

5 Invertierte Seiten-Kacheltabelle (2)

- Vorteile
 - ◆ wenig Platz zur Speicherung der Abbildung notwendig
 - ◆ Tabelle kann immer im Hauptspeicher gehalten werden
- ▲ Nachteile
 - ◆ prozesslokale SKT zusätzlich nötig für Seiten, die ausgelagert sind
 - diese können aber ausgelagert werden
 - ◆ Suche in der KST ist aufwendig
 - Einsatz von Assoziativspeichern und Hashfunktionen

6 Systemaufruf

- Ermitteln der Seitengröße des Betriebssystems

```
int getpagesize(void);
```

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

E-Memory.fm 1999-12-14 13.02

E.35

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

E.4 Fallstudie: Pentium

- Physikalische Adresse
 - ◆ 32 bit breit
- Segmente
 - ◆ CS – Codesegment: enthält Instruktionen
 - ◆ DS – Datensegment
 - ◆ SS – Stacksegment
 - ◆ ES, FS, GS – zusätzliche Segmente
- Befehle beziehen sich auf eines oder mehrere der Segmente
- Segmentadressierung
 - ◆ Segmentselektor zur Auswahl eines Segments:
 - 16 bit bezeichnen das Segment

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

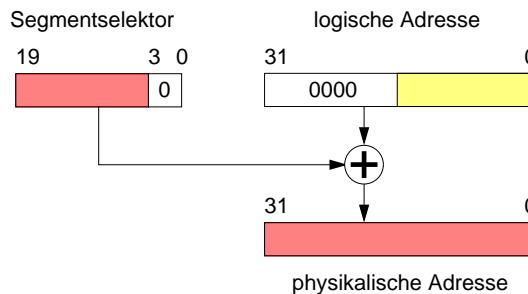
E-Memory.fm 1999-12-14 13.02

E.36

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Real Mode Adressierung

- Adressgenerierung im Real Mode
 - ◆ 16 bit breiter Segmentsselektor wird als 20 bit breite Adresse interpretiert und auf die logische Adresse addiert



SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

E-Memory.fm 1999-12-14 13.02

E.37

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Protected Mode Adressierung

- Vier Betriebsmodi (Stufen von Privilegien)
 - ◆ Stufe 0: höchste Privilegien (privilegierte Befehle, etc.): BS Kern
 - ◆ Stufe 1: BS Treiber
 - ◆ Stufe 2: BS Erweiterungen
 - ◆ Stufe 3: Benutzerprogramme
- Speicherverwaltung kann nur in Stufe 0 konfiguriert werden
- Segmentselektoren enthalten Privilegierungsstufe
 - ◆ Stufe des Codesegments entscheidet über Zugriffserlaubnis
- Segmentselektoren werden als Indizes interpretiert
 - ◆ Tabellen von Segmentdeskriptoren
 - Globale Deskriptor Tabelle
 - Lokale Deskriptor Tabelle

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

E-Memory.fm 1999-12-14 13.02

E.38

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Protected Mode Adressierung (2)

■ Deskriptortabelle

- ◆ enthält bis zu 8192 Segmentdeskriptoren
- ◆ Inhalt des Segmentdeskriptors:
 - physikalische Basisadresse
 - Längenangabe
 - Granularität (Angaben für Bytes oder Seiten)
 - Präsenzbit
 - Privilegierungsstufe
- ◆ globale Deskriptortabelle für alle Prozesse zugänglich (Register GDTR)
- ◆ lokale Deskriptortabelle pro Prozess möglich
(Register LDTR gehört zum Prozesskontext)

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

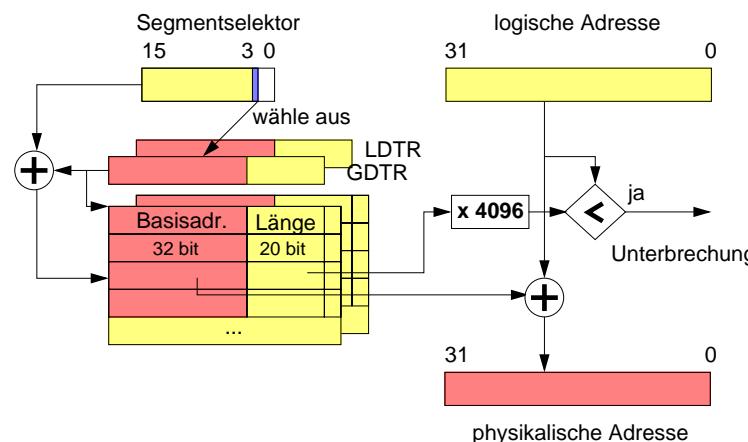
E-Memory.fm 1999-12-14 13.02

E.39

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Adressberechnung bei Segmentierung

■ Verwendung der Protected mode Adressierung



SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

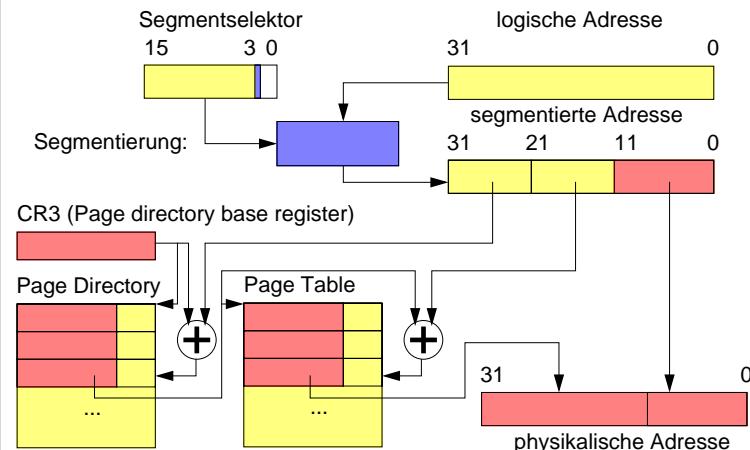
E-Memory.fm 1999-12-14 13.02

E.40

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Adressberechnung bei Paging

■ Seitenadressierung wird der Segmentierung nachgeschaltet



SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

E-Memory.fm 1999-12-14 13.02

E.41

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Adressberechnung bei Paging

■ Zweistufige Seitenadressierung

- ◆ Directory — Page table
- ◆ Seitengröße fest auf 4096 Bytes

■ Inhalt des Seitendeskriptor

- ◆ Kacheladresse
- ◆ Dirty Bit: Seite wurde beschrieben
- ◆ Access Bit: Seite wurde gelesen oder geschrieben
- ◆ Schreibschutz: Seite nur lesbar
- ◆ Präsenzbit: Seite ausgelagert (31 Bits für BS-Informationen nutzbar)
- ◆ Kontrolle des Prozessorcaches

■ Getrennte TLBs für Codesegment und Datensegmente

- ◆ 64 Einträge für Datenseiten; 32 Einträge für Codeseiten

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

E-Memory.fm 1999-12-14 13.02

E.42

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

E.5 Gemeinsamer Speicher (*Shared Memory*)

- Speicher, der mehreren Prozessen zur Verfügung steht
 - ◆ gemeinsame Segmente (gleiche Einträge in verschiedenen Segmenttabellen)
 - ◆ gemeinsame Seiten (gleiche Einträge in verschiedenen SKTs)
 - ◆ gemeinsame Seitenbereiche (gemeinsames Nutzen einer SKT bei mehrstufigen Tabellen)
- Gemeinsamer Speicher wird beispielsweise benutzt für
 - ◆ Kommunikation zwischen Prozessen
 - ◆ gemeinsame Befehlssegmente

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

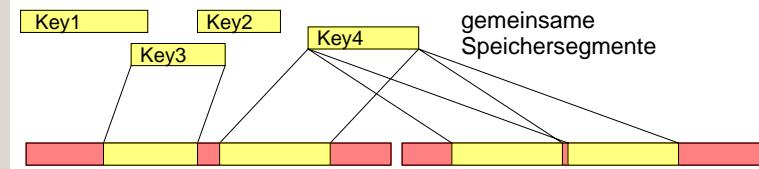
E-Memory.fm 1999-12-14 13.02

E.43

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

E.5 Gemeinsamer Speicher (3)

- Prinzip der `shm*` Operationen



Addressraum Prozess 1

Addressraum Prozess 2

```
id1= shmget( Key3, ...);      id1= shmget( Key4, ...);
shmat( id1, NULL, ...);      shmat( id1, NULL, ...);
id2= shmget( Key4, ...);      shmat( id1, NULL, ...);
shmat( id2, NULL, ...);
```

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

E-Memory.fm 1999-12-14 13.02

E.45

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

E.5 Gemeinsamer Speicher (2)

- Systemaufrufe unter Solaris 2.5
 - ◆ Erzeugen bzw. Holen eines gemeinsamen Speichersegments

```
int shmget( key_t key, int size, int shmflg );
```
 - ◆ Einblenden und Ausblenden des Segments in den Speicher

```
void *shmat( int shmid, void *shmaddr, int shmflg );
int shmdt( void *shmaddr );
```
 - ◆ Kontrolloperation

```
int shmctl( int shmid, int cmd, struct shmid_ds *buf );
```

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

E-Memory.fm 1999-12-14 13.02

E.44

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

E.5 Gemeinsamer Speicher (4)

- Verwendung des Keys
 - ◆ Alle Prozesse, die auf ein Speichersegment zugreifen wollen, müssen den Key kennen
 - ◆ Keys sind eindeutig innerhalb eines (Betriebs-)Systems
 - ◆ Ist ein Key bereits vergeben, kann kein Segment mit gleichem Key erzeugt werden
 - ◆ Ist ein Key bekannt, kann auf das Segment zugegriffen werden
 - gesetzte Zugriffsberechtigungen werden allerdings beachtet
 - ◆ Es können Segmente ohne Key erzeugt werden (private Segmente)
- Keys werden benutzt für:
 - ◆ Queues
 - ◆ Semaphore
 - ◆ Shared memory segments

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

E-Memory.fm 1999-12-14 13.02

E.46

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

E.6 Virtueller Speicher

- Entkoppelung des Speicherbedarfs vom verfügbaren Hauptspeicher
 - ◆ Prozesse benötigen nicht alle Speicherstellen gleich häufig
 - bestimmte Befehle werden selten oder gar nicht benutzt (z.B. Fehlerbehandlungen)
 - bestimmte Datenstrukturen werden nicht voll belegt
 - ◆ Prozesse benötigen evtl. mehr Speicher als Hauptspeicher vorhanden
- Idee
 - ◆ Vortäuschen eines großen Hauptspeichers
 - ◆ Einblenden benötigter Speicherbereiche
 - ◆ Abfangen von Zugriffen auf nicht-eingeblendete Bereiche
 - ◆ Bereitstellen der benötigten Bereiche auf Anforderung
 - ◆ Auslagern nicht-benötigter Bereiche

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

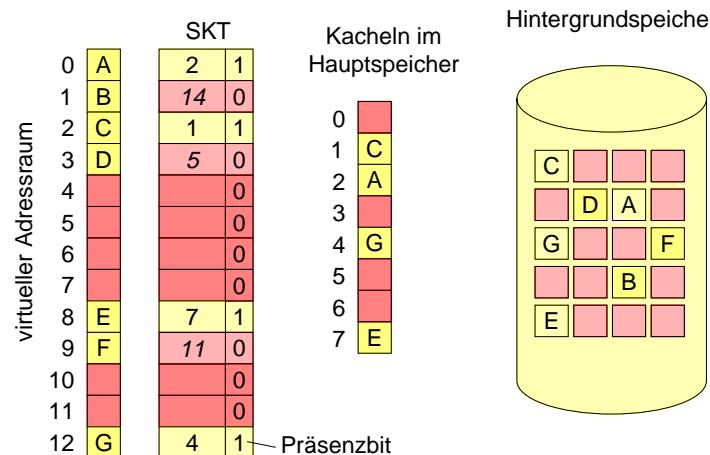
E-Memory.fm 1999-12-14 13.02

E.47

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Demand Paging

- Bereitstellen von Seiten auf Anforderung



SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

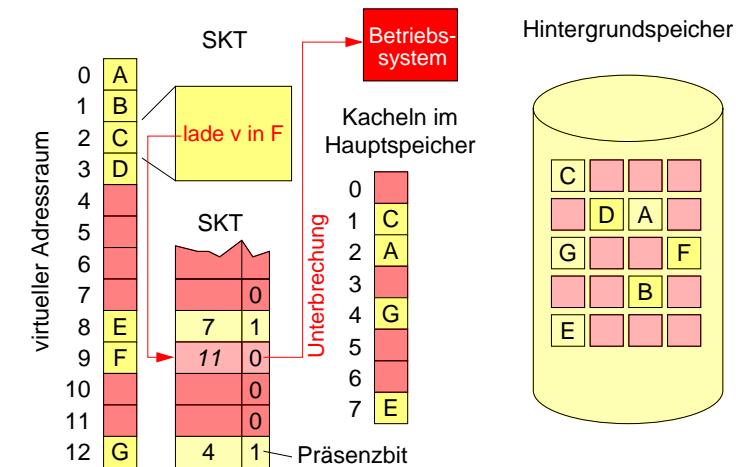
E-Memory.fm 1999-12-14 13.02

E.48

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Demand Paging (2)

- Reaktion auf Seitenfehler



SP I

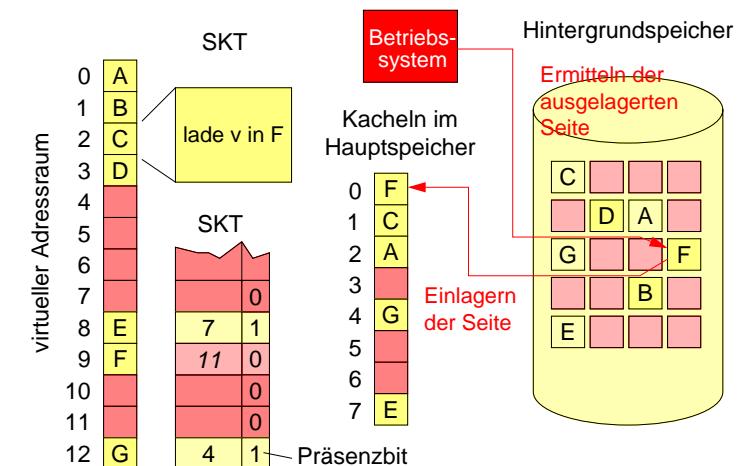
Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

E-Memory.fm 1999-12-14 13.02 E.49

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Demand Paging (3)

- Reaktion auf Seitenfehler



SP I

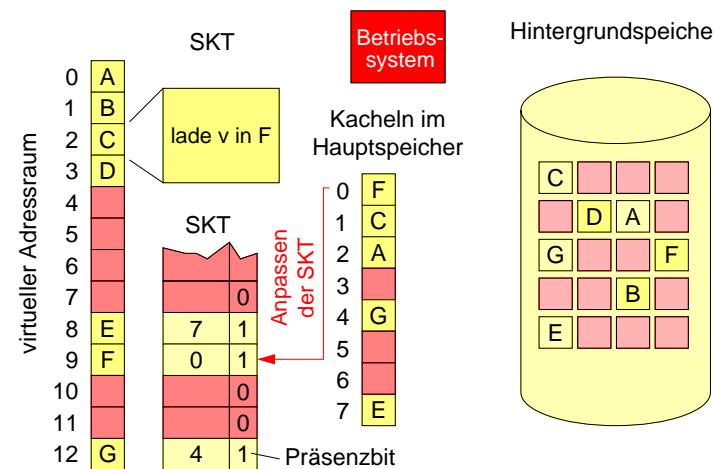
Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

E-Memory.fm 1999-12-14 13.02 E.50

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Demand Paging (4)

■ Reaktion auf Seitenfehler



SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

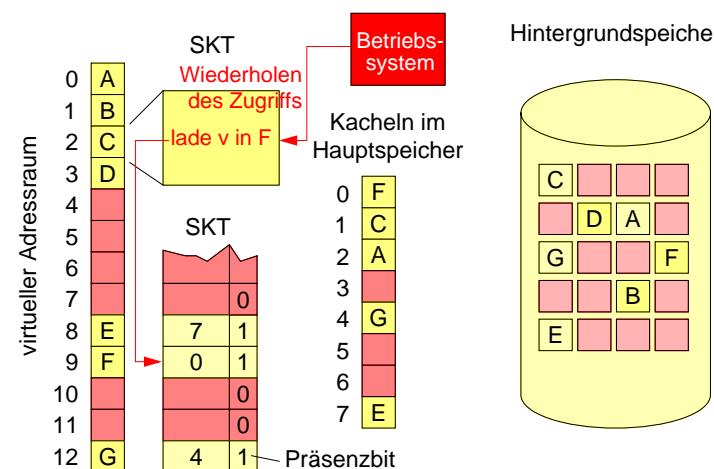
E-Memory.fm 1999-12-14 13.02

E.51

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Demand Paging (5)

■ Reaktion auf Seitenfehler



SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

E-Memory.fm 1999-12-14 13.02

E.52

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Demand Paging (6)

▲ Performanz von Demand paging

◆ Keine Seitenfehler

- effektive Zugriffszeit zw. 10 und 200 Nanosekunden

◆ Mit Seitenfehler

- p sei Wahrscheinlichkeit für Seitenfehler; p nahe Null
- Annahme: Zeit zum Einlagern einer Seite vom Hintergrundspeicher gleich 25 Millisekunden (8 ms Latenz, 15 ms Positionierzeit, 1 ms Übertragungszeit)
- Annahme: normale Zugriffszeit 100 ns
- Effektive Zugriffszeit:
$$(1 - p) \times 100 + p \times 25000000 = 100 + 24999900 \times p$$

▲ Seitenfehler müssen so niedrig wie möglich gehalten werden

■ Abwandlung: *Demand zero* für nicht initialisierte Daten

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

E-Memory.fm 1999-12-14 13.02

E.53

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Seitenersetzung

■ Was tun, wenn keine freie Kachel vorhanden?

- ◆ Eine Seite muss verdrängt werden, um Platz für neue Seite zu schaffen!
- ◆ Auswahl von Seiten, die nicht geändert wurden (*Dirty bit* in der SKT)
- ◆ Verdrängung erfordert Auslagerung, falls Seite geändert wurde

■ Vorgang:

- ◆ Seitenfehler (*Page fault*): Unterbrechung
- ◆ Auslagern einer Seite, falls keine freie Kachel verfügbar
- ◆ Einlagern der benötigten Seite
- ◆ Wiederholung des Zugriffs

▲ Problem

- ◆ Welche Seite soll ausgewählt werden?

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

E-Memory.fm 1999-12-14 13.02

E.54

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

E.7 Ersetzungsstrategien

- Betrachtung von Ersetzungsstrategien und deren Wirkung auf Referenzfolgen
- Referenzfolge
 - ◆ Folge von Seitennummern, die das Speicherzugriffsverhalten eines Prozesses abbildet
 - ◆ Ermittlung von Referenzfolgen z.B. durch Aufzeichnung der zugegriffenen Adressen
 - Reduktion der aufgezeichneten Sequenz auf Seitennummern
 - Zusammenfassung von unmittelbar hintereinanderstehenden Zugriffen auf die gleiche Seite
 - ◆ Beispiel für eine Referenzfolge: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

E-Memory.fm 1999-12-14 13.02

E.55

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 First-In, First-Out

- Älteste Seite wird ersetzt
- Notwendige Zustände:
 - ◆ Alter bzw. Einlagerungszeitpunkt für jede Kachel
- Ablauf der Ersetzungen (9 Einlagerungen)

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1	4	4	4	5	5	5	5	5	5
	Kachel 2		2	2	2	1	1	1	1	1	3	3	3
	Kachel 3			3	3	3	2	2	2	2	2	4	4
Kontrollzustände (Alter pro Kachel)	Kachel 1	0	1	2	0	1	2	0	1	2	3	4	5
	Kachel 2	>	0	1	2	0	1	2	3	4	0	1	2
	Kachel 3	>	>	0	1	2	0	1	2	3	4	0	1

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

E-Memory.fm 1999-12-14 13.02

E.56

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 First-In, First-Out

- Größerer Hauptspeicher mit 4 Kacheln (10 Einlagerungen)

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1	1	1	1	1	5	5	5	5	4
	Kachel 2		2	2	2	2	2	2	1	1	1	1	5
	Kachel 3			3	3	3	3	3	3	2	2	2	2
	Kachel 4				4	4	4	4	4	4	3	3	3
Kontrollzustände (Alter pro Kachel)	Kachel 1	0	1	2	3	4	5	0	1	2	3	0	1
	Kachel 2	>	0	1	2	3	4	5	0	1	2	3	0
	Kachel 3	>	>	0	1	2	3	4	5	0	1	2	3
	Kachel 4	>	>	>	0	1	2	3	4	5	0	1	2

- FIFO Anomalie (Belady's Anomalie, 1969)

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

E-Memory.fm 1999-12-14 13.02

E.57

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Optimale Ersetzungsstrategie

- Vorwärtsabstand
- Zeitdauer bis zum nächsten Zugriff auf die entsprechende Seite
- Strategie B_0 (OPT oder MIN) ist optimal (bei fester Kachelmenge): minimale Anzahl von Einlagerungen/Ersetzungen (hier 7)
 - ◆ „ersetze immer die Seite mit dem größten Vorwärtsabstand!“

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1	1	1	1	1	1	1	1	3	4
	Kachel 2		2	2	2	1	1	1	1	2	2	2	2
	Kachel 3			3	4	4	4	5	5	5	5	5	5
Kontrollzustände (Vorwärts- abstand)	Kachel 1	4	3	2	1	3	2	1	>	>	>	>	>
	Kachel 2	>	4	3	2	1	3	2	1	>	>	>	>
	Kachel 3	>	>	7	7	6	5	5	4	3	2	1	>

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

E-Memory.fm 1999-12-14 13.02

E.58

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Optimale Ersetzungsstrategie (2)

- Vergrößerung des Hauptspeichers (4 Kacheln): 6 Einlagerungen

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1	1	1	1	1	1	1	1	4	4
	Kachel 2	2	2	2	2	2	2	2	2	2	2	2	2
	Kachel 3		3	3	3	3	3	3	3	3	3	3	3
	Kachel 4		4	4	4	5	5	5	5	5	5	5	5
Kontrollzustände (Vorwärts- abstand)	Kachel 1	4	3	2	1	3	2	1	>	>	>	>	>
	Kachel 2	>	4	3	2	1	3	2	1	>	>	>	>
	Kachel 3	>	>	7	6	5	4	3	2	1	>	>	>
	Kachel 4	>	>	>	7	6	5	5	4	3	2	1	>

◆ keine Anomalie

SP I

3 Least Recently Used (LRU)

- Rückwärtsabstand
 - Zeitdauer, seit dem letzten Zugriff auf die Seite
- LRU Strategie (10 Einlagerungen)
 - „Ersetze die Seite mit dem größten Rückwärtsabstand!“

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1	4	4	4	5	5	5	3	3	3
	Kachel 2		2	2	2	1	1	1	1	1	1	1	4
	Kachel 3			3	3	3	2	2	2	2	2	2	5
	Kachel 4		0	1	2	0	1	2	0	1	2	0	1
Kontrollzustände (Rückwärts- abstand)	Kachel 1	>	0	1	2	0	1	2	0	1	2	0	1
	Kachel 2	>	>	0	1	2	0	1	2	0	1	2	0
	Kachel 3	>	>	0	1	2	0	1	2	0	1	2	0
	Kachel 4	>	>	>	0	1	2	3	4	5	0	1	2

SP I

2 Optimale Ersetzungsstrategie (3)

- Implementierung von B_0 nahezu unmöglich
 - Referenzfolge müsste vorher bekannt sein
 - B_0 meist nur zum Vergleich von Strategien brauchbar
- Suche nach Strategien, die möglichst nahe an B_0 kommen
 - z.B. *Least recently used* (LRU)

3 Least Recently Used (2)

- Vergrößerung des Hauptspeichers (4 Kacheln): 8 Einlagerungen

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1	1	1	1	1	1	1	1	1	5
	Kachel 2		2	2	2	2	2	2	2	2	2	2	2
	Kachel 3			3	3	3	3	5	5	5	5	4	4
	Kachel 4			4	4	4	4	4	4	4	3	3	3
Kontrollzustände (Rückwärts- abstand)	Kachel 1	0	1	2	3	0	1	2	0	1	2	3	0
	Kachel 2	>	0	1	2	3	0	1	2	0	1	2	3
	Kachel 3	>	>	0	1	2	3	0	1	2	3	0	1
	Kachel 4	>	>	>	0	1	2	3	4	5	0	1	2

SP I

SP I

3 Least Recently Used (3)

Keine Anomalie

- ◆ Allgemein gilt: Es gibt eine Klasse von Algorithmen (Stack-Algorithmen), bei denen keine Anomalie auftritt:
 - Bei Stack-Algorithmen ist bei n Kacheln zu jedem Zeitpunkt eine Untermenge der Seiten eingelagert, die bei $n+1$ Kacheln zum gleichen Zeitpunkt eingelagert wären!
 - LRU: Es sind immer die letzten n benutzten Seiten eingelagert
 - B_0 : Es sind die n bereits benutzten Seiten eingelagert, die als nächstes zugegriffen werden

Problem

- ◆ Implementierung von LRU nicht ohne Hardwareunterstützung möglich
- ◆ Es muss jeder Speicherzugriff berücksichtigt werden

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

E-Memory.fm 1999-12-14 13.02

E.63

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Least Recently Used (4)

Hardwareunterstützung durch Zähler

- ◆ CPU besitzt einen Zähler, der bei jedem Speicherzugriff erhöht wird (inkrementiert wird)
- ◆ bei jedem Zugriff wird der aktuelle Zählerwert in den jeweiligen Seitendeskriptor geschrieben
- ◆ Auswahl der Seite mit dem kleinsten Zählerstand

Aufwendige Implementierung

- ◆ viele zusätzliche Speicherzugriffe

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

E-Memory.fm 1999-12-14 13.02

E.64

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Second Chance (Clock)

Einsatz von Referenzbits

- ◆ Referenzbit im Seitendeskriptor wird automatisch durch Hardware gesetzt, wenn die Seite zugegriffen wird
 - einfacher zu implementieren
 - weniger zusätzliche Speicherzugriffe
 - moderne Prozessoren bzw. MMUs unterstützen Referenzbits (z.B. Pentium: Access bit)

Ziel: Annäherung von LRU

- ◆ das Referenzbit wird zunächst auf 0 gesetzt
- ◆ wird eine Opferseite gesucht, so werden die Kacheln reihum inspiziert
- ◆ ist das Referenzbit 1, so wird es auf 0 gesetzt (zweite Chance)
- ◆ ist das Referenzbit 0, so wird die Seite ersetzt

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

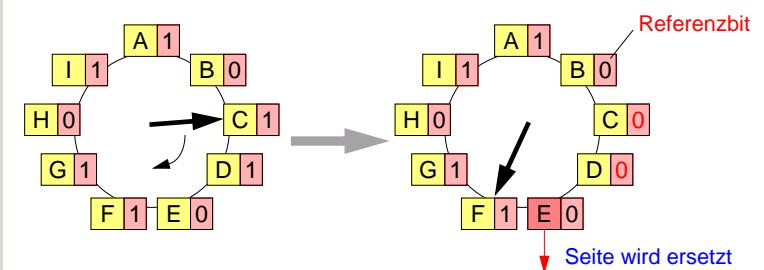
E-Memory.fm 1999-12-14 13.02

E.65

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Second Chance (2)

Implementierung mit umlaufendem Zeiger (Clock)



- ◆ an der Zeigerposition wird Referenzbit getestet

- falls Referenzbit eins, wird Bit gelöscht
- falls Referenzbit gleich Null, wurde ersetzbare Seite gefunden
 - Zeiger wird weitergestellt; falls keine Seite gefunden: Wiederholung
- ◆ falls alle Referenzbits auf 1 stehen, wird Second chance zu FIFO

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

E-Memory.fm 1999-12-14 13.02

E.66

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Second Chance (3)

Ablauf bei drei Kacheln (9 Einlagerungen)

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1	4	4	4	5	5	5	5	5	5
	Kachel 2		2	2	2	1	1	1	1	1	3	3	3
	Kachel 3			3	3	3	2	2	2	2	2	4	4
Kontrollzustände (Referenzbits)	Kachel 1	1	1	1	1	1	1	1	1	1	0	0	1
	Kachel 2	0	1	1	0	1	1	0	1	1	1	1	1
	Kachel 3	0	0	1	0	0	1	0	0	1	0	1	1
	Umlaufzeiger	2	3	1	2	3	1	2	2	2	3	1	1

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

E-Memory.fm 1999-12-14 13.02

E.67

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Second Chance (5)

Second chance zeigt FIFO Anomalie

- ♦ Wenn alle Referenzbits gleich 1, wird nach FIFO entschieden

Erweiterung

- ♦ Modifikationsbit kann zusätzlich berücksichtigt werden (*Dirty bit*)
- ♦ drei Klassen: (0,0), (1,0) und (1,1) mit (Referenzbit, Modifikationsbit)
- ♦ Suche nach der niedrigsten Klasse (Einsatz im MacOS)

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

E-Memory.fm 1999-12-14 13.02

E.69

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Second Chance (4)

Vergrößerung des Hauptspeichers (4 Kacheln): 10 Einlagerungen

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1	1	1	1	5	5	5	5	4	4
	Kachel 2		2	2	2	2	2	2	1	1	1	1	5
	Kachel 3			3	3	3	3	3	3	2	2	2	2
Kontrollzustände (Referenzbits)	Kachel 4				4	4	4	4	4	4	3	3	3
	Kachel 1	1	1	1	1	1	1	1	1	1	1	1	1
	Kachel 2	0	1	1	1	1	1	0	1	1	1	0	1
	Kachel 3	0	0	1	1	1	1	0	0	1	1	0	0
	Kachel 4	0	0	0	1	1	1	0	0	0	1	0	0
Umlaufzeiger		2	3	4	1	1	1	2	3	4	1	2	3

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

E-Memory.fm 1999-12-14 13.02

E.68

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

5 Freiseitenpuffer

Statt eine Seite zu ersetzen wird permanent eine Menge freier Seiten gehalten

- ♦ Auslagerung geschieht im „voraus“

- ♦ Effizienter: Ersetzungszeit besteht im Wesentlichen nur aus Einlagerungszeit

Behalten der Seitenzuordnung auch nach der Auslagerung

- ♦ Wird die Seite doch noch benutzt bevor sie durch eine andere ersetzt wird, kann sie mit hoher Effizienz wiederverwendet werden.

- ♦ Seite wird aus Freiseitenpuffer ausgetragen und wieder dem entsprechenden Prozess zugeordnet.

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

E-Memory.fm 1999-12-14 13.02

E.70

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

6 Seitenanforderung

- ▲ Problem: Zuordnung der Kacheln zu mehreren Prozessen
- Begrenzungen
 - ◆ Maximale Seitenmenge: begrenzt durch Anzahl der Kacheln
 - ◆ Minimale Seitenmenge: abhängig von der Prozessorarchitektur
 - Mindestens die Anzahl von Seiten nötig, die theoretisch bei einem Maschinenbefehl benötigt werden
(z.B. zwei Seiten für den Befehl, vier Seiten für die adressierten Daten)
- Gleiche Zuordnung
 - ◆ Anzahl der Prozesse bestimmt die Kachelmenge, die ein Prozess bekommt
- Größenabhängige Zuordnung
 - ◆ Größe des Programms fließt in die zugeteilte Kachelmenge ein

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

E-Memory.fm 1999-12-14 13.02

E.71

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

6 Seitenanforderung

- Globale und lokale Anforderung von Seiten
 - ◆ lokal: Prozess ersetzt nur immer seine eigenen Seiten
 - Seitenfehler-Verhalten liegt nur in der Verantwortung des Prozesses
 - ◆ global: Prozess ersetzt auch Seiten anderer Prozesse
 - bessere Effizienz, da ungenutzte Seiten von anderen Prozessen verwendet werden können

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

E-Memory.fm 1999-12-14 13.02

E.72

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

E.8 Seitenflattern (*Thrashing*)

- Ausgelagerte Seite wird gleich wieder angesprochen
 - ◆ Prozess verbringt mehr Zeit mit dem Warten auf das Beheben von Seitenfehler als mit der eigentlichen Ausführung
- Ursachen
 - ◆ Prozess ist nahe am Seitenminimum
 - ◆ zu viele Prozesse gleichzeitig im System
 - ◆ schlechte Ersetzungsstrategie
- ★ Lokale Seitenanforderung behebt Thrashing zwischen Prozessen
- ★ Zuteilung einer genügend großen Zahl von Kacheln behebt Thrashing innerhalb der Prozessseiten
 - ◆ Begrenzung der Prozessanzahl

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

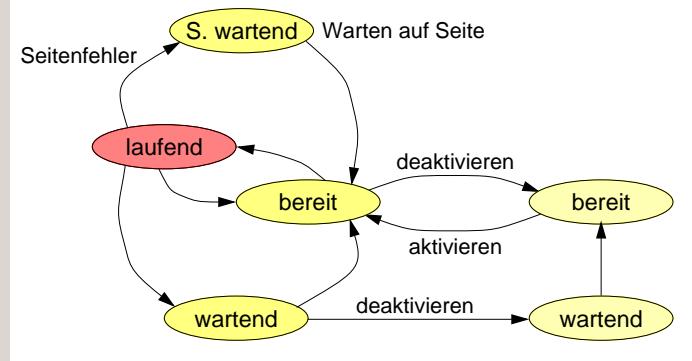
E-Memory.fm 1999-12-14 13.02

E.73

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Deaktivieren von Prozessen

- Einführung von „Superzuständen“



- ◆ inaktiver Prozess benötigt keine Kacheln; Prozess ist vollständig ausgelagert (swapped out)

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

E-Memory.fm 1999-12-14 13.02

E.74

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Deaktivieren von Prozessen (2)

- Sind zu viele Prozesse aktiv, werden welche deaktiviert
 - ◆ Kacheln teilen sich auf weniger Prozesse auf
 - ◆ Verbindung mit dem Scheduling nötig
 - Verhindern von Aushungerung
 - Erzielen kurzer Reaktionszeiten
 - ◆ guter Kandidat: Prozess mit wenigen Seiten im Hauptspeicher
 - geringe Latenz bei Wiedereinlagerung bzw. wenige Seitenfehler bei Aktivierung und Demand paging

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

E-Memory.fm 1999-12-14 13.02

E.75

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Arbeitsmengenmodell

- Menge der Seiten, die ein Prozess wirklich braucht (*Working set*)
 - ◆ kann nur angenähert werden, da üblicherweise nicht vorhersehbar
- Annäherung durch Betrachten der letzten Δ Seiten, die angesprochen wurden
 - ◆ geeignete Wahl von Δ
 - zu groß: Überlappung von lokalen Zugriffsmustern
 - zu klein: Arbeitsmenge enthält nicht alle nötigen Seiten



Referenzfolge

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

E-Memory.fm 1999-12-14 13.02

E.76

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Arbeitsmengenmodell (2)

- Beispiel: Arbeitsmengen bei verschiedenen Δ

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
$\Delta = 3$	Seite 1	x	x	x		x	x	x	x	x	x		
	Seite 2		x	x	x		x	x	x	x	x	x	
	Seite 3			x	x	x				x	x	x	
	Seite 4				x	x	x				x	x	
	Seite 5						x	x	x			x	
$\Delta = 4$	Seite 1	x	x	x	x	x	x	x	x	x	x	x	
	Seite 2		x	x	x	x	x	x	x	x	x	x	x
	Seite 3			x	x	x	x			x	x	x	
	Seite 4				x	x	x	x			x	x	
	Seite 5					x	x	x	x		x	x	

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

E-Memory.fm 1999-12-14 13.02

E.77

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Bestimmung der Arbeitsmenge

- Annäherung der Zugriffe durch die Zeit
 - ◆ bestimmtes Zeitintervall ist ungefähr proportional zu Anzahl von Speicherzugriffen
- ▲ Virtuelle Zeit des Prozesses muss gemessen werden
 - ◆ nur die Zeit relevant, in der der Prozess im Zustand laufend ist
 - ◆ Verwalten virtueller Uhren pro Prozess
- Referenzbit, Altersangabe und Timer-Interrupt
 - ◆ jede Seite enthält eine Altersangabe (Zeitintervall ohne Benutzung)
 - ◆ durch regelmäßigen Interrupt wird mittels Referenzbit die Altersangabe fortgeschrieben (bei Benutzung auf Null gesetzt, ansonsten erhöht); dabei werden nur die Seiten des gerade laufenden Prozesses „gealtert“
 - ◆ Seiten mit Alter > Δ sind nicht mehr in der Arbeitsmenge des jeweiligen Prozesses

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

E-Memory.fm 1999-12-14 13.02

E.78

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Bestimmung der Arbeitsmenge (2)

- ▲ Ungenau: System ist aber nicht empfindlich auf diese Ungenauigkeit
 - ◆ Verringerung der Zeitintervalle: höherer Aufwand, genauere Messung
- Algorithmus WSClock (Working set clock)
 - ◆ arbeitet wie Clock
 - ◆ Seite wird nur dann ersetzt, wenn sie nicht zur Arbeitsmenge ihres Prozesses gehört oder der Prozess deaktiviert ist
 - ◆ Bei Zurücksetzen des Referenzbits wird die virtuelle Zeit des jeweiligen Prozesses eingetragen, die z.B. im PCB gehalten und fortgeschrieben wird
 - ◆ Bestimmung der Arbeitsmenge erfolgt durch Differenzbildung von virtueller Zeit des Prozesses und Zeitstempel in der Kachel

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

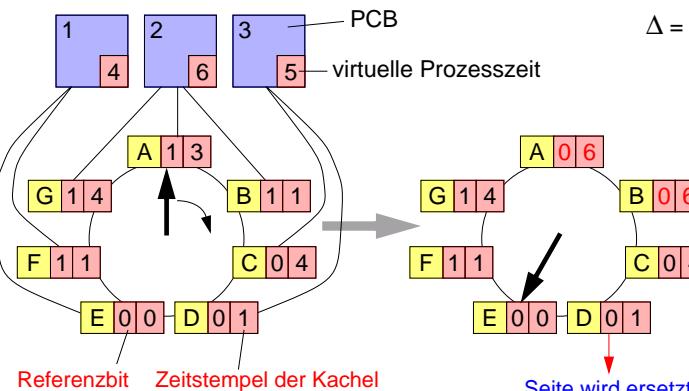
E-Memory.fm 1999-12-14 13.02

E.79

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Bestimmung der Arbeitsmenge (3)

- WSClock Algorithmus



SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

E-Memory.fm 1999-12-14 13.02

E.80

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Bestimmung der Arbeitsmenge (4)

- ▲ Zuordnung zu einem Prozess nicht immer möglich
 - ◆ gemeinsam genutzte Seiten in modernen Betriebssystemen eher die Regel als die Ausnahme
 - Seiten des Codesgments
 - Shared libraries
 - Gemeinsame Seiten im Datensegment (*Shared memory*)
 - ◆ moderne System bestimmen meist eine globale Arbeitsmenge von Seiten

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

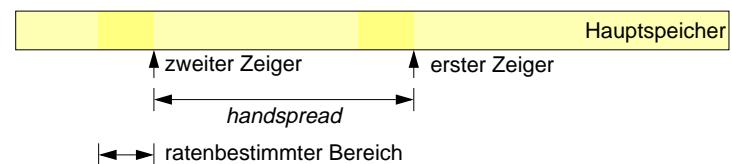
E-Memory.fm 1999-12-14 13.02

E.81

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Ersetzungsstrategie bei Solaris

- Prozess *pageout* arbeitet Clock-Strategie ab
 - ◆ Prozess läuft mehrmals die Sekunde (4x)
 - ◆ adaptierbare Rate: untersuchte Seiten pro Sekunde
 - ◆ statt ein Zeiger: zwei Zeiger
 - am ersten Zeiger werden Referenzbits zurückgesetzt
 - am zweiten Zeiger werden Seiten mit gelösctem Referenzbit ausgewählt
 - nötig, weil sonst Zeitspanne zwischen Löschen und Auswählen zu lang wird (großer Hauptspeicher; 64 MByte entsprechen 8.192 Seiten)
 - Zeigerabstand einstellbar (*handspread*)



SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

E-Memory.fm 1999-12-14 13.02

E.82

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Ersetzungsstrategie bei Solaris (2)

- ◆ ermittelte Seiten werden ausgelagert (falls nötig) und
- ◆ in eine Freiliste eingehängt
- ◆ aus der Freiliste werden Kacheln für Einlagerungen angefordert
- ◆ Seitenfehler können unbenutzte Seiten aus der Freiliste wieder zurückfordern (*Minor page faults*)

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

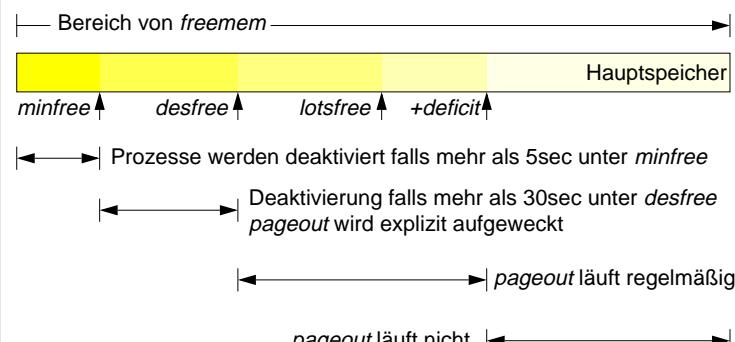
E-Memory.fm 1999-12-14 13.02

E.83

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Ersetzungsstrategie bei Solaris (3)

- Verhalten von *pageout* orientiert sich an Größe der Freiliste (Menge des freien Speichers)



- ◆ *deficit* wird dynamisch ermittelt (0 bis *lotsfree*) und auf *lotsfree* addiert
 - entspricht Vorschau auf künftige große Speicheranforderungen

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

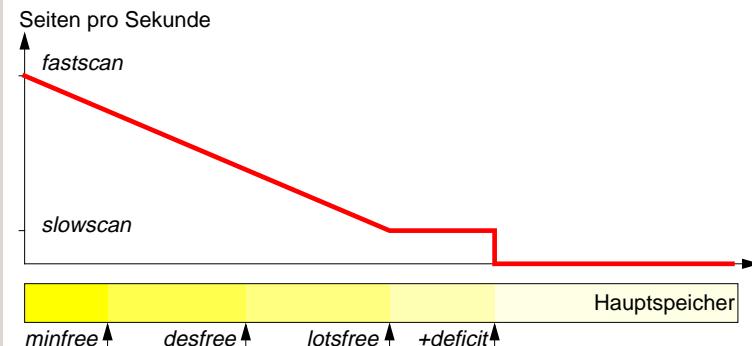
E-Memory.fm 1999-12-14 13.02

E.84

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Ersetzungsstrategie bei Solaris (4)

- Seitenuntersuchungsrate des *pageout* Prozesses



- ◆ je weniger freier Speicher verfügbar ist, desto höher wird die Untersuchungsrate
- ◆ *slowscan* und *fastscan* sind einstellbar

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

E-Memory.fm 1999-12-14 13.02

E.85

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Ersetzungsstrategie bei Solaris (5)

- Weitere Parameter

- ◆ *maxpgio*: maximale Transferrate bei Auslagerungen (vermeidet Plattsaturation)
- ◆ *autoup*: Zeitdauer des regelmäßigen Auslagerns alter Seiten durch den Prozess *flushd* (Default: alle 30 sec)

- Aktivieren und Deaktivieren (*Swap in*, *Swap out*)

- ◆ Auswahl wird dem Scheduler überlassen
- ◆ Deaktivierung wird lediglich von Speicherverwaltung angestoßen

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

E-Memory.fm 1999-12-14 13.02

E.86

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Ersetzungsstrategie bei Solaris (6)

■ Typische Werte

- ◆ *minfree*: 1/64 des Hauptspeichers (Solaris 2.2), 25 Seiten (Solaris 2.4)
- ◆ *desfree*: 1/32 des Hauptspeichers (Solaris 2.2), 50 Seiten (Solaris 2.4)
- ◆ *lotsfree*: 1/16 des Hauptspeichers (Solaris 2.2), 128 Seiten (Solaris 2.4)
- ◆ *deficit*: 0 bis *lotsfree*
- ◆ *fastscan*: min(1/4 Hauptspeicher, 64 MByte) pro Sekunde (Solaris 2.4)
- ◆ *slowscan*: 800 kBytes pro Sekunde (Solaris 2.4)
- ◆ *handspread*: wie *fastscan* (Solaris 2.4)

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

E-Memory.fm 1999-12-14 13.02

E.87

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

E.9 Zusammenfassung

■ Freispeicherverwaltung

- ◆ Speicherrepräsentation, Zuteilungsverfahren

■ Mehrprogrammbetrieb

- ◆ Relokation, Ein- und Auslagerung
- ◆ Segmentierung
- ◆ Seitenadressierung, Seitenadressierung und Segmentierung, TLB
- ◆ gemeinsamer Speicher

■ Virtueller Speicher

- ◆ Demand paging
- ◆ Seitenersetzungsstrategien: FIFO, B₀, LRU, 2nd chance (Clock)

■ Seitenflattern

- ◆ Super-Zustände, Arbeitsmengenmodell

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

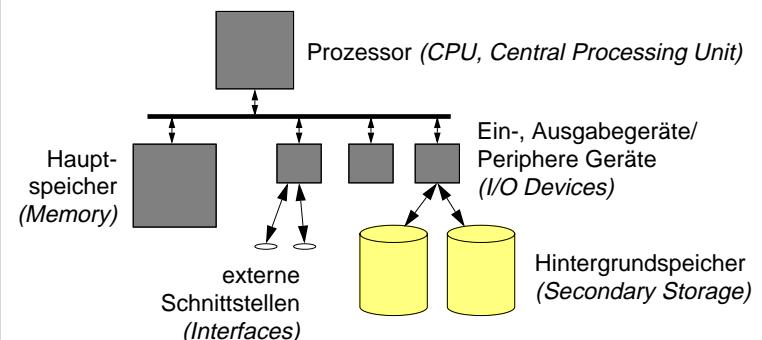
E-Memory.fm 1999-12-14 13.02

E.88

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

F Implementierung von Dateien

■ Einordnung



SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

F.1

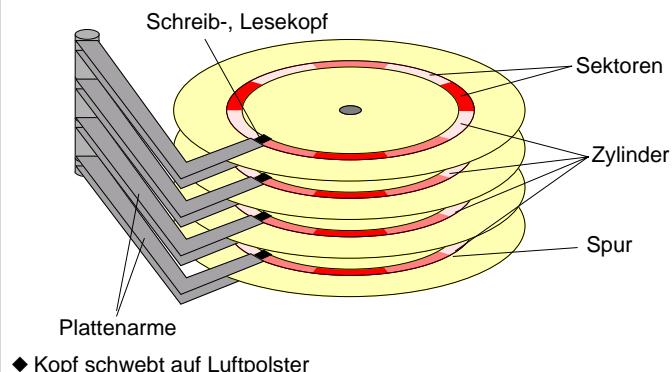
Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

F.1 Medien

1 Festplatten

■ Häufigstes Medium zum Speichern von Dateien

- ◆ Aufbau einer Festplatte



SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

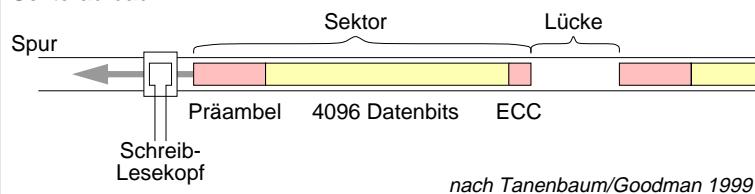
F-File.fm 1999-12-14 13.02

F.2

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Festplatten (2)

Sektaufbau



- ◆ Breite der Spur: 5–10 µm
- ◆ Spuren pro Zentimeter: 800–2000
- ◆ Breite einzelner Bits: 0,1–0,2 µm

Zonen

- ◆ Mehrere Zylinder (10–30) bilden eine Zone mit gleicher Sektorenanzahl (bessere Plattenausnutzung)

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

F.3

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Festplatten (3)

Datenblätter zweier Beispielplatten

Plattentyp	Seagate Medalist	Seagate Cheetah
Kapazität	10,2 GB	36,4 GB
Platten/Köpfe	3/6	12/24
Zylinderzahl	CHS 16383/16/83	9772
Cache	512 kB	4 MB
Positionierzeiten	Spur zu Spur	0,6/0,9 ms
	mittlere	9,5 ms
	maximale	12/13 ms
Transferrate	8,5 MB/s	18,3–28 MB/s
Rotationsgeschw.	5.400 U/min	10.000 U/min
eine Plattenumdrehung	11 ms	6 ms
Stromaufnahme	4,5 W	14 W

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

F.4

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Festplatten (4)

Zugriffsmerkmale

- ◆ blockorientierter und wahlfreier Zugriff
- ◆ Blockgröße zwischen 32 und 4096 Bytes (typisch 512 Bytes)
- ◆ Zugriff erfordert Positionierung des Schwenkarms auf den richtigen Zylinder und Warten auf den entsprechenden Sektor

Blöcke sind üblicherweise numeriert

- ◆ getrennte Numerierung: Zylindernummer, Sektornummer
- ◆ kombinierte Numerierung: durchgehende Nummern über alle Sektoren (Reihenfolge: aufsteigend innerhalb eines Zylinders, dann folgender Zylinder, etc.)

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

F.5

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Disketten

Ähnlicher Aufbau wie Festplatten

- ◆ maximal zwei Schreib-, Leseköpfe (oben, unten)
- ◆ Kopf berührt Diskettenoberfläche

Typische Daten

Diskettentyp	3,5" HD
Kapazität	1,44 MB
Köpfe	2
Spuren	80
Sektoren pro Spur	18
Transferrate	62,5 kB/s
Rotationsgeschw.	300 U/min
eine Umdrehung	200 ms

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

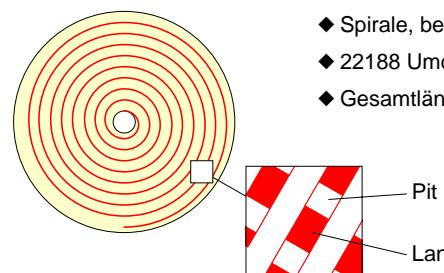
F-File.fm 1999-12-14 13.02

F.6

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 CD-ROM

■ Aufbau einer CD



- ◆ Spirale, beginnend im Inneren
- ◆ 22188 Umdrehungen (600 pro mm)
- ◆ Gesamtlänge 5,6 km

◆ **Pit:** Vertiefung, die von einem Laser abgetastet werden kann

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

F.7

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 CD-ROM (2)

■ Kodierung

- ◆ **Symbol:** ein Byte wird mit 14 Bits kodiert (kann bereits bis zu zwei Bitfehler korrigieren)
- ◆ **Frame:** 42 Symbole werden zusammengefaßt (192 Datenbits, 396 Fehlerkorrekturbits)
- ◆ **Sektor:** 98 Frames werden zusammengefaßt (16 Bytes Präambel, 2048 Datenbytes, 288 Bytes Fehlerkorrektur)
- ◆ **Effizienz:** 7203 Bytes transportieren 2048 Nutzbytes

■ Transferrate

- ◆ Single-Speed-Laufwerk:
75 Sektoren pro Sekunde (153.600 Bytes pro Sekunde)
- ◆ 40-fach-Laufwerk:
3000 Sektoren pro Sekunde (6.144.000 Bytes pro Sekunde)

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

F.8

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 CD-ROM (3)

■ Kapazität

- ◆ ca. 650 MB

■ Varianten

- ◆ **CD-R** (Recordable): einmal beschreibbar
- ◆ **CD-RW** (Rewritable): mehrfach beschreibbar

■ DVD (Digital Versatile Disk)

- ◆ kleinere Pits, engere Spirale, andere Laserlängenfarbe
- ◆ einseitig oder zweiseitig beschrieben
- ◆ ein- oder zweischichtig beschrieben
- ◆ Kapazität: 4,7 bis 17 GB

SP I Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

F.9

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

F.2 Speicherung von Dateien

- Dateien benötigen oft mehr als einen Block auf der Festplatte
 - ◆ Welche Blöcke werden für die Speicherung einer Datei verwendet?

1 Kontinuierliche Speicherung

- Datei wird in Blöcken mit aufsteigenden Blocknummern gespeichert
 - ◆ Zugriff auf alle Blöcke mit minimaler Positionierzeit des Schwenkarms
 - ◆ Einsatz z.B. bei Systemen mit Echtzeitanforderungen

▲ Probleme

- ◆ Finden des freien Platzes auf der Festplatte (Menge aufeinanderfolgender und freier Plattenblöcke)
- ◆ Fragmentierungsproblem (Verschnitt: nicht nutzbare Plattenblöcke; siehe auch Speicherverwaltung)

SP I Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

F.10

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Kontinuierliche Speicherung (2)

▲ Weiteres Problem

- ◆ Größe bei neuen Dateien oft nicht im voraus bekannt
- ◆ Erweitern ist problematisch
 - Umkopieren, falls kein freier angrenzender Block mehr verfügbar

■ Variation

- ◆ Unterteilen einer Datei in Folgen von Blocks (*Chunks, Extents*)
- ◆ Blockfolgen werden kontinuierlich gespeichert

▲ Problem

- ◆ Verschnitt innerhalb einer Folge

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

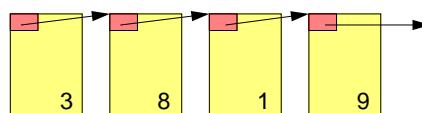
F-File.fm 1999-12-14 13.02

F.11

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Verkettete Speicherung

■ Blöcke einer Datei sind verkettet



- ◆ z.B. Commodore Systeme (CBM 64 etc.)

- Blockgröße 256 Bytes
- die ersten zwei Bytes bezeichnen Spur- und Sektornummer des nächsten Blocks
- wenn Spurnummer gleich Null: letzter Block
- 254 Bytes Nutzdaten

- ★ File kann wachsen und verlängert werden

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

F.12

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Verkettete Speicherung (2)

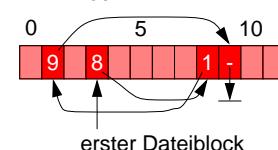
▲ Probleme

- ◆ Speicher für Verzeigerung geht von den Nutzdaten im Block ab (ungünstig im Zusammenhang mit Paging: Seite würde immer aus Teilen von zwei Plattenblöcken bestehen)
- ◆ Fehleranfälligkeit: Datei ist nicht restaurierbar, falls einmal Verzeigerung fehlerhaft

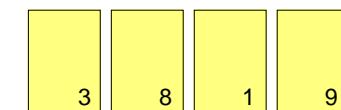
■ Verkettung wird in speziellen Plattenblocks gespeichert

- ◆ FAT-Ansatz (*FAT: File allocation table*), z.B. MS-DOS, Windows 95

FAT-Block



Blöcke der Datei: 3, 8, 1, 9



SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

F.13

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Verkette Speicherung (3)

★ Vorteile

- ◆ kompletter Inhalt des Datenblocks ist nutzbar (günstig bei Paging)
- ◆ mehrfache Speicherung der FAT möglich: Einschränkung der Fehleranfälligkeit

▲ Probleme

- ◆ mindestens ein zusätzlicher Block muss geladen werden (Caching der FAT zur Effizienzsteigerung nötig)
- ◆ FAT enthält Verkettungen für alle Dateien: das Laden der FAT-Blöcke lädt auch nicht benötigte Informationen
- ◆ aufwendige Suche nach dem zugehörigen Datenblock bei bekannter Position in der Datei

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

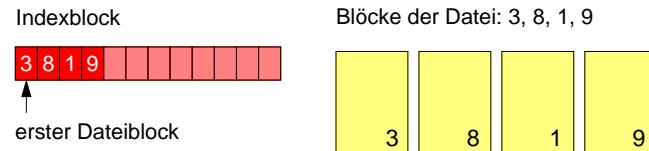
F-File.fm 1999-12-14 13.02

F.14

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Indiziertes Speichern

- Spezieller Plattenblock enthält Blocknummern der Datenblocks einer Datei



▲ Problem

- ◆ feste Anzahl von Blöcken
 - Verschnitt bei kleinen Dateien
 - Erweiterung nötig für große Dateien

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

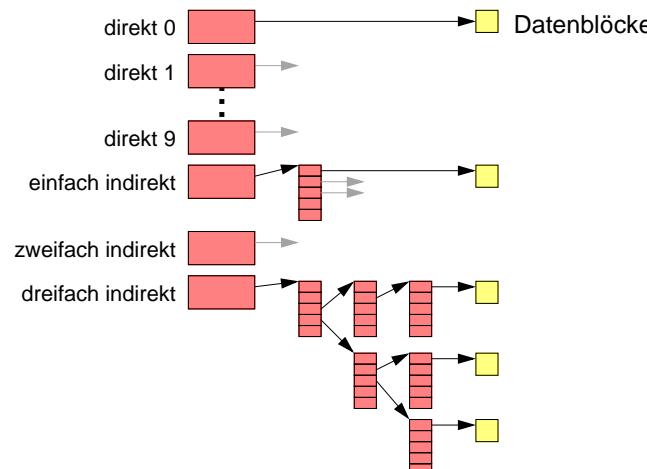
F-File.fm 1999-12-14 13.02

F.15

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Indiziertes Speichern (2)

- Beispiel UNIX Inode



SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

F.16

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Indiziertes Speichern (3)

- ★ Einsatz von mehreren Stufen der Indizierung
 - ◆ Inode benötigt sowieso einen Block auf der Platte (Verschnitt unproblematisch bei kleinen Dateien)
 - ◆ durch mehrere Stufen der Indizierung auch große Dateien adressierbar
- ▲ Nachteil
 - ◆ mehrere Blöcke müssen geladen werden (nur bei langen Dateien)

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

F.17

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Baumsequentielle Speicherung

- Satzorientierte Dateien
 - ◆ Schlüssel + Datensatz
 - ◆ effizientes Auffinden des Datensatz mit einem bekannten Schlüssel
 - ◆ Schlüsselmenge spärlich besetzt
 - ◆ häufiges Einfügen und Löschen von Datensätzen
- Einsatz von B-Bäumen zur Satzspeicherung
 - ◆ innerhalb von Datenbanksystemen
 - ◆ als Implementierung spezieller Dateitypen kommerzieller Betriebssysteme
 - z.B. VSAM-Dateien in MVS (*Virtual storage access method*)
 - z.B. NTFS Katalogimplementierung

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

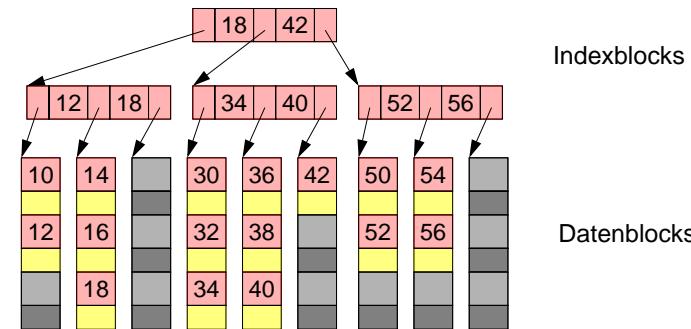
F-File.fm 1999-12-14 13.02

F.18

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Baumsequentielle Speicherung (2)

- Beispiel eines B*-Baums: Schlüssel sind Integer-Zahlen



- Blöcke enthalten Verweis auf nächste Ebene und den höchsten Schlüssel der nächsten Ebene
- Blocks der untersten Ebene enthalten Schlüssel und Sätze

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

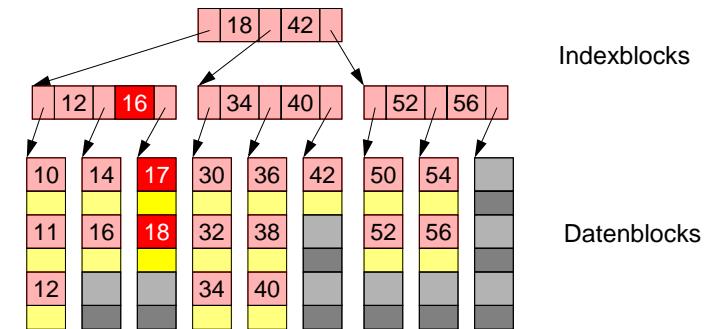
F-File.fm 1999-12-14 13.02

F.19

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Baumsequentielle Speicherung (4)

- Einfügen des Satzes mit Schlüssel „17“



- Satz mit Schlüssel „18“ wird verschoben (Indexblock wird angepasst)
- Satz mit Schlüssel „17“ in freien Platz eingefügt

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

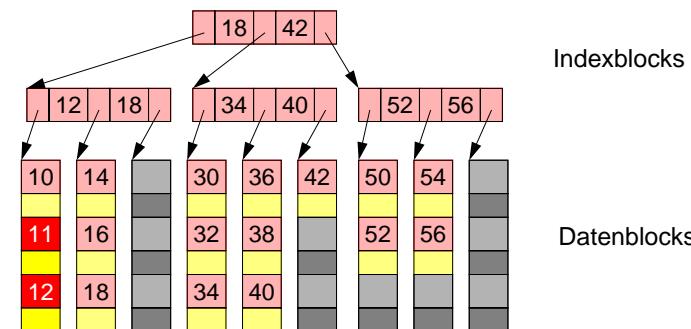
F-File.fm 1999-12-14 13.02

F.21

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Baumsequentielle Speicherung (3)

- Einfügen des Satzes mit Schlüssel „11“



- Satz mit Schlüssel „12“ wird verschoben
- Satz mit Schlüssel „11“ in freien Platz eingefügt

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

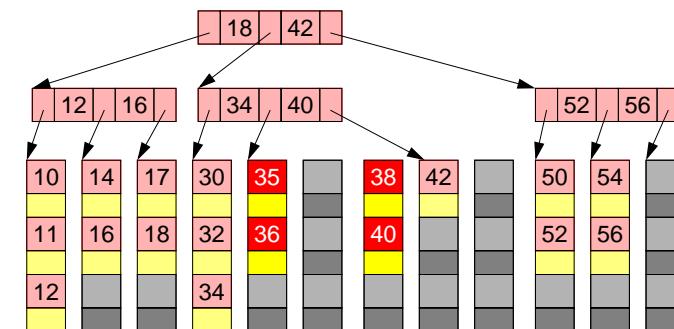
F-File.fm 1999-12-14 13.02

F.20

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Baumsequentielle Speicherung (5)

- Einfügen des Satzes mit Schlüssel „35“ (1. Schritt)



- Teilung des Blocks mit Satz „36“ und Einfügen des Satzes „35“
- Anfordern zweier weiterer, leerer Datenblöcke

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

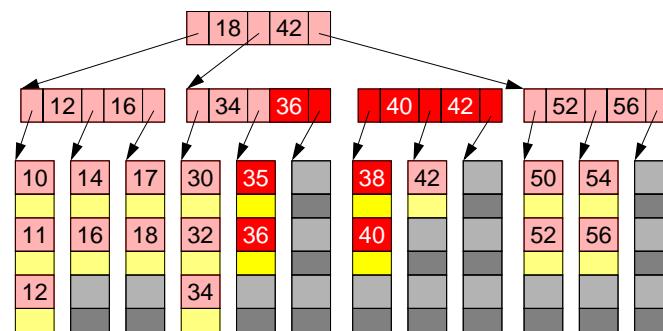
F-File.fm 1999-12-14 13.02

F.22

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Baumsequentielle Speicherung (6)

Einfügen des Satzes mit Schlüssel „35“ (2. Schritt)



- Teilung bzw. Erzeugung eines neuen Indexblocks und dessen Verzeigerung

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

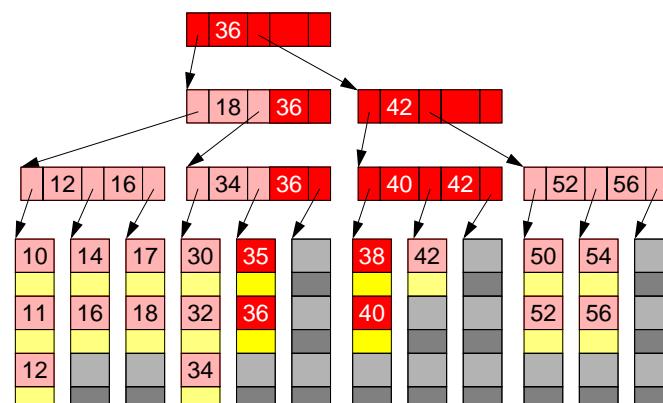
F-File.fm 1999-12-14 13.02

F.23

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Baumsequentielle Speicherung (7)

Einfügen des Satzes mit Schlüssel „35“ (3. Schritt)



- Spaltung des alten Wurzelknotens und Erzeugen eines neuen Wurzel

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

F.24

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Baumsequentielle Speicherung (8)

Effizientes Finden von Sätzen

- Baum ist sehr niedrig im Vergleich zur Menge der Sätze
 - viele Schlüssel pro Indexblock vorhanden (je nach Schlüssellänge)

Gutes Verhalten im Zusammenhang mit Paging

- jeder Block entspricht einer Seite
- Demand paging sorgt für das automatische Anhäufen der oberen Indexblocks im Hauptspeicher
 - schneller Zugriff auf die Indexstrukturen

Erlaubt nebenläufige Operationen durch geeignetes Sperren von Indexblöcken

Löschen erfolgt ähnlich wie Einfügen

- Verschmelzen von schlecht belegten Datenblöcken nötig

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

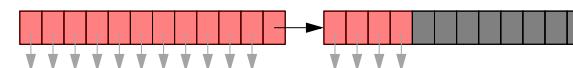
F.25

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

F.3 Freispeicherverwaltung

prinzipiell ähnlich wie Verwaltung von freiem Hauptspeicher

- Bitvektoren zeigen für jeden Block Belegung an
- verkettete Listen repräsentieren freie Blöcke
 - Verkettung kann in den freien Blöcken vorgenommen werden
 - Optimierung: aufeinanderfolgende Blöcke werden nicht einzeln aufgenommen, sondern als Stück verwaltet
 - Optimierung: ein freier Block enthält viele Blocknummern weiterer freier Blöcke und evtl. die Blocknummer eines weiteren Blocks mit den Nummern freier Blöcke



SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

F.26

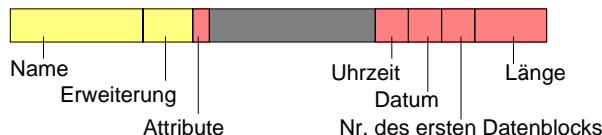
Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

F.4 Implementierung von Katalogen

1 Kataloge als Liste

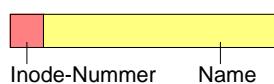
- Einträge gleicher Länge werden hintereinander in eine Liste gespeichert

◆ z.B. *FAT File systems*



◆ für VFAT werden mehrere Einträge zusammen verwendet, um den langen Namen aufzunehmen

◆ z.B. *UNIX System V.3*



SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

F.27

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Kataloge als Liste (2)

▲ Problem

- ◆ Lineare Suche durch die Liste nach bestimmtem Eintrag
- ◆ Sortierte Liste: binäre Suche, aber Sortieraufwand

2 Einsatz von Hashfunktionen

■ Hashing

- ◆ Spärlich besetzter Schlüsselraum wird auf einen anderen, meist dichter besetzten Schlüsselraum abgebildet
- ◆ Beispiel: Menge der möglichen Dateinamen wird nach $[0 - N-1]$ abgebildet ($N = \text{Länge der Katalogliste}$)

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

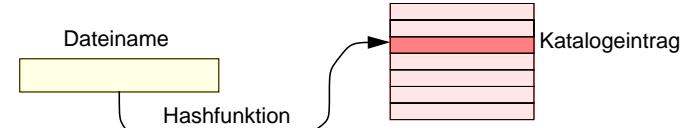
F.28

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Einsatz von Hashfunktionen (2)

■ Hashfunktion

- ◆ Funktion bildet Dateinamen auf einen Index in die Katalogliste ab schnellerer Zugriff auf den Eintrag möglich (kein lineares Suchen)
- ◆ (einfaches aber schlechtes) Beispiel: $(\sum \text{Zeichen}) \bmod N$



▲ Probleme

- ◆ Kollisionen (mehrere Dateinamen werden auf gleichen Eintrag abgebildet)
- ◆ Anpassung der Listengröße, wenn Liste voll

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

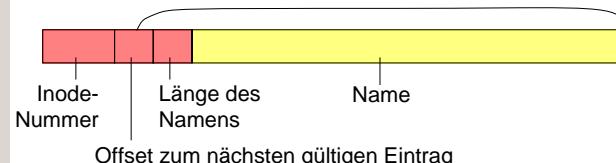
F-File.fm 1999-12-14 13.02

F.29

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Variabel lange Listenelemente

■ Beispiel *BSD 4.2, System V.4*, u.a.



▲ Probleme

- ◆ Verwaltung von freien Einträgen in der Liste
- ◆ Speicherverschnitt (Kompaktifizieren, etc.)

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

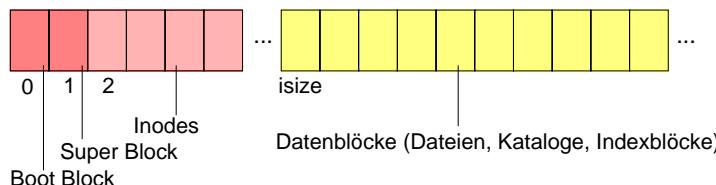
F.30

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

F.5 Beispiel: UNIX File Systems

1 System V File System

■ Blockorganisation



- ◆ Boot Block enthält Informationen zum Laden eines initialen Programms
- ◆ Super Block enthält Verwaltungsinformationen für ein Dateisystem
 - Anzahl der Blöcke, Anzahl der Inodes
 - Anzahl und Liste freier Blöcke
 - Anzahl und Liste freier Inodes
 - Attribute (z.B. *Modified flag*)

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

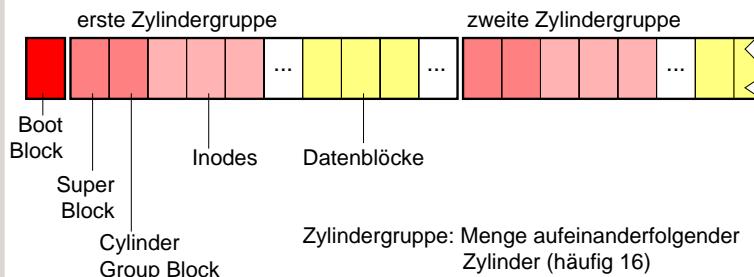
F-File.fm 1999-12-14 13.02

F.31

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 BSD 4.2 (Berkeley Fast File System)

■ Blockorganisation



- ◆ Kopie des Super Blocks in jeder Zylindergruppe
- ◆ freie Inodes und freie Datenblöcke werden im Cylinder group block gehalten
- ◆ eine Datei wird möglichst innerhalb einer Zylindergruppe gespeichert

- ★ Vorteil: kürzere Positionierungszeiten

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

F.32

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Block Buffer Cache

- Pufferspeicher für alle benötigten Plattenblocks
 - ◆ Verwaltung mit Algorithmen ähnlich wie bei Paging
 - ◆ *Read ahead*: beim sequentiellen Lesen wird auch der Transfer des Folgeblocks angestoßen
 - ◆ *Lazy write*: Block wird nicht sofort auf Platte geschrieben (erlaubt Optimierung der Schreibzugriffe und blockiert den Schreiber nicht)
 - ◆ Verwaltung freier Blöcke in einer Freiliste
 - Kandidaten für Freiliste werden nach LRU Verfahren bestimmt
 - bereits freie aber noch nicht anderweitig benutzte Blöcke können reaktiviert werden (*Reclaim*)

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

F.33

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Block Buffer Cache (2)

- Schreiben erfolgt, wenn
 - ◆ Datei geschlossen wird,
 - ◆ keine freien Puffer mehr vorhanden sind,
 - ◆ regelmäßig vom System (*fsflush* Prozess, *update* Prozess),
 - ◆ beim Systemaufruf *sync()*,
 - ◆ und nach jedem Schreibauftrag im Modus *O_SYNC*.
- Adressierung
 - ◆ Adressierung eines Blocks erfolgt über ein Tupel: (Gerätenummer, Blocknummer)
 - ◆ Über die Adresse wird ein Hashwert gebildet, der eine der möglichen Pufferliste auswählt

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

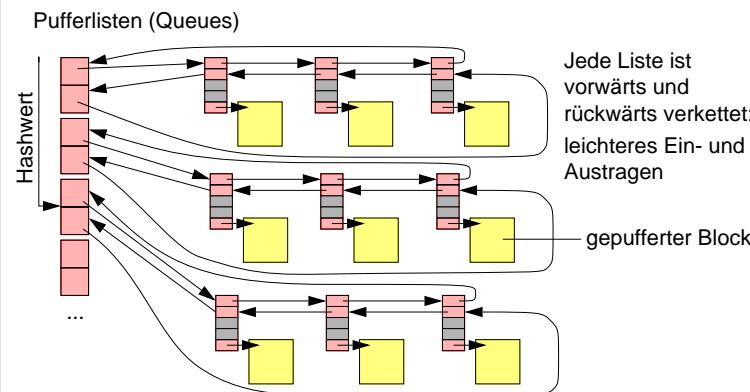
F-File.fm 1999-12-14 13.02

F.34

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Block Buffer Cache (3)

Aufbau des Block buffer cache



SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

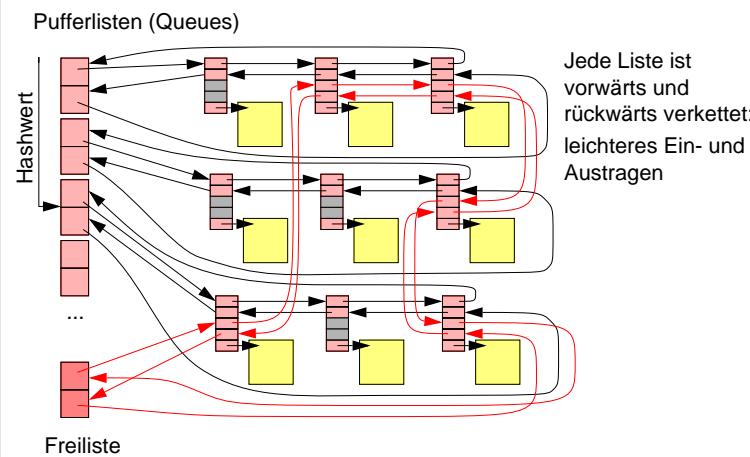
F-File.fm 1999-12-14 13.02

F.35

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Block Buffer Cache (4)

Aufbau des Block buffer cache



SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

F.36

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Block Buffer Cache (5)

Block Buffer Cache teilweise obsolet durch moderne Pageing-Systeme

- ◆ Kacheln des Hauptspeichers ersetzen den Block Buffer Cache
- ◆ Kacheln können Seiten aus einem Adressraum und/oder Seiten aus einer Datei beherbergen

Problem

- ◆ Kopieren großer Dateien führt zum Auslagern noch benötigter Adressraumseiten

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

F.37

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Systemaufrufe

Bestimmen der Kachelgröße

```
int getpagesize( void );
```

Abbildung von Dateien in den virtuellen Adressraum

- ◆ Einblenden einer Datei

```
caddr_t mmap( caddr_t addr, size_t len, int prot, int flags,  
              int fd, off_t off );
```

- Einblenden an bestimmte oder beliebige Adresse
- lesbar, schreibbar, ausführbar

- ◆ Ausblenden einer Datei

```
int munmap( caddr_t addr, size_t len );
```

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

F.38

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Systemaufrufe (2)

- ◆ Kontrolloperation

```
int mctl( caddr_t addr, size_t len, int func, void *arg );
```

- zum Ausnehmen von Seiten aus dem Paging (Fixieren im Hauptspeicher)
- zum Synchronisieren mit der Datei

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

F.39

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

F.6 Beispiel: Windows NT (NTFS)

- File System für Windows NT
- Datei
 - ◆ einfache, unstrukturierte Folge von Bytes
 - ◆ beliebiger Inhalt; für das Betriebssystem ist der Inhalt transparent
 - ◆ dynamisch erweiterbare Dateien
 - ◆ Rechte verknüpft mit NT Benutzern und Gruppen
 - ◆ Datei kann automatisch komprimiert abgespeichert werden
 - ◆ große Dateien bis zu 8.589.934.592 Gigabytes lang
 - ◆ Hard links: mehrere Einträge derselben Datei in verschiedenen Katalogen möglich

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

F.40

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

F.6 Beispiel: NTFS (2)

- Katalog
 - ◆ baumförmig strukturiert
 - Knoten des Baums sind Kataloge
 - Blätter des Baums sind Dateien
 - ◆ Rechte wie bei Dateien
 - ◆ alle Dateien des Katalogs automatisch komprimierbar
- Partitionen heißen Volumes
 - ◆ Volume wird (in der Regel) durch einen Laufwerksbuchstaben dargestellt z.B. c:

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

F.41

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Rechte

- Eines der folgenden Rechte pro Benutzer oder Benutzergruppe
 - ◆ *no access*: kein Zugriff
 - ◆ *list*: Anzeige von Dateien in Katalogen
 - ◆ *read*: Inhalt von Dateien lesen und *list*
 - ◆ *add*: Hinzufügen von Dateien zu einem Katalog und *list*
 - ◆ *read&add*: wie *read* und *add*
 - ◆ *change*: Ändern von Dateiinhalten, Löschen von Dateien und *read&add*
 - ◆ *full*: Ändern von Eigentümer und Zugriffsrechten und *change*

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

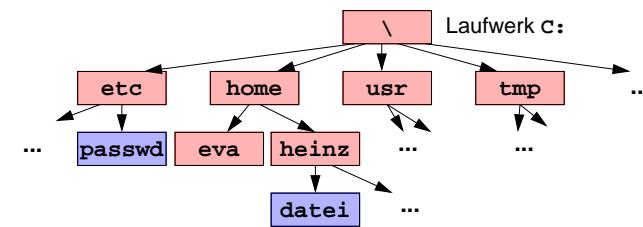
F-File.fm 1999-12-14 13.02

F.42

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Pfadnamen

■ Baumstruktur



■ Pfade

- ◆ wie unter FAT-Filesystem
- ◆ z.B. „c:\home\heinz\datei“, „\tmp“, „C:..\heinz\datei“

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

F.43

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Pfadnamen (2)

■ Namenskonvention

- ◆ 255 Zeichen inklusive Sonderzeichen
(z.B. „Eigene Programme“)
- ◆ automatischer Kompatibilitätsmodus: 8 Zeichen Name, 3 Zeichen Erweiterung, falls „langer Name“ unter MS-DOS ungültig
(z.B. AUTOEXEC.BAT)

■ Kataloge

- ◆ Jeder Katalog enthält einen Verweis auf sich selbst („..“) und einen Verweis auf den darüberliegenden Katalog im Baum („...“)
- ◆ Hard links aber keine symbolischen Namen direkt im NTFS

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

F.44

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Dateiverwaltung

■ Basiseinheit „Cluster“

- ◆ 512 Bytes bis 4 Kilobytes (beim Formatieren festgelegt)
- ◆ wird auf eine Menge von hintereinanderfolgenden Blöcken abgebildet
- ◆ logische Cluster-Nummer als Adresse (LCN)

■ Basiseinheit „Strom“

- ◆ jede Datei kann mehrere (Daten-)Ströme speichern
- ◆ einer der Ströme wird für die eigentlichen Daten verwendet
- ◆ Dateiname, MS-DOS Dateiname, Zugriffsrechte, Attribute und Zeitstempel werden jeweils in eigenen Datenströmen gespeichert
(leichte Erweiterbarkeit des Systems)

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

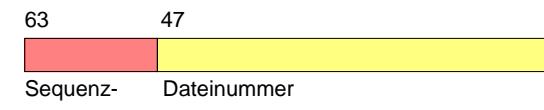
F.45

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Dateiverwaltung (2)

■ File-Reference

- ◆ Bezeichnet eindeutig eine Datei oder einen Katalog



- Dateinummer ist Index in eine globale Tabelle (*MFT: Master File Table*)
- Sequenznummer wird hochgezählt, für jede neue Datei mit gleicher Dateinummer

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

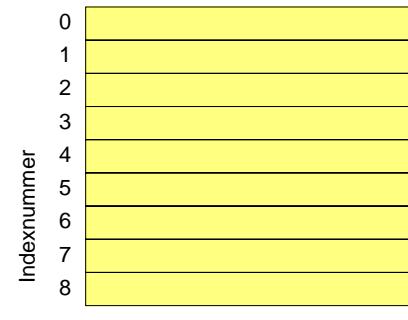
F.46

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Master-File-Table

Rückgrat des gesamten Systems

- ◆ große Tabelle mit gleich langen Elementen
(1KB, 2KB oder 4KB groß, je nach Clustergröße)



- ◆ Index in die Tabelle ist Teil der *File-Reference*

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

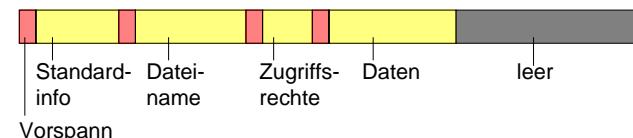
F-File.fm 1999-12-14 13.02

F.47

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Master-File-Table (2)

Eintrag für eine kurze Datei



Ströme

- ◆ Standard Information (immer in der MFT)
 - enthält Länge, MS-DOS Attribute, Zeitstempel, Anzahl der Hard links, Sequenznummer der gültigen File-Reference
- ◆ Dateiname (immer in der MFT)
 - kann mehrfach vorkommen (Hard links, MS-DOS Name)
- ◆ Zugriffsrechte (*Security Descriptor*)
- ◆ Eigentliche Daten

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

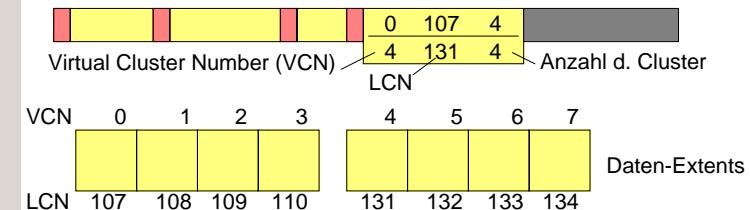
F-File.fm 1999-12-14 13.02

F.48

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Master-File-Table (3)

Eintrag für eine längere Datei



- ◆ Extents werden außerhalb der MFT in aufeinanderfolgenden Clustern gespeichert
- ◆ Lokalisierungsinformationen werden in einem eigenen Strom gespeichert

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

F.49

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Master-File-Table (4)

Mögliche weitere Ströme (*Attributes*)

- ◆ Index
 - Index über einen Attribut schlüssel (z.B. Dateinamen) implementiert Katalog
- ◆ Indexbelegungstabelle
 - Belegung der Struktur eines Index
- ◆ Attributliste (immer in der MFT)
 - wird benötigt, falls nicht alle Ströme in einen MFT Eintrag passen
 - referenzieren weitere MFT Einträge und deren Inhalt

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

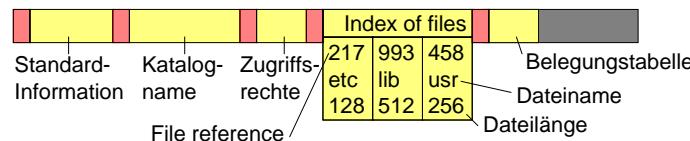
F-File.fm 1999-12-14 13.02

F.50

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Master File Table (3)

■ Eintrag für einen kurzen Katalog



- ◆ Dateien des Katalogs werden mit File references benannt
- ◆ Name und Länge der im Katalog enthaltenen Dateien und Kataloge werden auch im Index gespeichert
(doppelter Aufwand beim Update; schnellerer Zugriff beim Kataloglisten)

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

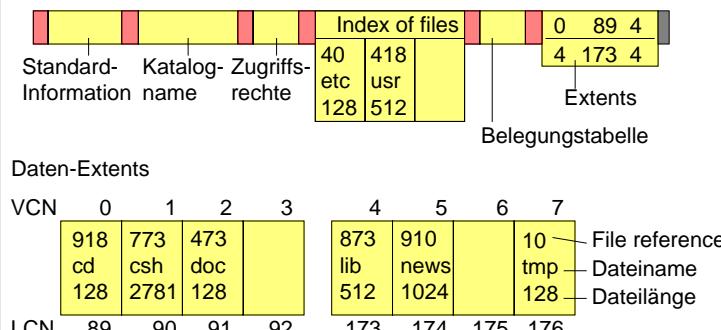
F-File.fm 1999-12-14 13.02

F.51

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Master File Table (4)

■ Eintrag für einen längeren Katalog



- ◆ Speicherung als B+-Baum (sortiert, schneller Zugriff)
- ◆ in einen Cluster passen zwischen 3 und 15 Dateien (im Bild nur eine)

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

F.52

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

5 Metadaten

■ Alle Metadaten werden in Dateien gehalten

0	MFT
1	MFT Kopie (teilweise)
2	Log File
3	Volume Information
4	Attributabelle
5	Wurzelkatalog
6	Clusterbelegungstabelle
7	Boot File
8	Bad Cluster File
...	
16	Benutzerdateien u. -kataloge
17	
...	

Feste Dateien in der MFT

SP I Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

F.53

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

5 Metadaten (2)

■ Bedeutung der Metadateien

- ◆ MFT und MFT Kopie: MFT wird selbst als Datei gehalten (d.h. Cluster der MFT stehen im Eintrag 0)
MFT Kopie enthält die ersten 16 Einträge der MFT (Fehlertoleranz)
- ◆ Log File: enthält protokollierte Änderungen am Dateisystem
- ◆ Volume Information: Name, Größe und ähnliche Attribute des Volumes
- ◆ Attributabelle: definiert mögliche Ströme in den Einträgen
- ◆ Wurzelkatalog
- ◆ Clusterbelegungstabelle: Bitmap für jeden Cluster des Volumes
- ◆ Boot File: enthält initiales Programm zum Laden, sowie ersten Cluster der MFT
- ◆ Bad Cluster File: enthält alle nicht lesbaren Cluster der Platte
NTFS markiert automatisch alle schlechten Cluster und versucht die Daten in einen anderen Cluster zu retten

SP I Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

F.54

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

6 Fehlererholung

- Mögliche Fehler
 - ◆ Stromausfall (dummer Benutzer schaltet einfach Rechner aus)
 - ◆ Systemabsturz
- Auswirkungen auf das Dateisystem
 - ◆ inkonsistente Metadaten
 - z.B. Katalogeintrag fehlt zur Datei oder umgekehrt
 - z.B. Block ist benutzt aber nicht als belegt markiert
 - Programme wie `chkdsk` oder `fsck` können inkonsistente Metadaten reparieren
 - Datenverluste möglich
 - ◆ unvollständige Daten in Dateien

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

F.55

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

6 Fehlererholung (2)

- ★ Log-Structured File-System
 - ◆ Alle Änderungen treten als Teil von Transaktionen auf.
 - ◆ Beispiele für Transaktionen:
 - Erzeugen, löschen, erweitern, verkürzen von Dateien
 - Dateiattribute verändern
 - Datei umbenennen
 - ◆ Protokollieren aller Änderungen am Dateisystem zusätzlich in einer Protokolldatei (*Log File*)
 - ◆ Beim Bootvorgang wird Protokolldatei mit den aktuellen Änderungen abgeglichen und damit werden Inkonsistenzen vermieden.

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

F.56

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

6 Fehlererholung (3)

- Protokollierung
 - ◆ Für jeden Einzelvorgang einer Transaktion wird zunächst ein Logeintrag erzeugt und
 - ◆ danach die Änderung am Dateisystem vorgenommen.
- ◆ Dabei gilt:
 - Der Logeintrag wird immer vor der eigentlichen Änderung auf Platte geschrieben.
 - Wurde etwas auf Platte geändert, steht auch der Protokolleintrag dazu auf der Platte.

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

F.57

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

6 Fehlererholung (4)

- Eigentliche Fehlererholung
 - ◆ Beim Bootvorgang wird überprüft, ob die protokollierten Änderungen vorhanden sind:
 - Transaktion kann wiederholt bzw. abgeschlossen werden (*Redo*) falls alle Logeinträge vorhanden.
 - Angefangene aber nicht beendete Transaktionen werden rückgängig gemacht (*Undo*).

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

F.58

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

6 Fehlererholung (5)

- Beispiel: Löschen einer Datei
 - ◆ Vorgänge der Transaktion
 - Beginn der Transaktion
 - Freigeben der Extents durch Löschen der entsprechenden Bits in der Belegungstabelle (gesetzte Bits kennzeichnen belegten Cluster)
 - Freigeben des MFT Eintrags der Datei
 - Löschen des Katalogeintrags der Datei (evtl. Freigeben eines Extents aus dem Index)
 - Ende der Transaktion
 - ◆ Alle Vorgänge werden unter der File-Reference im Log-File protokolliert, danach jeweils durchgeführt.
 - Protokolleinträge enthalten Informationen zum *Redo* und zum *Undo*

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

F.59

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

6 Fehlererholung (6)

- ◆ Log vollständig (Ende der Transaktion wurde protokolliert und steht auf Platte):
 - *Redo* der Transaktion:
alle Operationen werden wiederholt, falls nötig
- ◆ Log unvollständig (Ende der Transaktion steht nicht auf Platte):
 - *Undo* der Transaktion:
in umgekehrter Reihenfolge werden alle Operation rückgängig gemacht
- Checkpoints
 - ◆ Log-File ist nicht unendlich groß
 - ◆ gelegentlich wird für einen konsistenten Zustand auf Platte gesorgt (*Checkpoint*) und dieser Zustand protokolliert (alle Protokolleinträge von vorher können gelöscht werden)
 - ◆ Ähnlich verfährt NTFS, wenn Ende des Log-Files erreicht wird

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

F.60

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

6 Fehlererholung (7)

- ★ Ergebnis
 - ◆ eine Transaktion ist entweder vollständig durchgeführt oder gar nicht
 - ◆ Benutzer kann ebenfalls Transaktionen über mehrere Dateizugriffe definieren (?)
 - ◆ keine inkonsistente Metadaten möglich
 - ◆ Hochfahren eines abgestürzten Systems benötigt nur den relativ kurzen Durchgang durch das Log-File
 - Alternative `chkdsk` benötigt viel Zeit bei großen Platten
- ▲ Nachteile
 - ◆ etwas ineffizienter
 - ◆ nur für Volumes >400 MB geeignet

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

F.61

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

7 Limitierung der Plattenutzung

- Mehrbenutzersysteme
 - ◆ einzelnen Benutzern sollen verschiedene Kontingente zur Verfügung stehen
 - ◆ gegenseitige Beeinflussung soll vermieden werden (*Disk-full* Fehlermeldung)
- Quota-Systeme (Quantensysteme)
 - ◆ Tabelle enthält maximale und augenblickliche Anzahl von Blöcken für die Dateien und Kataloge eines Benutzers
 - ◆ Tabelle steht auf Platte und wird vom File-System fortgeschrieben
 - ◆ Benutzer erhält Disk-full Meldung, wenn sein Quota verbraucht ist
 - ◆ üblicherweise gibt es eine weiche und eine harte Grenze (weiche Grenze kann für eine bestimmte Zeit überschritten werden)

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

F.62

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

F.8 Fehlerhafte Plattenblöcke

- Blöcke, die beim Lesen Fehlermeldungen erzeugen
 - ◆ z.B. Prüfsummenfehler
- Hardwarelösung
 - ◆ Platte und Plattencontroller bemerken selbst fehlerhafte Blöcke und maskieren diese aus
 - ◆ Zugriff auf den Block wird vom Controller automatisch auf einen „gesunden“ Block umgeleitet
- Softwarelösung
 - ◆ File-System bemerkt fehlerhafte Blöcke und markiert diese auch als belegt

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

F.63

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

F.9 Datensicherung

- Schutz vor dem Totalausfall von Platten
 - ◆ z.B. durch Head-Crash oder andere Fehler
- Sichern der Daten auf Tertiärspeicher
 - ◆ Bänder
 - ◆ WORM Speicherplatten (*Write Once Read Many*)
- Sichern großer Datenbestände
 - ◆ Total-Backups benötigen lange Zeit
 - ◆ Inkrementelle Backups sichern nur Änderungen ab einem bestimmten Zeitpunkt
 - ◆ Mischen von Total-Backups mit inkrementellen Backups

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

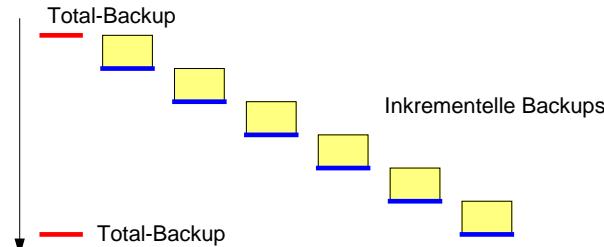
F-File.fm 1999-12-14 13.02

F.64

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Beispiele für Backup Scheduling

- Gestaffelte inkrementelle Backups



- ◆ z.B. alle Woche ein Total-Backup und jeden Tag ein inkrementelles Backup zum Vortag: maximal 7 Backups müssen eingespielt werden

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

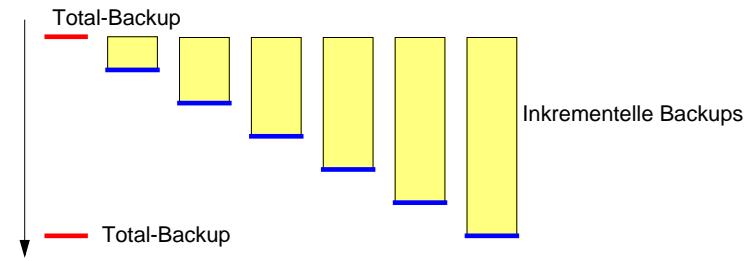
F-File.fm 1999-12-14 13.02

F.65

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Beispiele für Backup Scheduling (2)

- Gestaffelte inkrementelle Backups zum letzten Total-Backup



- ◆ z.B. alle Woche ein Total-Backup und jeden Tag ein inkrementelles Backup zum letzten Total-Backup: maximal 2 Backups müssen eingespielt werden

- Hierarchie von Backup-Läufen

- ◆ mehrstufige inkrementelle Backups zum Backup der nächst höheren Stufe
- ◆ optimiert Archivmaterial und Restaurierungszeit

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

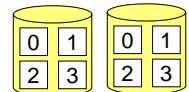
F.66

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Einsatz mehrere redundanter Platten

■ Gespiegelte Platten (*Mirroring; RAID 0*)

- ◆ Daten werden auf zwei Platten gleichzeitig gespeichert



- ◆ Implementierung durch Software (File-System, Plattentreiber) oder Hardware (spez. Controller)
- ◆ eine Platte kann ausfallen
- ◆ schnelleres Lesen (da zwei Platten unabhängig voneinander beauftragt werden können)

▲ Nachteil

- ◆ doppelter Speicherbedarf
- ◆ wenig langsameres Schreiben durch Warten auf zwei Plattentransfers

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

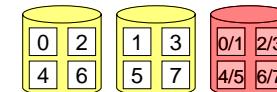
F.67

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Einsatz mehrere redundanter Platten (3)

■ Paritätsplatte (RAID 4)

- ◆ Daten werden über mehrere Platten gespeichert, eine Platte enthält Parität



- ◆ Paritätsblock enthält byteweise XOR-Verknüpfungen von den zugehörigen Blöcken aus den anderen Streifen
- ◆ eine Platte kann ausfallen
- ◆ schnelles Lesen
- ◆ prinzipiell beliebige Plattenanzahl (ab drei)

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

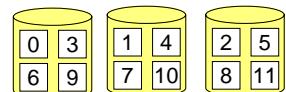
F.69

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Einsatz mehrere redundanter Platten (2)

■ Gestreifte Platten (*Striping; RAID 1*)

- ◆ Daten werden über mehrere Platten gespeichert



- ◆ Datentransfers sind nun schneller, da mehrere Platten gleichzeitig angesprochen werden können

▲ Nachteil

- ◆ keinerlei Datensicherung: Ausfall einer Platte lässt Gesamtsystem ausfallen

■ Verknüpfung von RAID 0 und 1 möglich (RAID 0+1)

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

F.68

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Einsatz mehrerer redundanter Platten (4)

▲ Nachteil von RAID 4

- ◆ jeder Schreibvorgang erfordert auch das Schreiben des Paritätsblocks
- ◆ Erzeugung des Paritätsblocks durch Speichern des vorherigen Blockinhalts möglich: $P_{\text{neu}} = P_{\text{alt}} \oplus B_{\text{alt}} \oplus B_{\text{neu}}$ (P =Parity, B =Block)
- ◆ Schreiben eines kompletten Streifens benötigt nur einmaliges Schreiben des Paritätsblocks
- ◆ Paritätsplatte ist hoch belastet (meist nur sinnvoll mit SSD [Solid state disk])

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

F-File.fm 1999-12-14 13.02

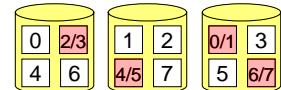
F.70

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Einsatz mehrere redundanter Platten (5)

■ Verstreuter Paritätsblock (RAID 5)

- ◆ Paritätsblock wird über alle Platten verstreut



- ◆ zusätzliche Belastung durch Schreiben des Paritätsblocks wird auf alle Platten verteilt
- ◆ heute gängigstes Verfahren redundanter Platten
- ◆ Vor- und Nachteile wie RAID 4

▲ Problem

- ◆ fehlerhafter Paritätsblock zerstört mehrere Blöcke, falls eine Platte ausfällt

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

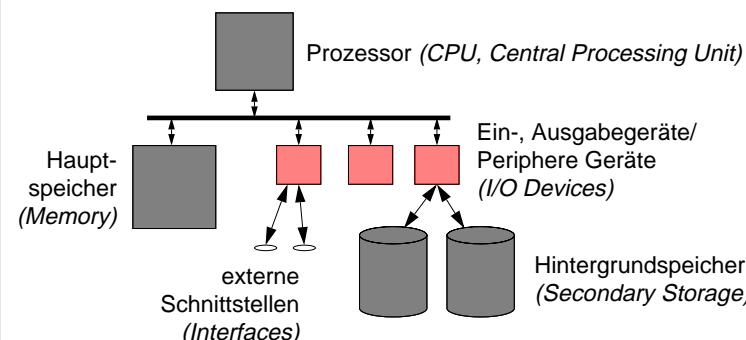
F-File.fm 1999-12-14 13.02

F.71

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

G Ein- und Ausgabe

■ Einordnung



SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

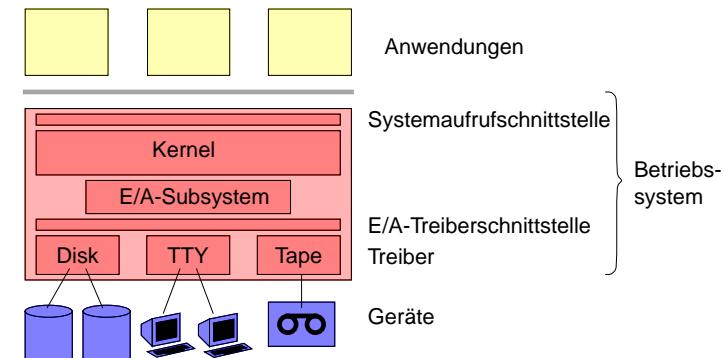
G-InOut.fm 1999-12-13 14.43

G.1

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

G.1 Gerätzugang und Treiber

■ Schichtung der Systemsoftware bis zum Gerät



SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

G-InOut.fm 1999-12-13 14.43

G.2

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Gerätrepräsentation in UNIX

■ Periphere Geräte werden als Spezialdateien repräsentiert

- ◆ Geräte können wie Dateien mit Lese- und Schreiboperationen angesprochen werden
- ◆ Öffnen der Spezialdateien schafft eine Verbindung zum Gerät, die durch einen Treiber hergestellt wird
- ◆ direkter Durchgriff vom Anwender auf den Treiber

■ Blockorientierte Spezialdateien

- ◆ Plattenlaufwerke, Bandlaufwerke, Floppy Disks, CD-ROMs

■ Zeichenorientierte Spezialdateien

- ◆ Serielle Schnittstellen, Drucker, Audiokanäle etc.
- ◆ blockorientierte Geräte haben meist auch eine zusätzliche zeichenorientierte Repräsentation

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

G-InOut.fm 1999-12-13 14.43

G.3

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Gerätrepräsentation in UNIX (2)

- Eindeutige Beschreibung der Geräte durch ein Tupel:
(Gerätetyp, Major number, Minor number)
 - ◆ Gerätetyp: Block device, Character device
 - ◆ Major number: Auswahlnummer für einen Treiber
 - ◆ Minor number: Auswahl eines Gerätes innerhalb eines Treibers

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

G-InOut.fm 1999-12-13 14.43

G.4

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Gerätrepräsentation in UNIX (3)

- Beispiel eines Kataloglisting von `/dev` (Ausschnitt)

```
crw----- 1 fzhauck 108, 0 Oct 16 1996 audio
crw----- 1 fzhauck 108,128 Oct 16 1996 audioctl
crw-rw-rw- 1 root    21, 0 May  3 1996 conslog
brw-rw-rw- 1 root    36, 2 Oct 16 1996 fd0
crw----- 1 fzhauck 17, 0 Oct 16 1996 mouse
crw-rw-rw- 1 root    13, 2 Jan 13 09:09 null
crw-rw-rw- 1 root    36, 2 Jul  2 1997 rfd0
crw-r---- 1 root    32, 0 Oct 16 1996 rsd3a
crw-r---- 1 root    32, 1 Oct 16 1996 rsd3b
crw-r---- 1 root    32, 2 Oct 16 1996 rsd3c
brw-r---- 1 root    32, 0 Oct 16 1996 sd3a
brw-r---- 1 root    32, 1 Oct 16 1996 sd3b
brw-r---- 1 root    32, 2 Oct 16 1996 sd3c
crw-rw-rw- 1 root    22, 0 Sep 19 09:11 tty
crw-rw-rw- 1 root    29, 0 Oct 16 1996 ttya
crw-rw-rw- 1 root    29, 1 Oct 16 1996 ttyb
```

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

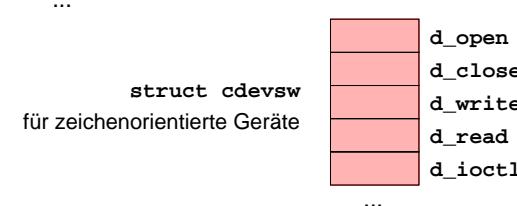
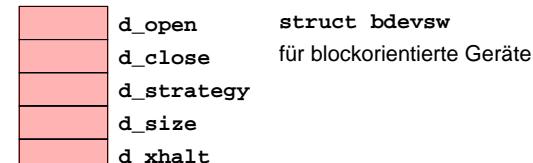
G-InOut.fm 1999-12-13 14.43

G.5

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Gerätrepräsentation in UNIX (4)

- Interne Treiberschnittstelle
 - ◆ Vektor von Funktionszeigern pro Treiber (Major number):



SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

G-InOut.fm 1999-12-13 14.43

G.6

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Gerätrepräsentation in UNIX (5)

- Funktionen eines Block device-Treibers

- ◆ **d_open:** Öffnen des Gerätes
- ◆ **d_close:** Schließen des Gerätes
- ◆ **d_strategy:** Abgeben von Lese- und Schreibaufträgen auf Blockbasis
- ◆ **d_size:** Ermitteln der Gerätegröße (z.B. Partitions- oder Plattengröße)
- ◆ **d_xhalt:** Abschalten des Gerätes
- ◆ u.a.

- Funktionen eines Character device-Treibers

- ◆ **d_open, d_close:** Öffnen und Schließen des Gerätes
- ◆ **d_read, d_write:** Lesen und Schreiben von Zeichen
- ◆ **d_ioctl:** generische Kontrolloperation
- ◆ u.a.

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

G-InOut.fm 1999-12-13 14.43

G.5

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

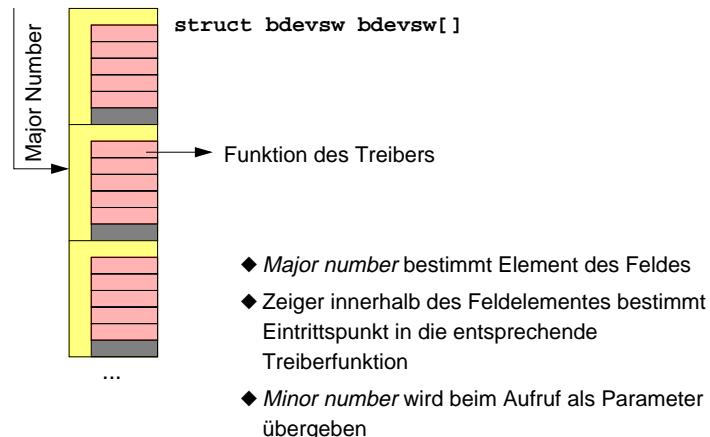
G-InOut.fm 1999-12-13 14.43

G.7

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Gerätrepräsentation in UNIX (6)

Felder für den Aufruf von Treibern (`bdevsw[]` und `cdevsw[]`)



SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

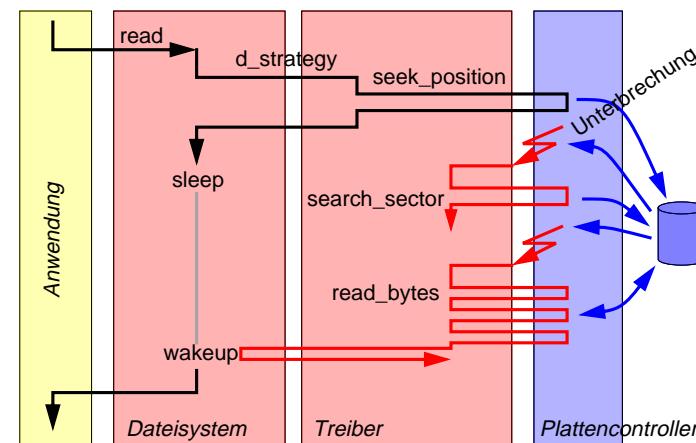
G-InOut.fm 1999-12-13 14.43

G.8

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Einfacher Treiber

Ablauf eines Leseaufrufs



SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

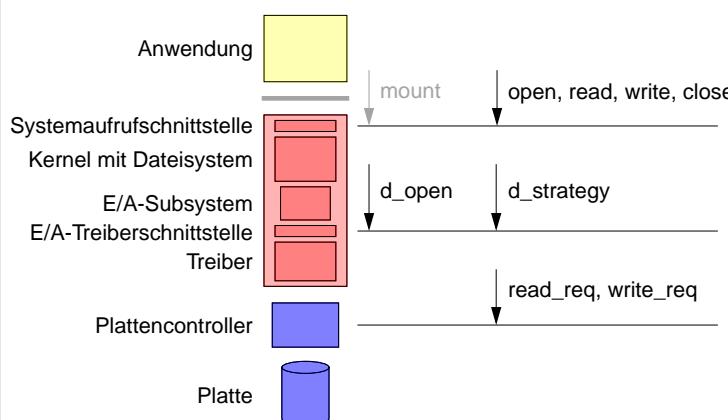
G-InOut.fm 1999-12-13 14.43

G.10

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

G.2 Plattentreiber

Software und Hardware zwischen Anwender und Platte



SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

G-InOut.fm 1999-12-13 14.43

G.9

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Einfacher Plattentreiber (2)

- ◆ Anwendung führt `read()` Systemaufruf aus.
- ◆ Dateisystem prüft, ob entsprechender Block im Speicher vorhanden.
- ◆ Falls der Block nicht vorhanden ist, wird ein Speicherplatz bereitgestellt und `d_strategy` im entsprechenden Treiber aufgerufen.
- ◆ Die Ausführung von `d_strategy` stößt Plattenpositionierung an.
- ◆ Die Anwendung blockiert sich im Kern. System kann andere Prozesse ablaufen lassen.
- ◆ Plattencontroller meldet sich bei erfolgreicher Positionierung durch eine Unterbrechung.
- ◆ Unterbrechungsbehandlung stößt Sektorschreibe an.
- ◆ In erneuter Unterbrechung nach gefundenem Sektor werden die Daten im Pollingbetrieb eingelesen.
- ◆ Schließlich wird der Anwendungsprozess wieder aufgeweckt (in den Zustand bereit überführt).

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

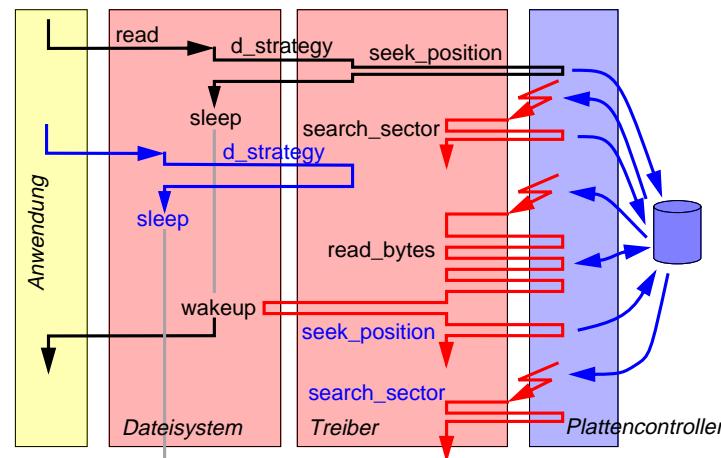
G-InOut.fm 1999-12-13 14.43

G.11

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Einfacher Plattentreiber (3)

Ablauf mehrerer Leseaufrufe



SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

G-InOut.fm 1999-12-13 14.43

G.12

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Treiber mit DMA

DMA (Direct Memory Access) erlaubt Einlesen und Schreiben ohne Prozessorbeteiligung

- ◆ DMA Controller erhält verschiedene Parameter:
 - die Hauptspeicheradresse zum Abspeichern bzw. Auslesen eines Plattenblocks
 - die Adresse des Plattencontrollers zum Abholen bzw. Abgeben der Daten
 - die Länge der zu transferierenden Daten
 - ◆ DMA Controller löst bei Fertigstellung eine Unterbrechung aus
- ★ Vorteile
- ◆ Prozessor muss Zeichen eines Plattenblocks nicht selbst abnehmen (kein Polling sondern Interrupt)
 - ◆ Plattentransferzeit kann zum Ablauf anderer Prozesse genutzt werden

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

G-InOut.fm 1999-12-13 14.43

G.14

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Einfacher Plattentreiber

- Unterbrechungsbehandlung ist auch für weitere Aufträge zuständig
 - ◆ Ist der Auftrag abgeschlossen, muss die Unterbrechungsbehandlung den nächsten Auftrag auswählen und aufsetzen, da der zugehörige Prozess bereits blockiert ist.
 - ◆ Die Unterbrechungen laufender Aufträge sorgen für die Abwicklung der folgenden Aufträge.

SP I

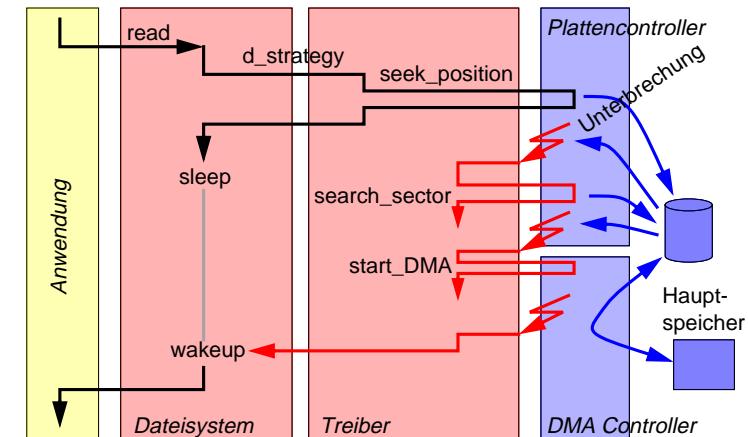
Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

G-InOut.fm 1999-12-13 14.43

G.13

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Treiber mit DMA (2)



SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD 4, 1997-2000

G-InOut.fm 1999-12-13 14.43

G.15

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.