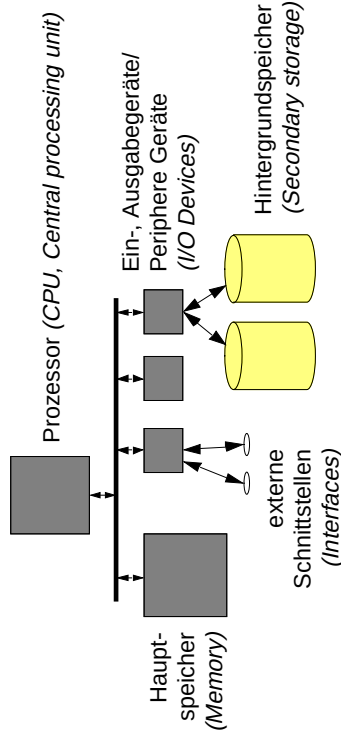


F Implementierung von Dateien

■ Einordnung



SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998

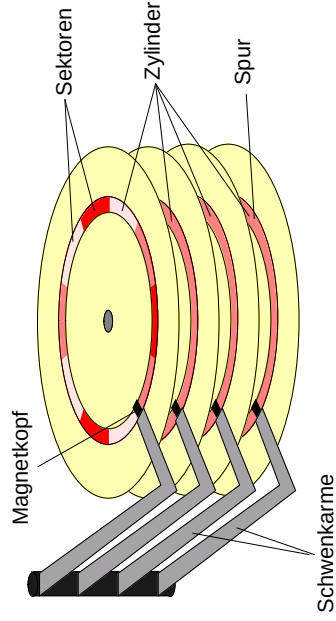
F-File.doc: 1998-01-05 17:08

F.1

Reproduktion oder andere Verwendung dieser Unterlagen, selbst in Teilversionen, ist ohne schriftliche Genehmigung der Universität Erlangen-Nürnberg, IMMD IV, 1998.

F.1 Festplatten

- Häufigstes Medium zum Speichern von Dateien
- ◆ Floppy Disks sind in der Handhabung ähnlich
- Aufbau einer Festplatte



SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998

F-File.doc: 1998-01-05 17:08

F.2

Reproduktion oder andere Verwendung dieser Unterlagen, selbst in Teilversionen, ist ohne schriftliche Genehmigung der Universität Erlangen-Nürnberg, IMMD IV, 1998.

F.1 Festplatten

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

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998

F-File.doc: 1998-01-05 17:08

F.3

Reproduktion oder andere Verwendung dieser Unterlagen, selbst in Teilversionen, ist ohne schriftliche Genehmigung der Universität Erlangen-Nürnberg, IMMD IV, 1998.

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, Universität Erlangen-Nürnberg, IMMD IV, 1998

F-File.doc: 1998-01-05 17:08

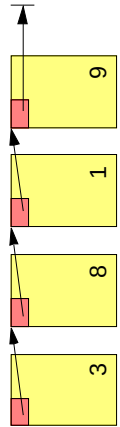
F.4

Reproduktion oder andere Verwendung dieser Unterlagen, selbst in Teilversionen, ist ohne schriftliche Genehmigung der Universität Erlangen-Nürnberg, IMMD IV, 1998.

1 Kontinuierliche Speicherung (2)

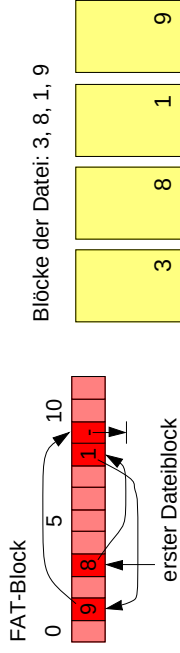
- ▲ Weiteres Problem
 - ◆ Größe bei neuen Dateien oft nicht im voraus bekannt
 - ◆ Erweitern ist problematisch
 - Umkopieren, falls kein freier angrenzender Block mehr verfügbar
- Variation
 - ◆ Unterteilen einer Datei in Folgen von Blocks (*Chunks, Extents*)
 - ◆ Blockfolgen werden kontinuierlich gespeichert
- ▲ Problem
 - ◆ Verschnitt innerhalb einer Folge

2 Verkettete Speicherung

- Blöcke einer Datei sind verkettet
 
- ◆ z.B. Commodore Systeme (CBM 64 etc.)
 - Blockgröße 256 Bytes
 - die ersten zwei Bytes bezeichnen Zylinder- und Sektornummer des nächsten Blocks
 - wenn Zylindernummer gleich Null: letzter Block
 - 254 Bytes Nutzdaten
- ★ File kann wachsen und verlängert werden

2 Verkettete Speicherung (2)

- ▲ Probleme
 - ◆ Speicher für Verzeigerung geht von den Nutzdaten im Block ab (ungünstig im Zusammenhang mit Paging: Seite besteht immer aus Teilen von zwei Plattenblöcken)
 - ◆ 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

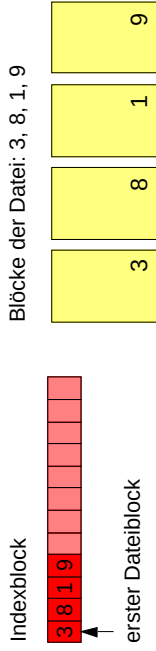


2 Verkettete Speicherung (3)

- ★ 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

3 Indiziertes Speichern

- Spezieller Plattenblock enthält Blocknummern der Datenblocks einer Datei



- ▲ Problem
 - ◆ feste Anzahl von Blöcken
 - Verschnitt bei kleinen Dateien
 - Erweiterung nötig bei großen Dateien

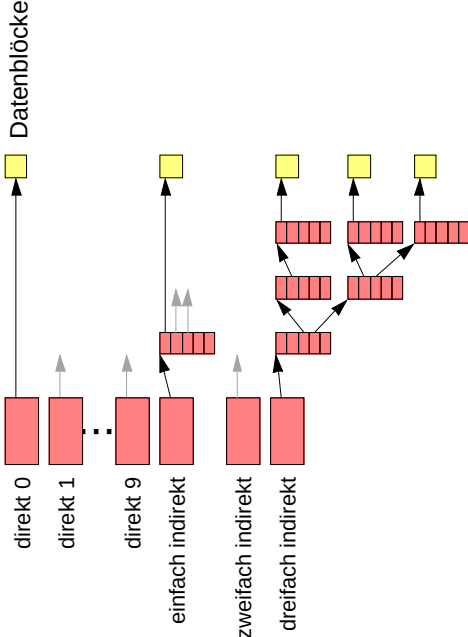
SP I

F.9

Reproduzieren oder Abdruck ist ohne schriftliche Genehmigung der Universität Erlangen-Nürnberg. Jede Art der Vervielfältigung ist strafbar.

3 Indiziertes Speichern (2)

- Beispiel UNIX Inode



SP I

F.10

Reproduzieren oder Abdruck ist ohne schriftliche Genehmigung der Universität Erlangen-Nürnberg. Jede Art der Vervielfältigung ist strafbar.

3 Indizierte Speicherung

- ★ 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

F.11

Reproduzieren oder Abdruck ist ohne schriftliche Genehmigung der Universität Erlangen-Nürnberg. Jede Art der Vervielfältigung ist strafbar.

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

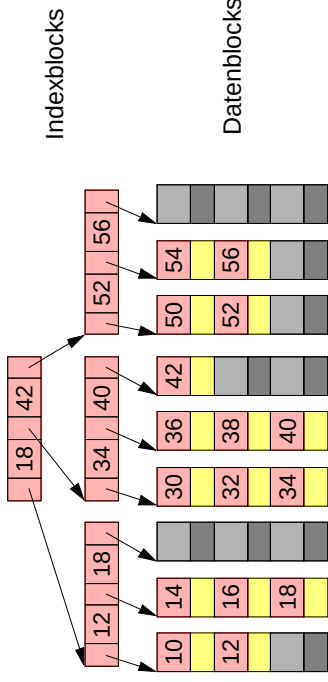
SP I

F.12

Reproduzieren oder Abdruck ist ohne schriftliche Genehmigung der Universität Erlangen-Nürnberg. Jede Art der Vervielfältigung ist strafbar.

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, Universität Erlangen-Nürnberg, IMMD IV, 1998

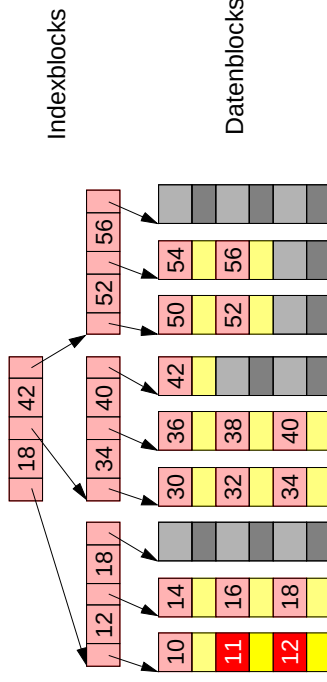
F-File.doc (1998-01-05 17:08)

F.13

Reproduktion oder die Verbreitung dieser Unterlagen, selbst in Teilform, ist ohne schriftliche Genehmigung der Universität Erlangen-Nürnberg strafbar.

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, Universität Erlangen-Nürnberg, IMMD IV, 1998

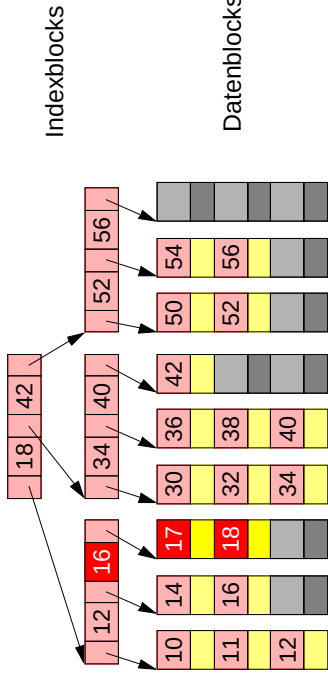
F-File.doc (1998-01-05 17:08)

F.14

Reproduktion oder die Verbreitung dieser Unterlagen, selbst in Teilform, ist ohne schriftliche Genehmigung der Universität Erlangen-Nürnberg strafbar.

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, Universität Erlangen-Nürnberg, IMMD IV, 1998

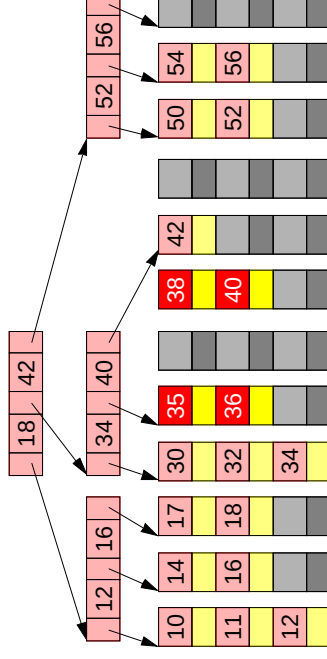
F-File.doc (1998-01-05 17:08)

F.15

Reproduktion oder die Verbreitung dieser Unterlagen, selbst in Teilform, ist ohne schriftliche Genehmigung der Universität Erlangen-Nürnberg strafbar.

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, Universität Erlangen-Nürnberg, IMMD IV, 1998

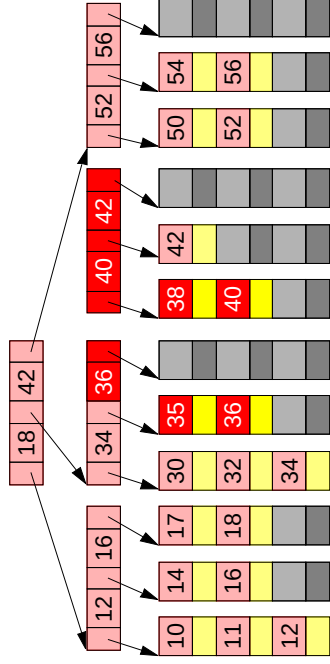
F-File.doc (1998-01-05 17:08)

F.16

Reproduktion oder die Verbreitung dieser Unterlagen, selbst in Teilform, ist ohne schriftliche Genehmigung der Universität Erlangen-Nürnberg strafbar.

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, Universität Erlangen-Nürnberg, IMMD IV, 1998

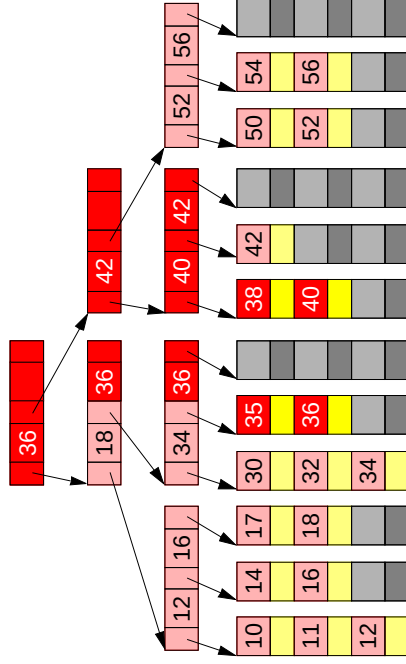
F-File.doc (1998-01-05 17:08)

F.17

Reproduktion oder andere Verwendung dieser Unterlagen ohne in jeder Hinsicht an die Universität Erlangen-Nürnberg schriftlich die 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, Universität Erlangen-Nürnberg, IMMD IV, 1998

F-File.doc (1998-01-05 17:08)

F.18

Reproduktion oder andere Verwendung dieser Unterlagen ohne in jeder Hinsicht an die Universität Erlangen-Nürnberg schriftlich die 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, Universität Erlangen-Nürnberg, IMMD IV, 1998

F-File.doc (1998-01-05 17:08)

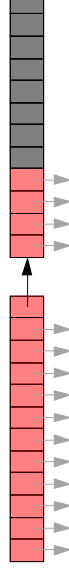
F.19

Reproduktion oder andere Verwendung dieser Unterlagen ohne in jeder Hinsicht an die Universität Erlangen-Nürnberg schriftlich die 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, Universität Erlangen-Nürnberg, IMMD IV, 1998

F-File.doc (1998-01-05 17:08)

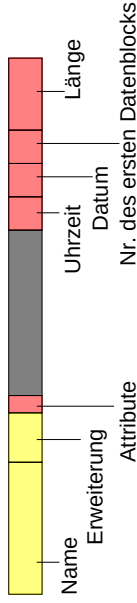
F.20

Reproduktion oder andere Verwendung dieser Unterlagen ohne in jeder Hinsicht an die Universität Erlangen-Nürnberg schriftlich die Zustimmung des Autors.

F.4 Implementierung von Katalogen

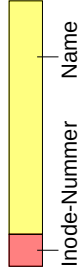
- Liste gleichgroßer Elemente

- ◆ Einträge werden hintereinander in eine Liste gespeichert
z.B. *FAT File systems*



- für *Protected mode FAT* werden mehrere Einträge zusammen verwendet, um den langen Namen aufzunehmen

- ◆ z.B. UNIX System V.3



Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998

F-File.doc: 1998-01-05 17:08

Reproduzieren Sie diese Vorlesung ohne Weiteres in der Universität Erlangen-Nürnberg. Jegliche Art der Zerstörung des Dokuments ist strafbar.

F.21

F.4 Implementierung von Katalogen

- Hashfunktion

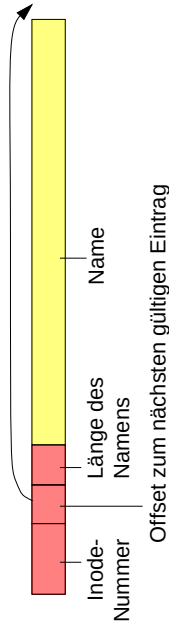
- ◆ Funktion bildet Dateinamen auf einen Index in der Liste ab
(Hashing: spärlich besetzter Schlüsselraum wird auf einen anderen, meist dichter besetzten Schlüsselraum abgebildet)

- ◆ schnellerer Zugriff auf den Eintrag möglich (kein lineares Suchen)

- ◆ Probleme: Kollisionen, Anpassung der Listengröße

- Liste variabel langer Elemente

- ◆ z.B. BSD 4.2, System V.4, u.a.



Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998

F-File.doc: 1998-01-05 17:08

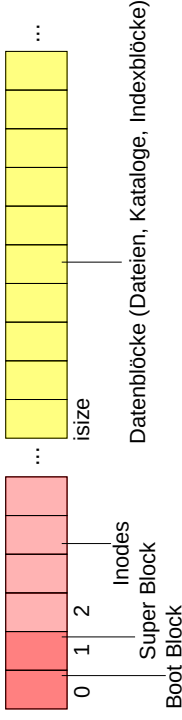
Reproduzieren Sie diese Vorlesung ohne Weiteres in der Universität Erlangen-Nürnberg. Jegliche Art der Zerstörung des Dokuments ist strafbar.

F.22

F.5 Beispiel: UNIX File Systems

1 System V File System

- Blockorganisation



- ◆ Boot Block enthält Informationen zum Laden eines initialen Programms
- ◆ Super Block enthält Verwaltungsinformation für ein Dateisystem

- Anzahl der Blöcke, Anzahl der Inodes
- Anzahl und Liste freier Blöcke
- Anzahl und Liste freier Inodes
- Attribute (z.B. *Modified flag*)

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998

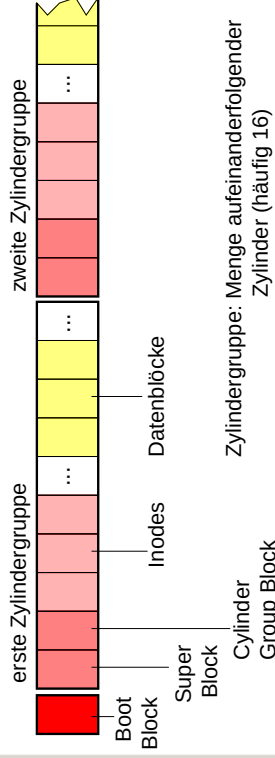
F-File.doc: 1998-01-05 17:08

Reproduzieren Sie diese Vorlesung ohne Weiteres in der Universität Erlangen-Nürnberg. Jegliche Art der Zerstörung des Dokuments ist strafbar.

F.23

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

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998

F-File.doc: 1998-01-05 17:08

Reproduzieren Sie diese Vorlesung ohne Weiteres in der Universität Erlangen-Nürnberg. Jegliche Art der Zerstörung des Dokuments ist strafbar.

F.24

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, Universität Erlangen-Nürnberg, IMMD IV, 1998

F-File.doc: 1998-01-05 17:08

F.25

Reproduction of this article for other purposes without the written consent of the author is prohibited.

3 Block Buffer Cache (2)

- Schreiben erfolgt, wenn
 - ◆ Datei geschlossen wird,
 - ◆ keine freien Puffer mehr vorhanden sind,
 - ◆ regelmäßig vom System (*flush* Prozeß, *update* Prozeß),
 - ◆ beim Systemaufruf *sync()*,
 - ◆ und nach jedem Schreibaufruf 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, Universität Erlangen-Nürnberg, IMMD IV, 1998

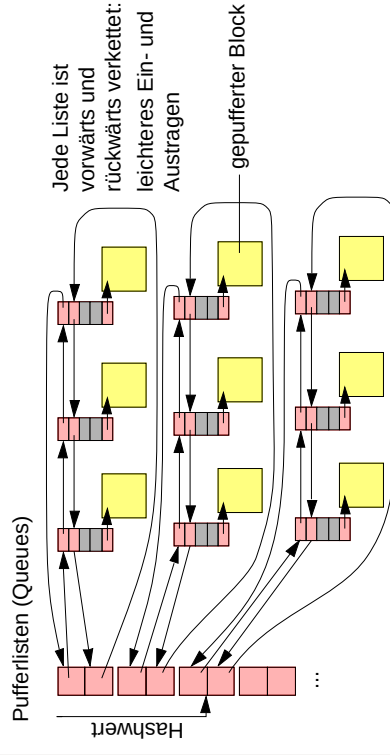
F-File.doc: 1998-01-05 17:08

F.26

Reproduction of this article for other purposes without the written consent of the author is prohibited.

3 Block Buffer Cache (3)

- Aufbau des Block buffer cache



SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998

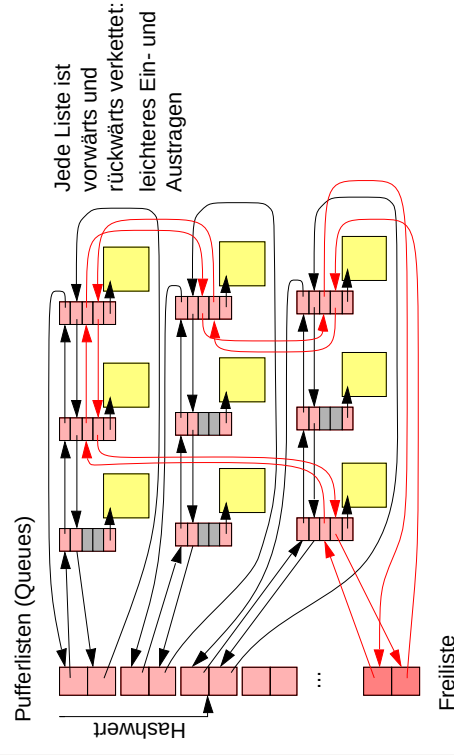
F-File.doc: 1998-01-05 17:08

F.27

Reproduction of this article for other purposes without the written consent of the author is prohibited.

3 Block Buffer Cache (4)

- Aufbau des Block buffer cache



SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998

F-File.doc: 1998-01-05 17:08

F.28

Reproduction of this article for other purposes without the written consent of the author is prohibited.

4 Systemaufrufe

- Bestimmen der Kachelgröße
`int getpagesize( void );`
- Abbildung von Dateien in den virtuellen Adresraum
 - ◆ 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, Universität Erlangen-Nürnberg, IMMD IV, 1998

F-File.doc (1998-01-05 17:08)

F.29

Reproduzieren oder Anzulegen ist ohne Verwendung dieser Vorlesung nicht zulässig. Dieser Vorlesung ist die Universität Erlangen-Nürnberg vorbehalten.

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, Universität Erlangen-Nürnberg, IMMD IV, 1998

F-File.doc (1998-01-05 17:08)

F.30

Reproduzieren oder Anzulegen ist ohne Verwendung dieser Vorlesung nicht zulässig. Dieser Vorlesung ist die Universität Erlangen-Nürnberg vorbehalten.

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 erweiterbar
 - ◆ Rechte verknüpft mit NT Benutzern und Gruppen
 - *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 Eigentümern und Zugriffsrechten und *change*

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998

F-File.doc (1998-01-05 17:08)

F.31

Reproduzieren oder Anzulegen ist ohne Verwendung dieser Vorlesung nicht zulässig. Dieser Vorlesung ist die Universität Erlangen-Nürnberg vorbehalten.

F.6 Beispiel: NTFS (2)

- ◆ 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
- 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 durch einen Laufwerksbuchstaben dargestellt
z.B. **C:**

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998

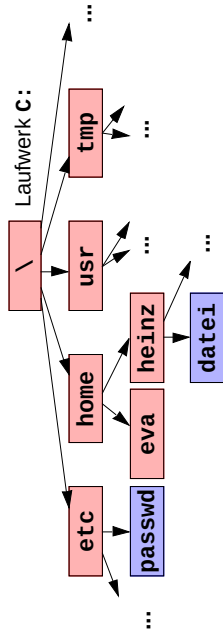
F-File.doc (1998-01-05 17:08)

F.32

Reproduzieren oder Anzulegen ist ohne Verwendung dieser Vorlesung nicht zulässig. Dieser Vorlesung ist die Universität Erlangen-Nürnberg vorbehalten.

1 Pfadnamen

Baumstruktur



Pfade

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

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998

F-File.doc: 1998-01-05 17:08

F.33

Reproduzieren oder Verändern dieser Unterlagen ohne in jeder Hinsicht an der Universität Erlangen-Nürnberg beauftragte Zustimmung des Autors.

1 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 („..“)
(Ausnahme Wurzelkatalog)
- ◆ Hard links aber keine symbolischen Namen direkt im NTFS

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998

F-File.doc: 1998-01-05 17:08

F.34

Reproduzieren oder Verändern dieser Unterlagen ohne in jeder Hinsicht an der Universität Erlangen-Nürnberg beauftragte Zustimmung des Autors.

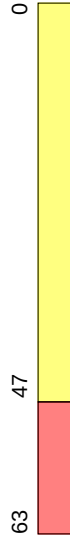
2 Dateiverwaltung

Basiseinheit Cluster

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

Basiseinheit Datei

- ◆ adressiert durch eine *File reference*



Sequenz-
nummer Dateinummer

- 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, Universität Erlangen-Nürnberg, IMMD IV, 1998

F-File.doc: 1998-01-05 17:08

F.35

Reproduzieren oder Verändern dieser Unterlagen ohne in jeder Hinsicht an der Universität Erlangen-Nürnberg beauftragte Zustimmung des Autors.

2 Dateiverwaltung (2)

Ströme

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

Master file table

- ◆ große Tabelle mit gleich langen Elementen (1KB, 2KB oder 4KB groß, je nach Clustergröße)
- ◆ Index in die Tabelle ist Teil der *File reference*
- ◆ entsprechender Eintrag für eine *File reference* enthält Informationen über bzw. die Ströme der Datei

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998

F-File.doc: 1998-01-05 17:08

F.36

Reproduzieren oder Verändern dieser Unterlagen ohne in jeder Hinsicht an der Universität Erlangen-Nürnberg beauftragte Zustimmung des Autors.