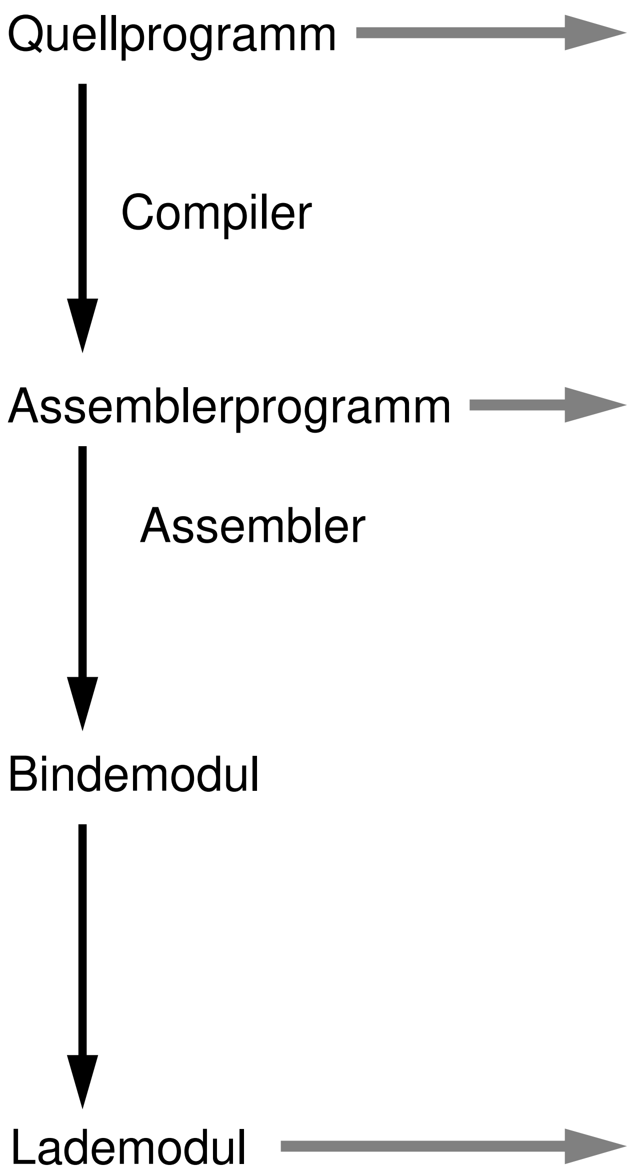


- Beispiele für Strukturkomponenten
  - Dateien (Behälter zur langfristigen Speicherung von Daten)
  - Prozesse (in Ausführung befindliche Programme)
  - Klassen (Vorlagen zur Bildung von Instanzen)
  - Instanzen
  - Prozeduren
  - Sockets (Kommunikationsendpunkte, “Kommunikations-steckdosen”)
  - Pipes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

- 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

1.2 Wie funktionieren Betriebssysteme?

- Vom Quellprogramm zum ausführbaren Speicherabbild




```
int x, c;

int main()
{ int a, b;
  a = 3;
  b = 5;
  c = a + b;
}
```

```
  . . .
main:   pushl %ebp
        movl %esp,%ebp
        subl $8,%esp
        movl $3,-4(%ebp)
        movl $5,-8(%ebp)
        movl -4(%ebp),%edx
        addl -8(%ebp),%edx
        movl %edx,c
        xorl %eax,%eax
        jmp .L1
        .align 16
        jmp .L1
        .align 16
.L1:    movl %ebp,%esp
        popl %ebp
        ret
```

```
00: 55
01: 89 E5
03: 83 EC 08
06: C7 45 FC 03 00 00 00
0D: C7 45 F8 05 00 00 00
14: 8B 55 FC
17: 03 55 F8
1A: 89 15 00 00 00 00
20: 31 C0
22: EB 1C
...
relocations
  Add value of "c" at
                address 1C
```

Lademodul

```
00: 55
01: 89 E5
03: 83 EC 08
06: C7 45 FC 03 00 00 00
0D: C7 45 F8 05 00 00 00
14: 8B 55 FC
17: 03 55 F8
1A: 89 15 00 00 00 00
20: 31 C0
22: EB 1C
...
relocations
  Add value of "a" at
                 address 1C
```

Lader (Loader)

dabei z. B.

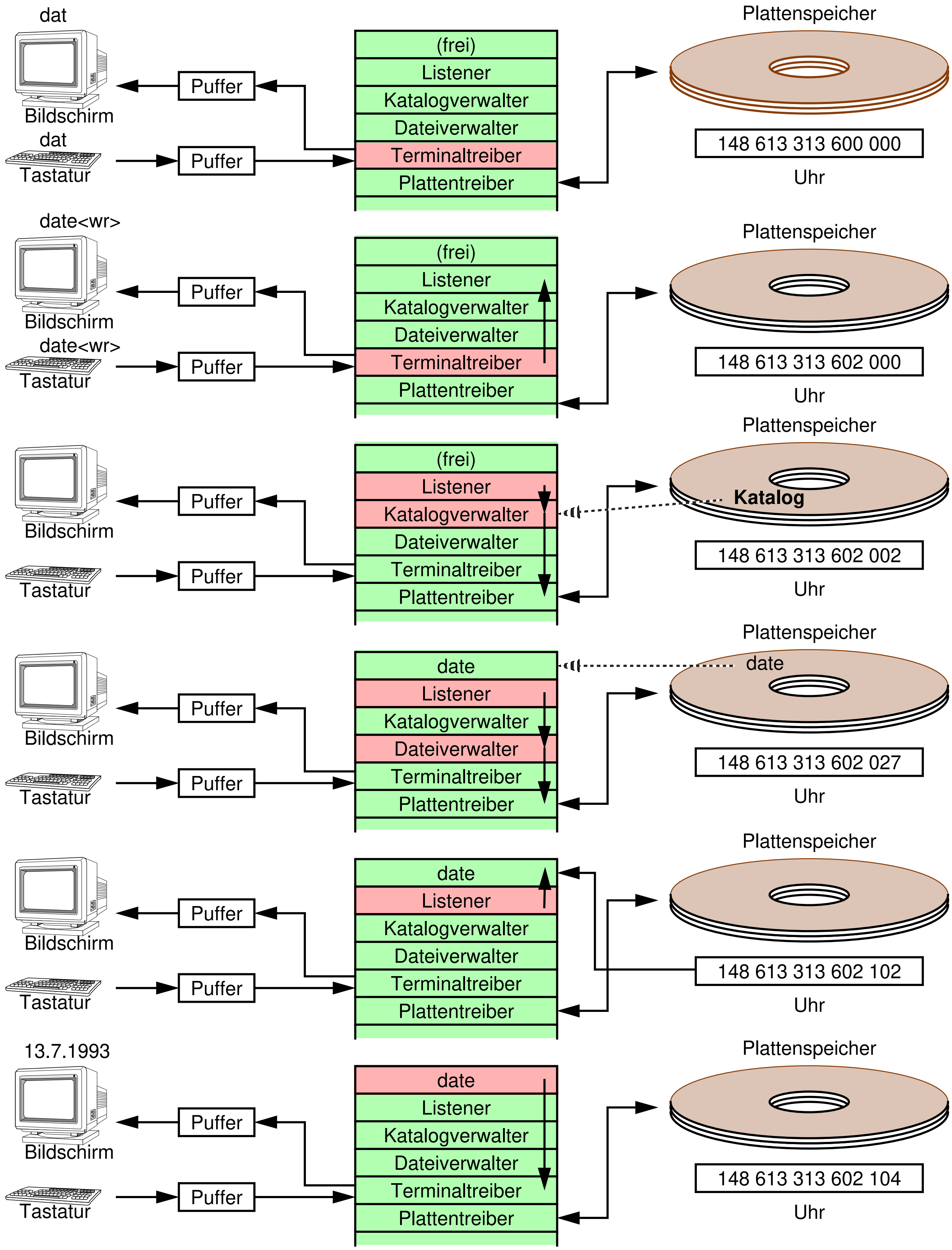
Ladeadresse für Befehle: 1000
Ladebefehle für Daten: 2100

Speicherabbild

```
1000: 55
1001: 89 E5
1003: 83 EC 08
1006: C7 45 FC 03 00 00 00
100D: C7 45 F8 05 00 00 00
1014: 8B 55 FC
1017: 03 55 F8
101A: 89 15 04 21 00 00
1020: 31 C0
1022: EB 1C
...
```

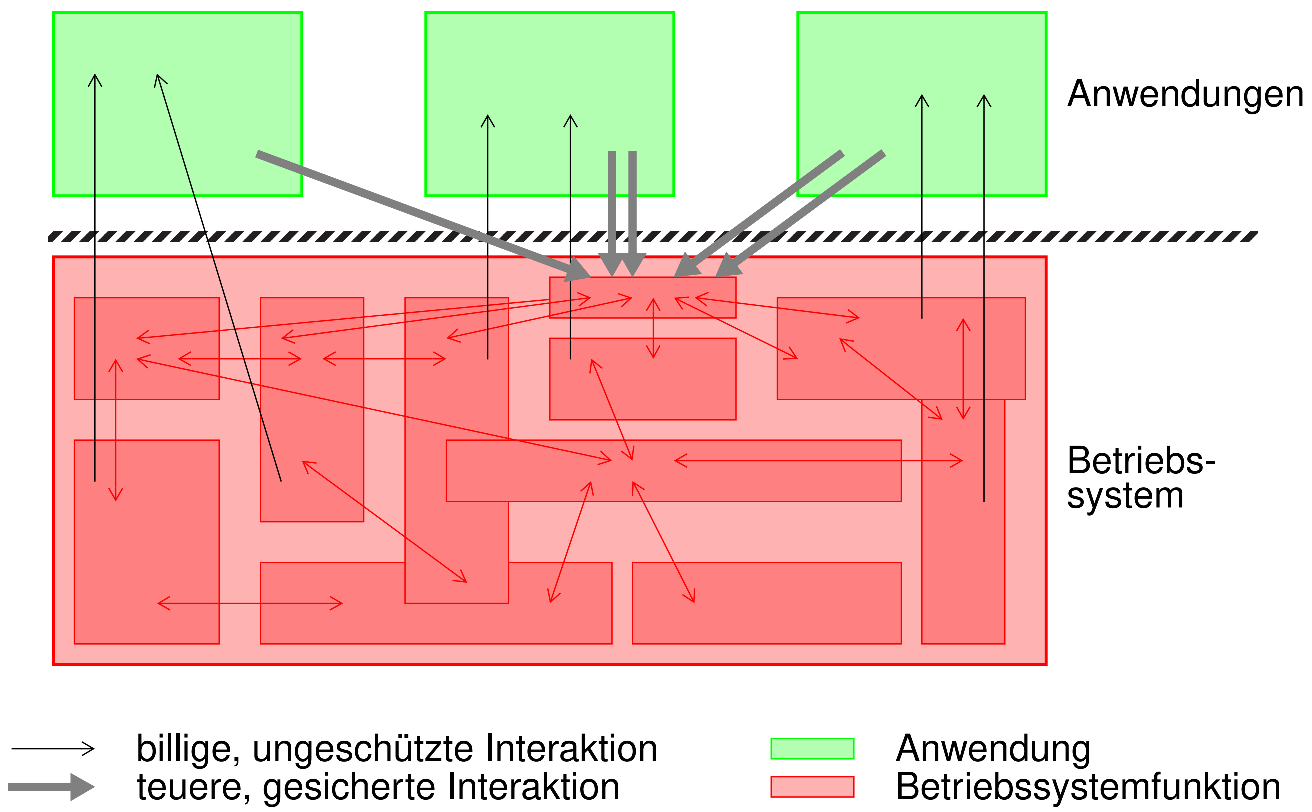
Ausführung

❑ Ausführungsbeispiel (nach Rembold)

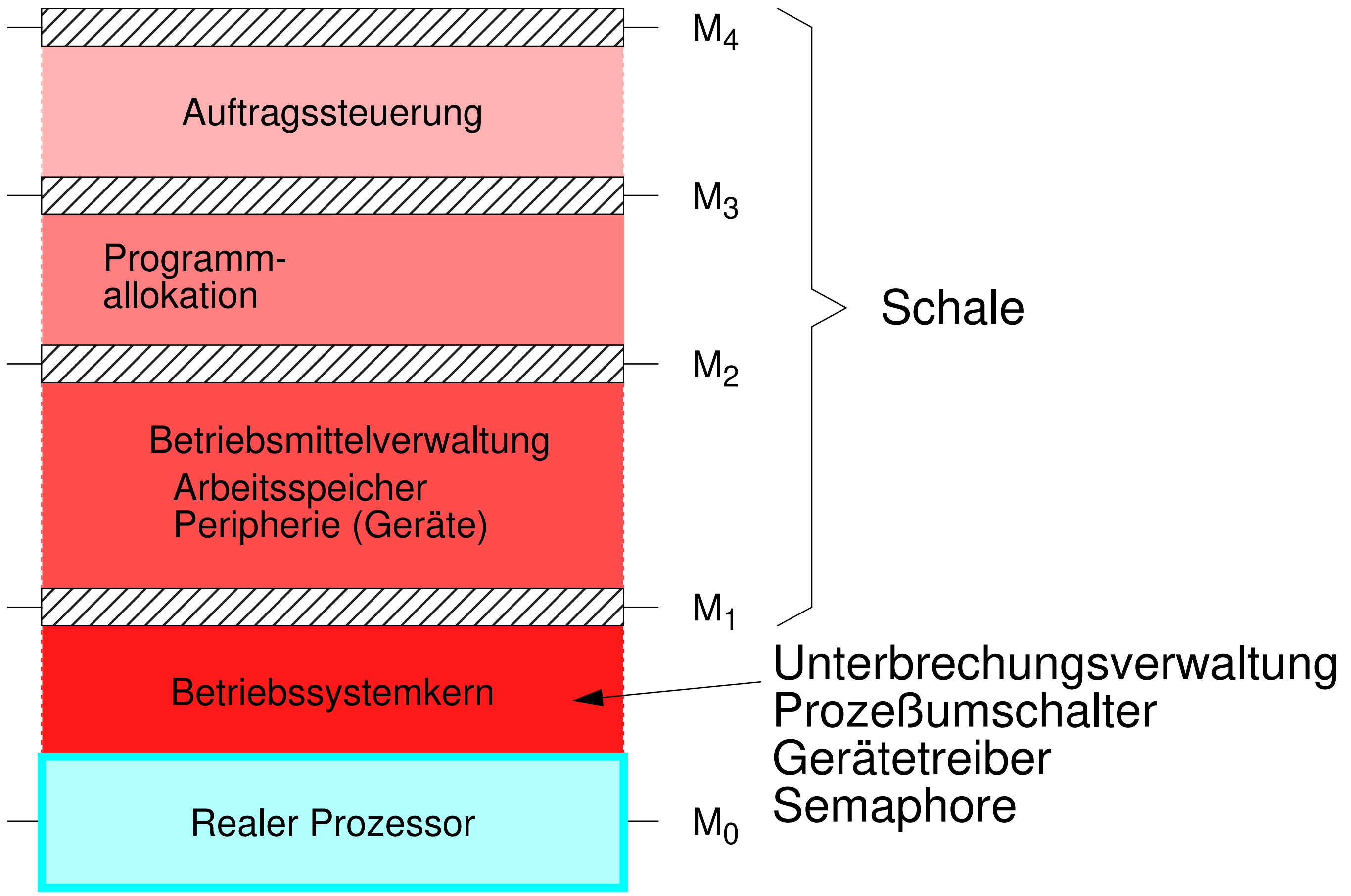


Wie werden Betriebssysteme realisiert?

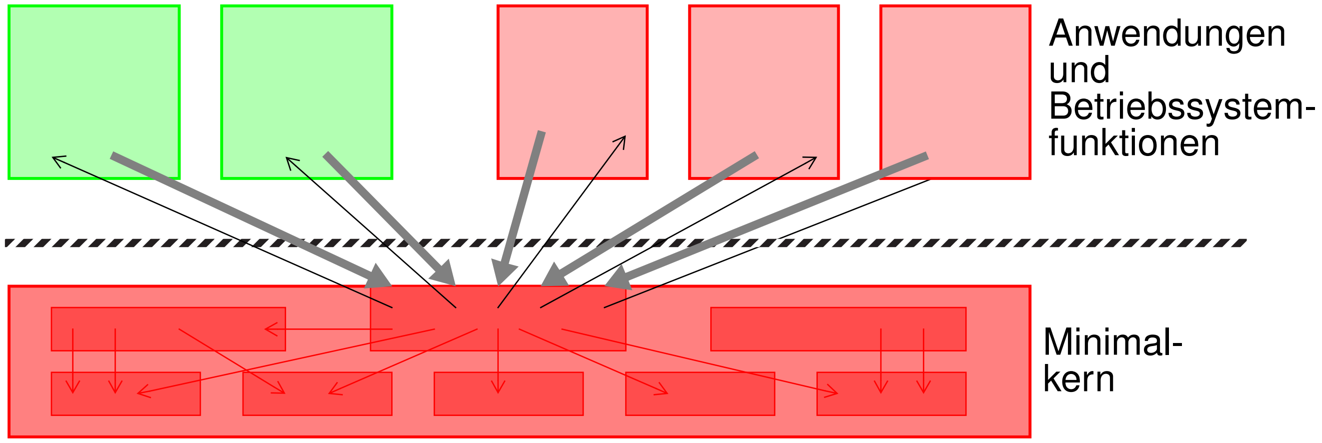
- Monolithische Systeme



❑ Geschichtete Systeme



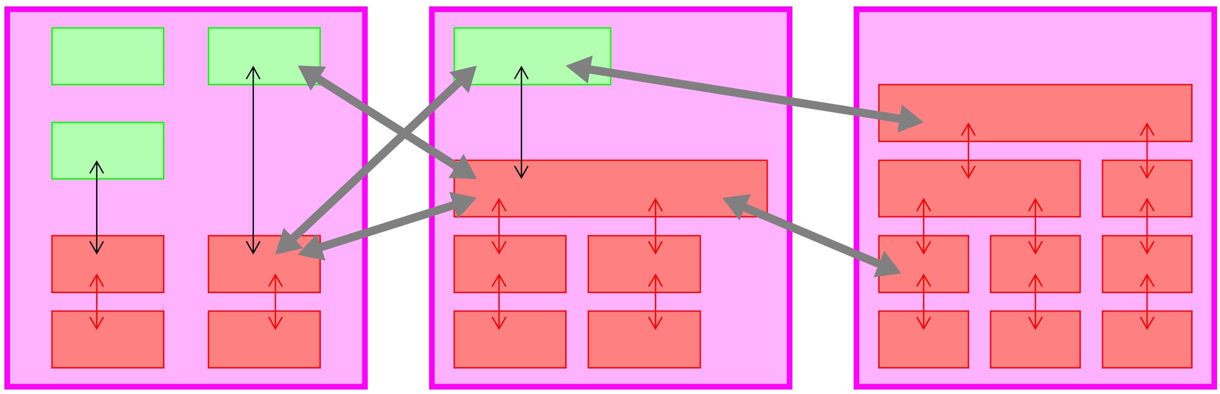
❏ Minimalkerne



billige, ungeschützte Interaktion
teuere, gesicherte Interaktion

Anwendung
Betriebssystemfunktion

❏ Objektbasierte, offene Betriebssysteme



billige, durch Objekt-Kapselung geschützte Interaktion

teuere, durch Adreßraumgrenze geschützte Interaktion

Anwendungsobjekt
Betriebssystemfunktion
Adreßraum

1.3 Warum sollte man das wissen?

- Rasche Einarbeitung in spezielle Systeme
  - MVS, BS2000, VM, Solaris, Unix, Windows NT, MS-DOS, ...
- Strukturierung komplexer Programmsysteme
  - Unterteilung in Komponenten
  - Interaktion zwischen den Komponenten
- Konzeption spezialisierter Systeme
  - embedded systems
  - Automatisierungssysteme
- Erstellung fehlertoleranter Systeme
- Ökonomische Nutzung der Hardware
- Laufzeitoptimierung anspruchsvoller Anwendungen

Ein Beispiel für rechenintensive, numerische Programme (Vorbesetzung zweier großer Matrizen)

```
#define DIM 2000

void main()
{ long i, j;
  long *b[DIM], *c[DIM];
  for (i = 0; i < DIM; i++) {
    b[i] = new long[DIM];
    c[i] = new long[DIM];
  }

  measure("begin");

  for (i = 0; i < DIM; i++) {
    for (j = 0; j < DIM; j++) {
      b[i][j] = 2;
      c[i][j] = 3;
    }
  }

  measure("change");

  for (j = 0; j < DIM; j++) {
    for (i = 0; i < DIM; i++) {
      b[i][j] = 2;
      c[i][j] = 3;
    }
  }

  measure("end");

  printTimes();
}
```

```
begin

usertime     =   0.33;  systemtime =   4.69;
waitingtime =  -4.22;  realtime   =   0.80

change

usertime     =   6.70;  systemtime =   7.57;
waitingtime =  33.79;  realtime   =  48.06

differences:
usertime     =   6.37;  systemtime =   2.88;
waitingtime =  38.02;  realtime   =  47.27

utilization = 19%, portion of systemtime = 31%


end

usertime     =  13.94;  systemtime =  15.49;
waitingtime = 260.49;  realtime   = 289.92

differences:
usertime     =   7.24;  systemtime =   7.92;
waitingtime = 226.70;  realtime   = 241.86

utilization = 6%,  portion of systemtime = 52%
```

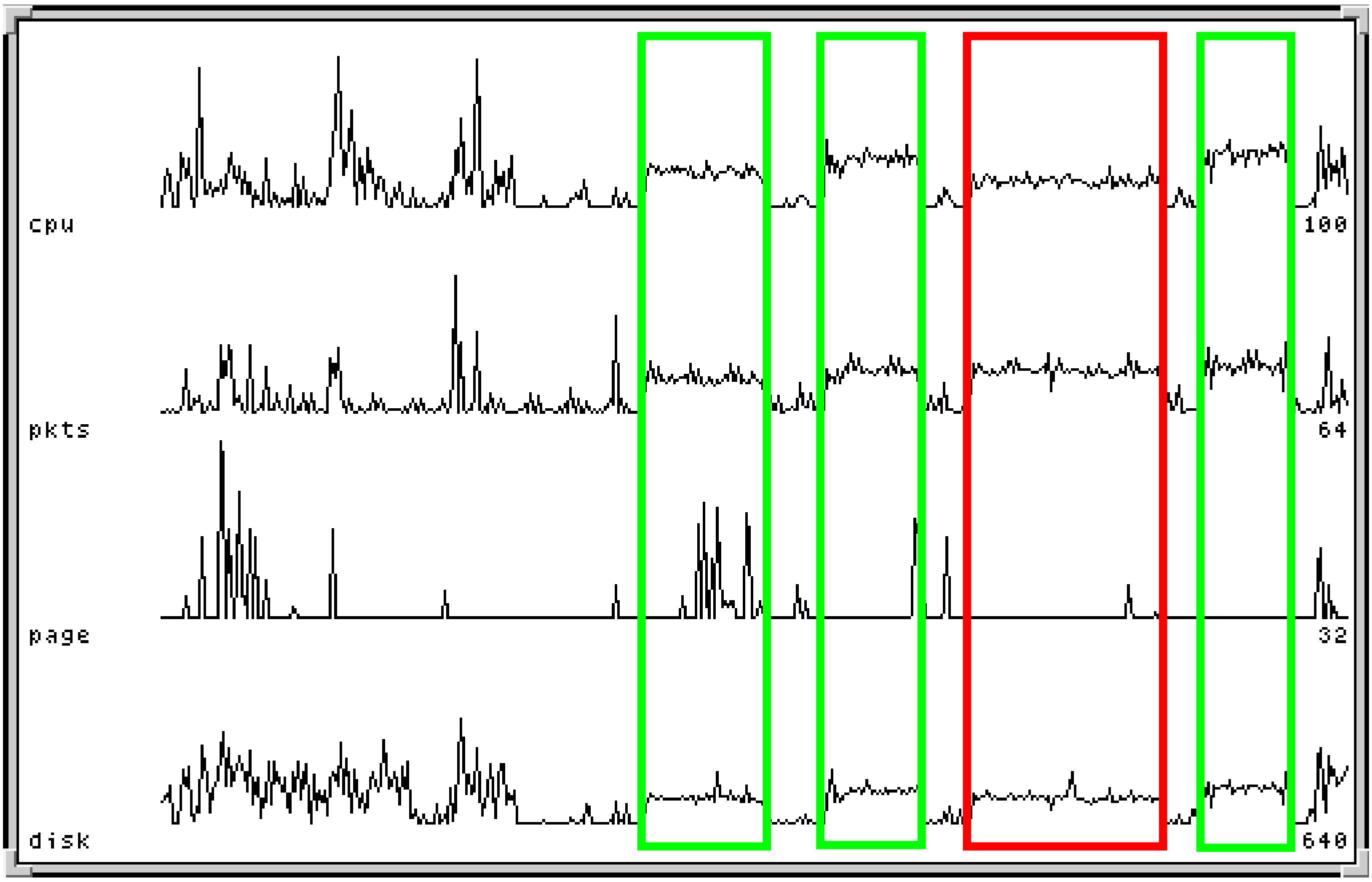
Ein Beispiel mit intensiver Dateibearbeitung
(Sequentielles Beschreiben mit unterschiedlichen Pufferlängen und in verschiedenen Modi)

```
rechenzeit.messen("Anfang");

for (i = 1; i <= length; i++) {

  // creating cpu load
  x = 1;
  for (k = 0; k < last; k++) x = x / y;

  buffer[l++] = i & 255;
  if (l >= bufferlength) {
    result = write(fd, buffer, l);
    l = 0;
  }
}

rechenzeit.messen("Ende");
```

Synchrones sequentielles Beschreiben einer Datei mit 8192 kB

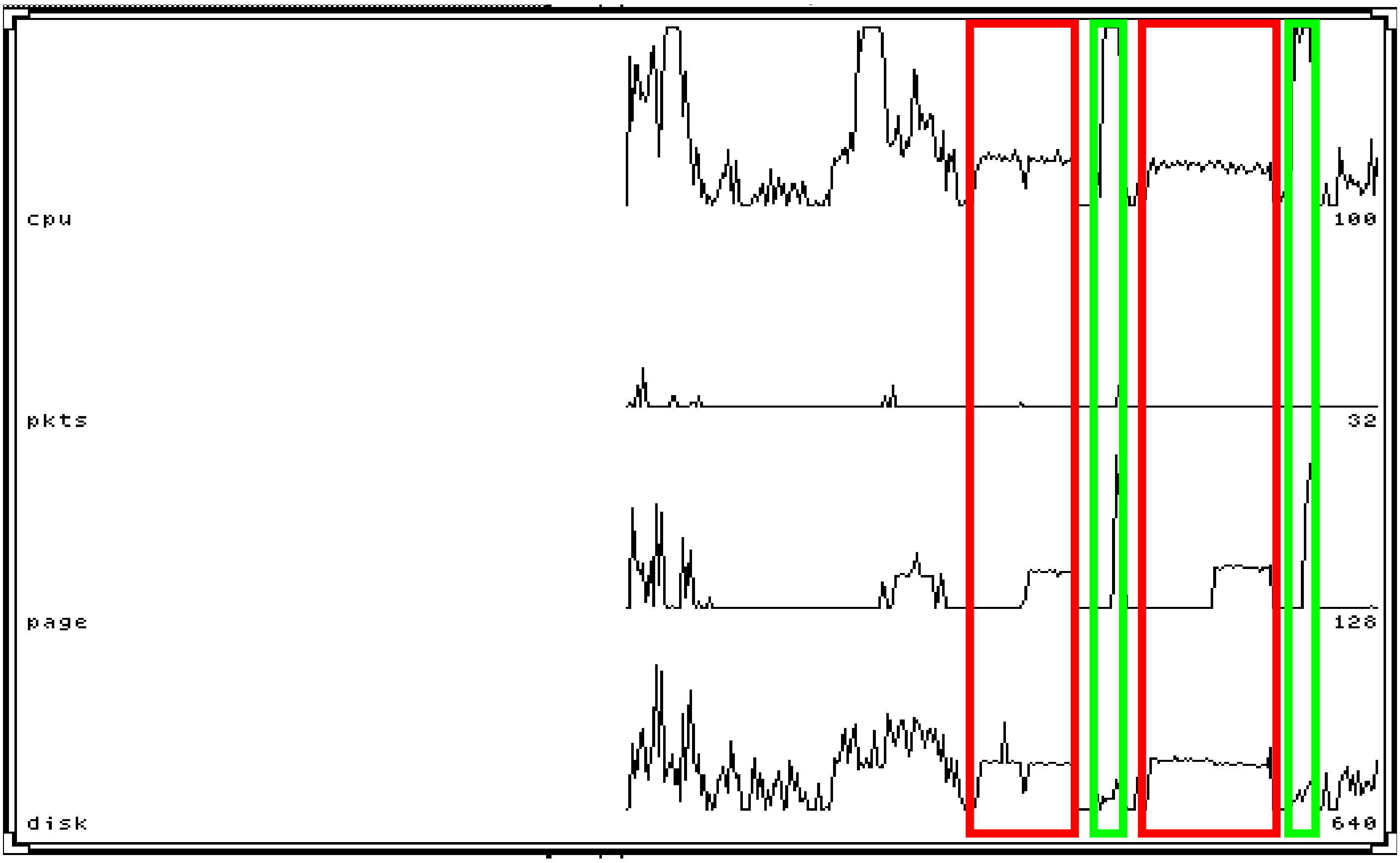
Puffer-länge	Verweil-zeit	Puffer-länge	Verweil-zeit	Puffer-länge	Verweil-zeit
4093	136.14	8189	96.61	16381	75.42
4094	135.84	8190	94.19	16382	75.06
4095	135.71	8191	93.91	16383	74.73
4096	**111.73**	**8192**	**67.30**	**16384**	**61.45**
4097	136.12	8193	96.60	16385	75.75
4098	135.91	8194	97.35	16386	76.45
4099	136.26	8195	99.14	16387	75.87
4100	135.69	8196	96.72	16388	76.12
4101	136.68	8197	96.91	16389	75.88

a) Ausgabe zum Dateiserver über Ethernet



Zahl der write-Aufrufe	1000	1000	1000	1000
Divisionszahl	0	0	0	0
Ausgabe asynchron (0) oder synchron (1)	1	0	1	0
Pufferlaenge in Bytes	8192	8192	8191	8191
Benutzermodus (sec)	13.85	13.71	13.89	13.56
Systemmodus (sec)	3.86	2.50	6.19	2.99
Wartezeit (sec)	73.57	54.61	121.89	46.51
Realzeit (sec)	91.28	70.82	141.97	63.06
Auslastung	19%	22%	14%	26%
Anteil der Systemzeit	21%	15%	30%	18%

b) Ausgabe auf lokalen Plattenspeicher



Zahl der write-Aufrufe	1000	1000	1000	1000
Divisionszahl	0	0	0	0
Ausgabe asynchron (0) oder synchron (1)	1	0	1	0
Pufferlaenge in Bytes	8192	8192	8191	8191
Benutzermodus (sec)	13.63	13.18	13.73	13.25
Systemmodus (sec)	3.34	1.54	4.30	2.04
Wartezeit (sec)	55.18	0.24	74.80	0.54
Realzeit (sec)	72.15	14.96	92.83	15.83
Auslastung	23%	98%	19%	96%
Anteil der Systemzeit	19%	10%	23%	13%