

Systemprogrammierung

Dateisystem

3. Februar 2011
9. Februar 2011

Überblick

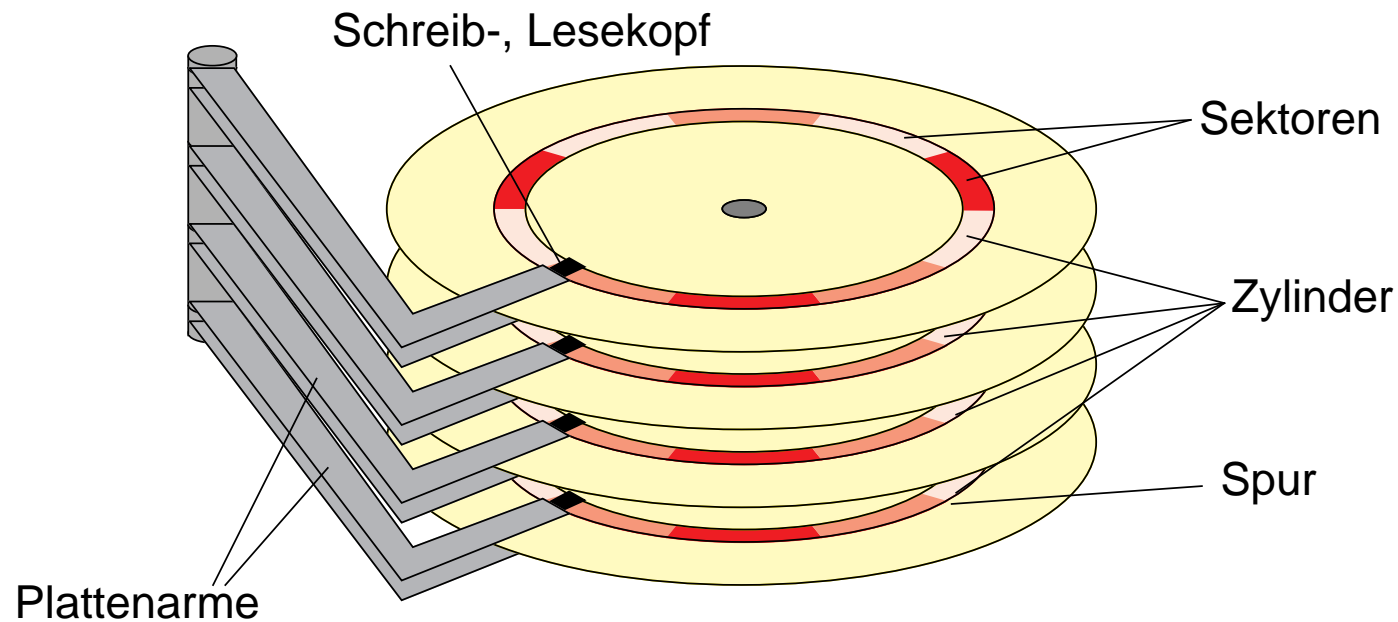
- Medien
- Speicherung von Dateien
- Freispeicherverwaltung
- Beispiele: Dateisysteme unter UNIX und Windows
- Dateisysteme mit Fehlererholung
- Datensicherung

Medien

2.1 Festplatten

- Häufigstes Medium zum Speichern von Dateien

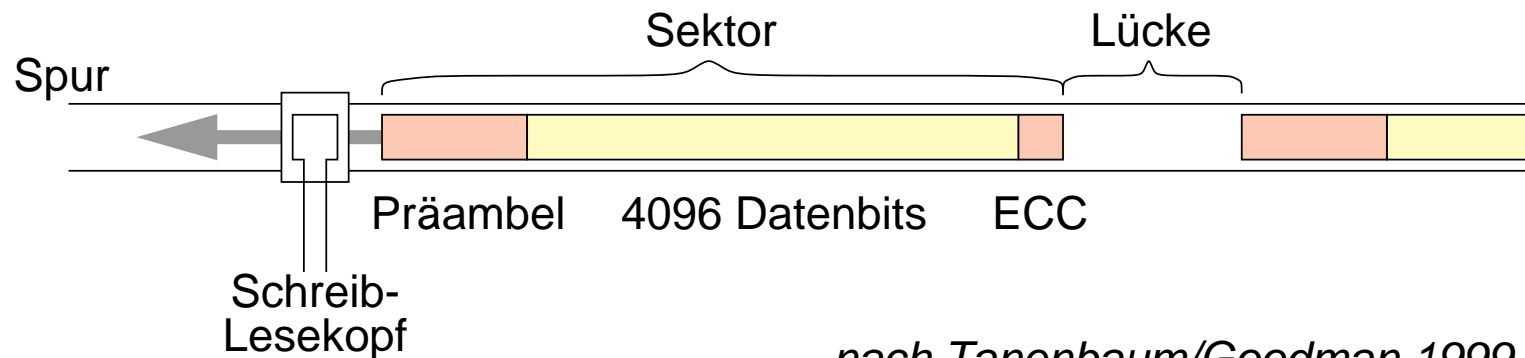
- ◆ Aufbau einer Festplatte



- ◆ Kopf schwebt auf Luftpolster

2.1 Festplatten (2)

■ Sektoraufbau



nach Tanenbaum/Goodman 1999

- ◆ 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)

2.1 Festplatten (3)

■ Datenblätter von drei Beispielplatten

Plattentyp		Fujitsu M2344 (1987)	Seagate Cheetah	Seagate Barracuda
Kapazität		690 MB	300 GB	400 GB
Platten/Köpfe		8 / 28	4 / 8	781.422.768 Sektoren
Zylinderzahl		624	90.774	
Cache		-	4 MB	8 MB
Positionier- zeiten	Spur zu Spur	4 ms	0,5 ms	-
	mittlere	16 ms	5,3 ms	8 ms
	maximale	33 ms	10,3 ms	-
Transferrate		2,4 MB/s	320 MB/s	-150 MB/s
Rotationsgeschw.		3.600 U/min	10.000 U/min	7.200 U/min
eine Plattenumdrehung		16 ms	6 ms	8 ms
Stromaufnahme		?	16-18 W	12,8 W

Januar 2011: Kapazität bis 3 TB, bis 15.000 U/min, Transferrate bis 1,6 GB/s

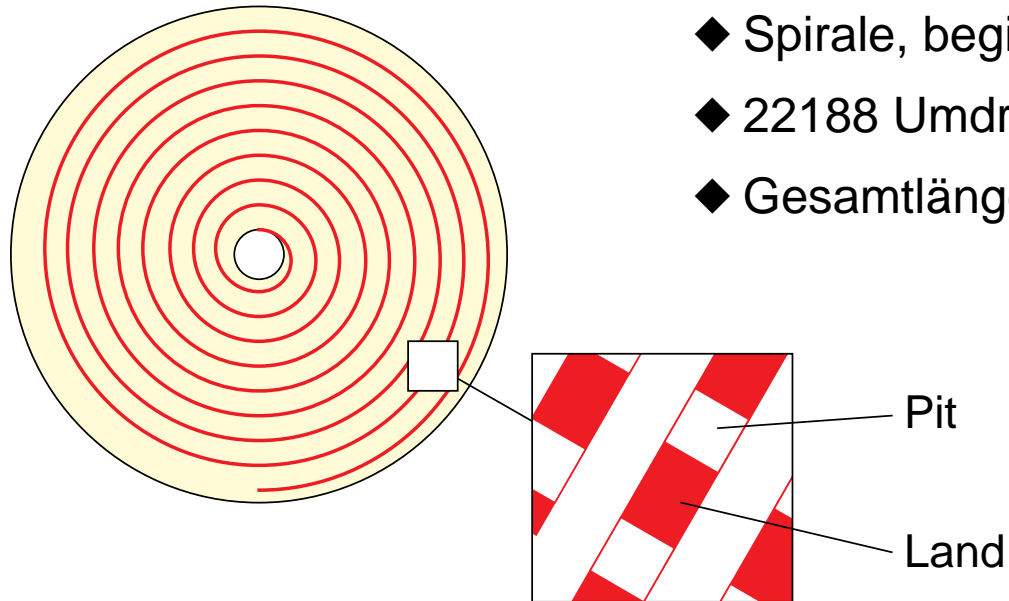
2.1 Festplatten (4)

- 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
 - ◆ heutige Platten haben internen Cache und verbergen die Hardware-Details

- Blöcke sind üblicherweise nummeriert
 - ◆ früher getrennte Nummerierung: Zylindernummer, Sektornummer
 - ◆ heute durchgehende Nummerierung der Blöcke
 - Kompatibilität zu alten Betriebssystemen wird durch *logical CHS (Cylinder/Head/Sector)*-Umrechnung hergestellt

2.2 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

2.2 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)
- ◆ 52-fach-Laufwerk: 7.987.200 Bytes pro Sekunde

2.2 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 Laserlichtfarbe
- ◆ einseitig oder zweiseitig beschrieben
- ◆ ein- oder zweischichtig beschrieben
- ◆ Kapazität: 4,7 bis 17 GB

Speicherung von Dateien

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

3.1 Kontinuierliche Speicherung

- Datei wird in Blöcken mit aufsteigenden Blocknummern gespeichert
 - ◆ Nummer des ersten Blocks und Anzahl der Folgeblöcke muss gespeichert werden
- ★ Vorteile
 - ◆ Zugriff auf alle Blöcke mit minimaler Positionierzeit des Schwenkarms
 - ◆ Schneller direkter Zugriff auf bestimmter Dateiposition
 - ◆ Einsatz z. B. bei Systemen mit Echtzeitanforderungen

3.1 Kontinuierliche Speicherung (2)

▲ Probleme

- ◆ Finden des freien Platzes auf der Festplatte (Menge aufeinanderfolgender und freier Plattenblöcke)
- ◆ Fragmentierungsproblem (Verschnitt: nicht nutzbare Plattenblöcke; siehe auch Speicherverwaltung)
- ◆ Größe bei neuen Dateien oft nicht im Voraus bekannt
- ◆ Erweitern ist problematisch
 - Umkopieren, falls kein freier angrenzender Block mehr verfügbar

3.1 Kontinuierliche Speicherung (3)

■ Variation

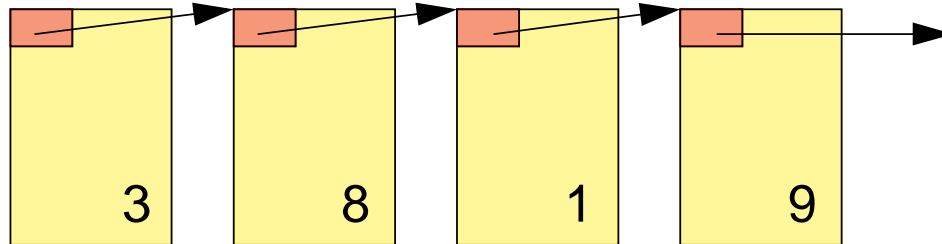
- ◆ Unterteilen einer Datei in Folgen von Blöcken (*Chunks, Extents*)
- ◆ Blockfolgen werden kontinuierlich gespeichert
- ◆ Pro Datei muss erster Block und Länge jedes einzelnen Chunks gespeichert werden

▲ Problem

- ◆ Verschnitt innerhalb einer Folge (siehe auch Speicherverwaltung: interner Verschnitt bei Seitenadressierung)

3.2 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

- ★ Datei kann wachsen und verlängert werden

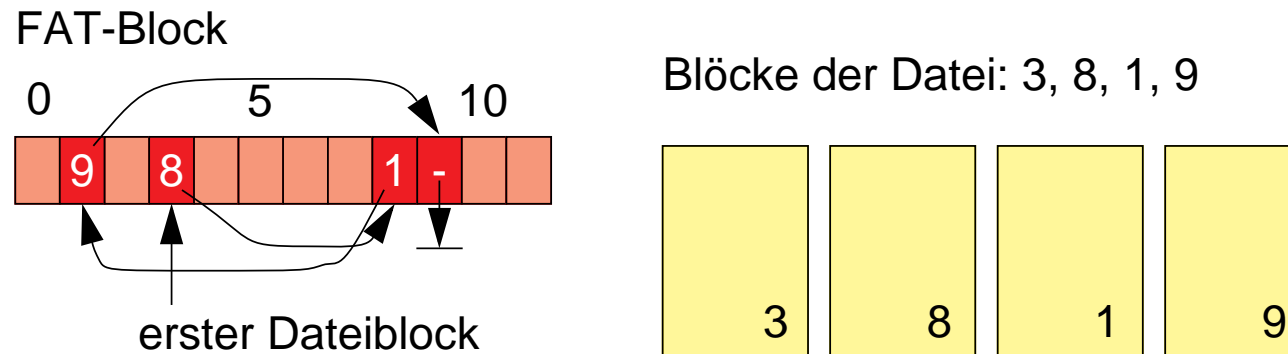
3.2 Verkettete Speicherung (2)

▲ 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
- ◆ schlechter direkter Zugriff auf bestimmte Dateiposition
- ◆ häufiges Positionieren des Schreib-, Lesekopfs bei verstreuten Datenblöcken

3.2 Verkettete Speicherung (3)

- Verkettung wird in speziellen Plattenblocks gespeichert
 - ◆ FAT-Ansatz (*FAT: File Allocation Table*), z. B. MS-DOS, Windows 95



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

3.2 Verkettete Speicherung (4)

▲ 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
- ◆ aufwändige Suche nach dem zugehörigen Datenblock bei bekannter Position in der Datei
- ◆ häufiges Positionieren des Schreib-, Lesekopfs bei verstreuten Datenblöcken

3.3 Indiziertes Speichern

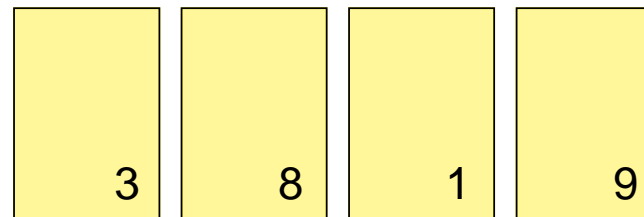
- Spezieller Plattenblock enthält Blocknummern der Datenblocks einer Datei

Indexblock



↑
erster Dateiblock

Blöcke der Datei: 3, 8, 1, 9

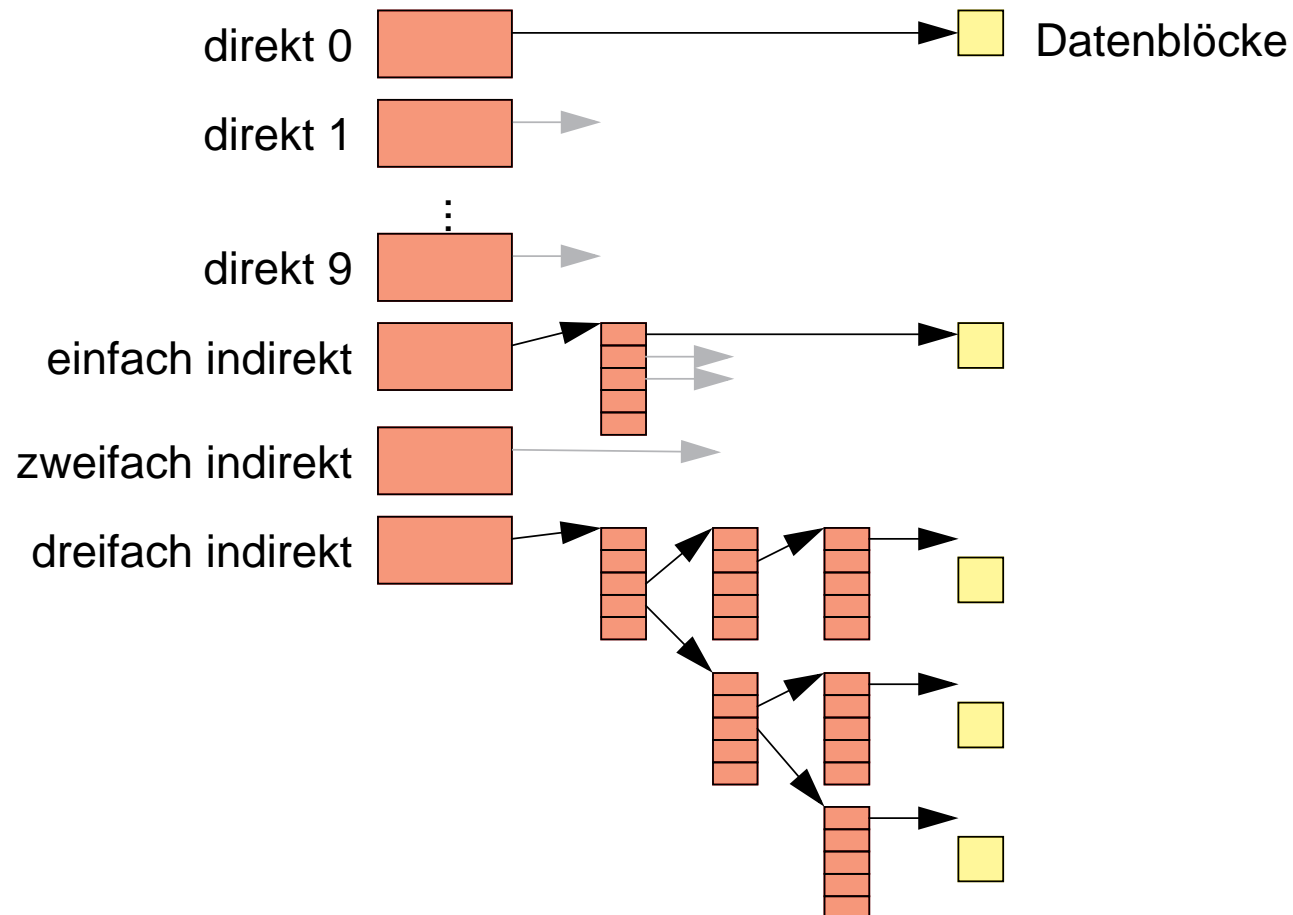


▲ Problem

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

3.3 Indiziertes Speichern (2)

■ Beispiel UNIX Inode

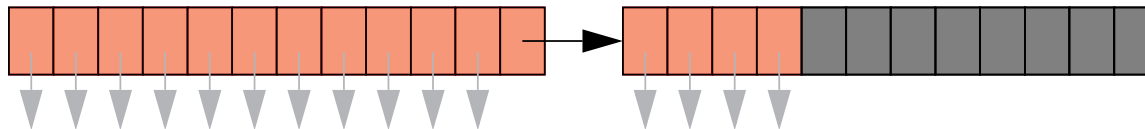


3.3 Indiziertes Speichern (3)

- ★ 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)

Freispeicherverwaltung

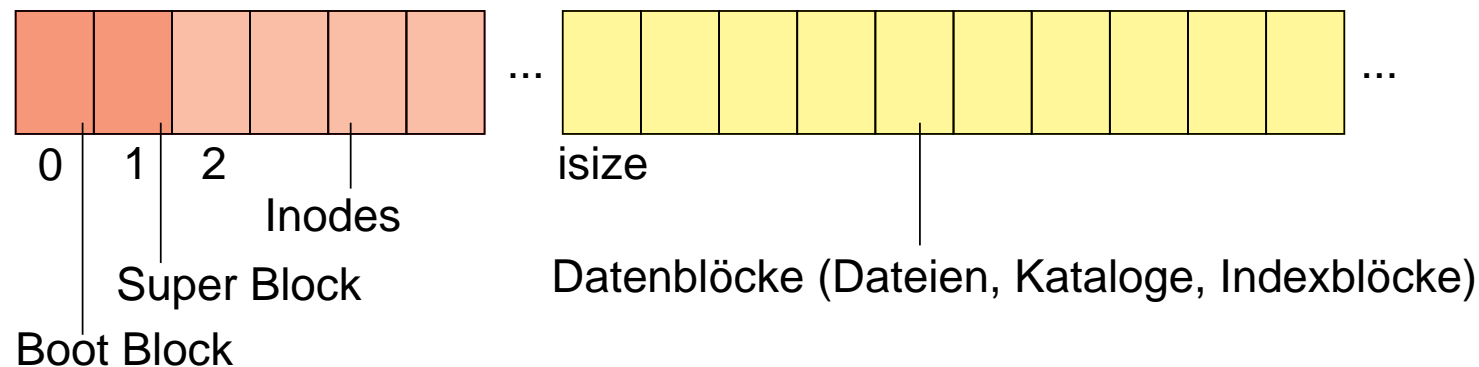
- Prinzipiell ähnlich wie Verwaltung von freiem Hauptspeicher
 - ◆ Bitvektoren zeigen für jeden Block Belegung an
 - ◆ verkettete Listen repräsentieren freie Blöcke
 - Verkettung kann in den freien Blöcken vorgenommen werden
 - Optimierung: aufeinanderfolgende Blöcke werden nicht einzeln aufgenommen, 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



Beispiel: UNIX File Systems

5.1 System V File System

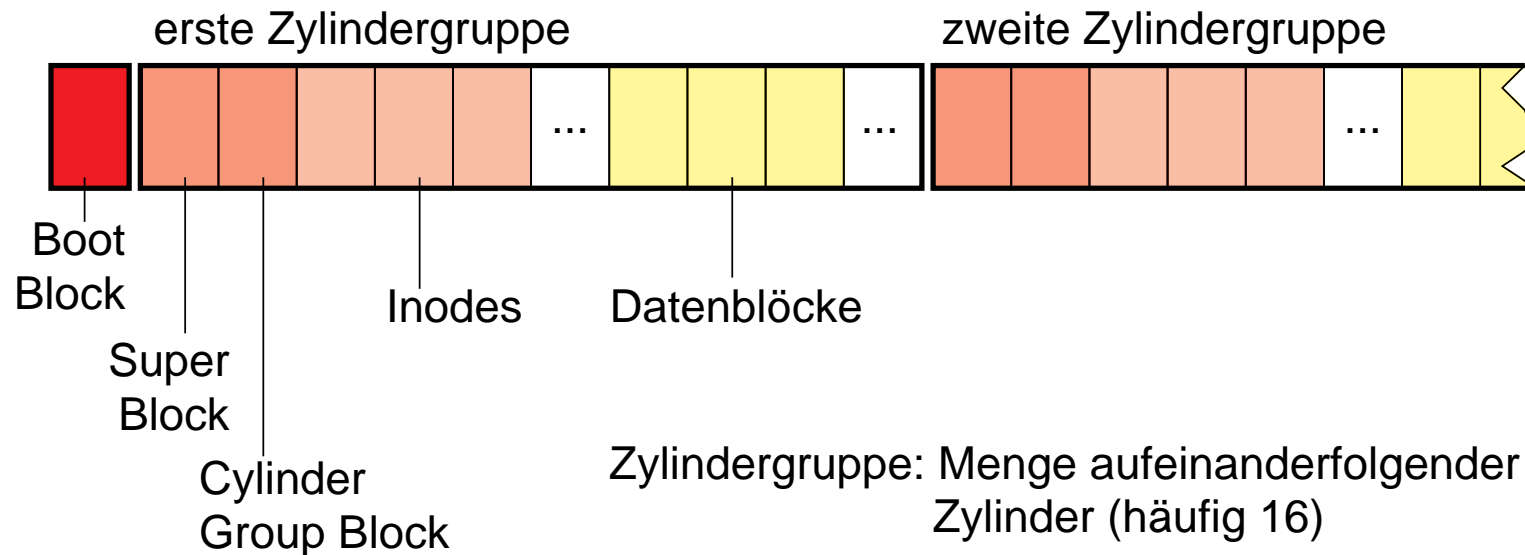
■ Blockorganisation



- ◆ 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 und freier Inodes
 - Attribute (z.B. *Modified flag*)

5.2 BSD 4.2 (Berkeley Fast File System)

■ Blockorganisation

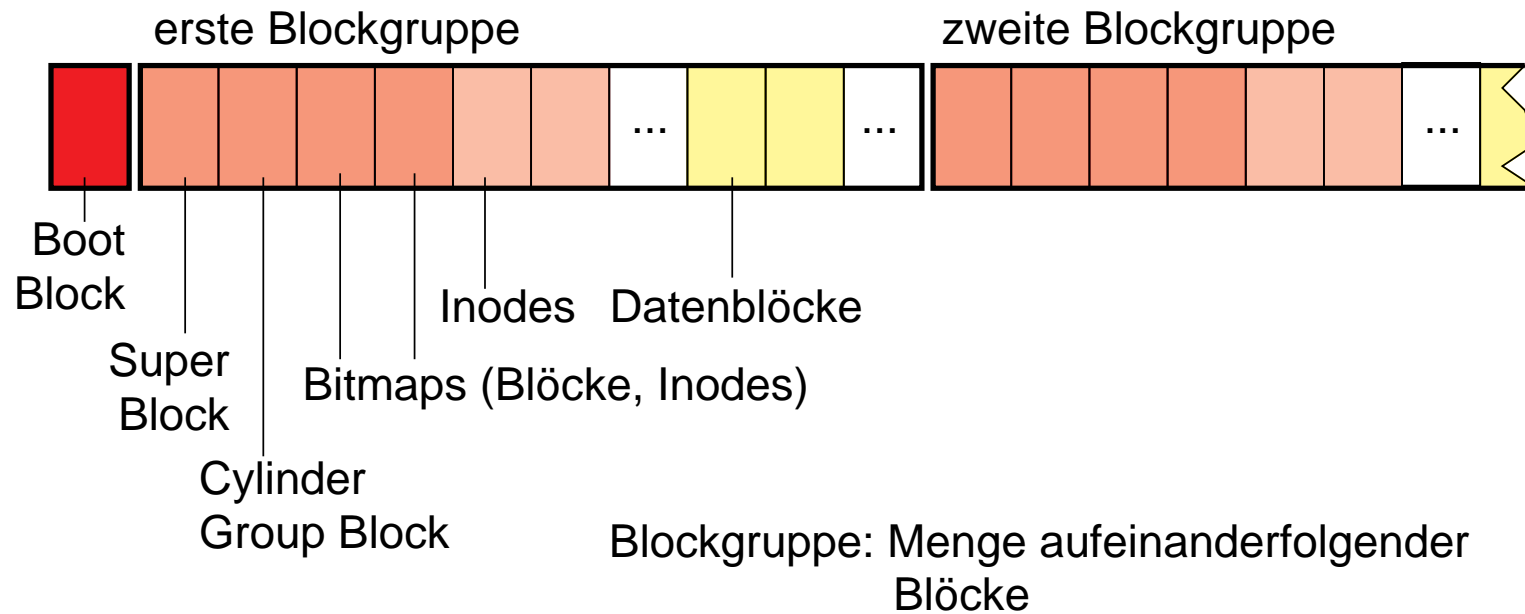


- ◆ Kopie des Super Blocks in jeder Zylindergruppe
- ◆ freie Inodes u. freie Datenblöcke werden im *Cylinder Group Block* gehalten
- ◆ eine Datei wird möglichst innerhalb einer Zylindergruppe gespeichert

★ Vorteil: kürzere Positionierungszeiten

5.3 Linux EXT2 File System

■ Blockorganisation



- ◆ Ähnliches Layout wie BSD FFS
- ◆ Blockgruppen unabhängig von Zylindern

Beispiel: Windows NT (NTFS)

- Dateisystem für Windows NT
- Datei
 - ◆ beliebiger Inhalt; für das Betriebssystem ist der Inhalt transparent
 - ◆ Rechte verknüpft mit NT-Benutzern und -Gruppen
 - ◆ Datei kann automatisch komprimiert oder verschlüsselt gespeichert werden
 - ◆ große Dateien bis zu 2^{64} Bytes lang
 - ◆ Hard links: mehrere Einträge derselben Datei in verschiedenen Katalogen möglich
- Dateiinhalt: Sammlung von *Streams*
 - ◆ *Stream*: einfache, unstrukturierte Folge von Bytes
 - ◆ "normaler Inhalt" = unbenannter Stream (default stream)
 - ◆ dynamisch erweiterbar
 - ◆ Syntax: dateiname:streamname

6.1 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)

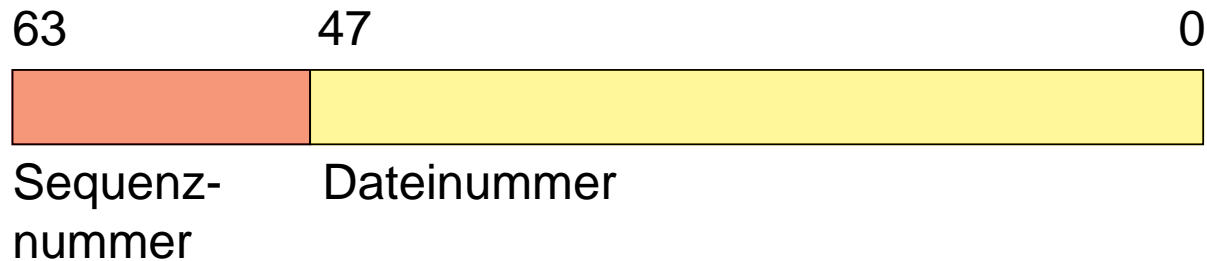
■ Basiseinheit „Strom“

- ◆ 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)

6.1 Dateiverwaltung (2)

■ *File-Reference*

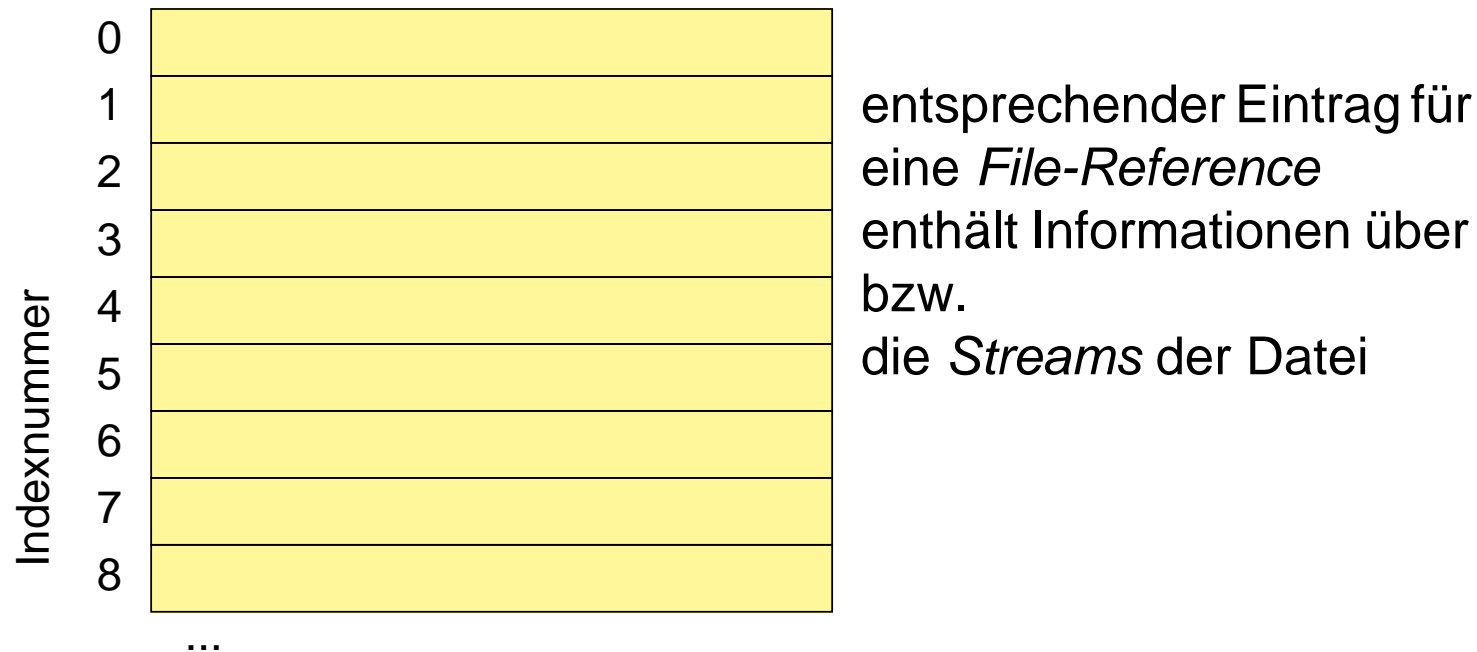
- ◆ Bezeichnet eindeutig eine Datei oder einen Katalog



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

6.2 Master-File-Table

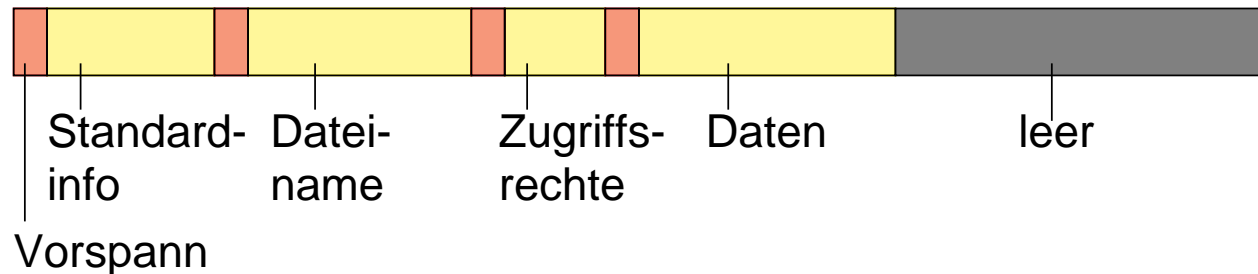
- Rückgrat des gesamten Systems
 - ◆ große Tabelle mit gleich langen Elementen (1KB, 2KB oder 4KB groß, je nach Clustergröße)
 - ◆ kann dynamisch erweitert werden



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

6.2 Master-File-Table (2)

■ Eintrag für eine kurze Datei



■ Streams

◆ Standard-Information (immer in der MFT)

- enthält Länge, Standard-Attribute, Zeitstempel, Anzahl der Hard links, Sequenznummer der gültigen File-Reference

◆ Dateiname (immer in der MFT)

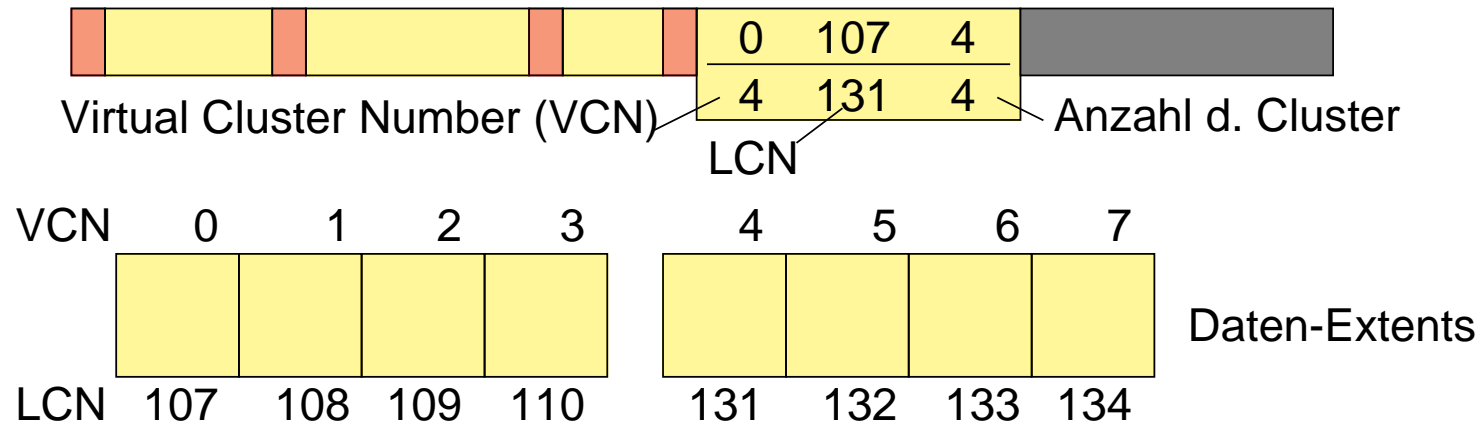
- kann mehrfach vorkommen (Hard links)

◆ Zugriffsrechte (*Security Descriptor*)

◆ Eigentliche Daten

6.2 Master-File-Table (3)

■ Eintrag für eine längere Datei



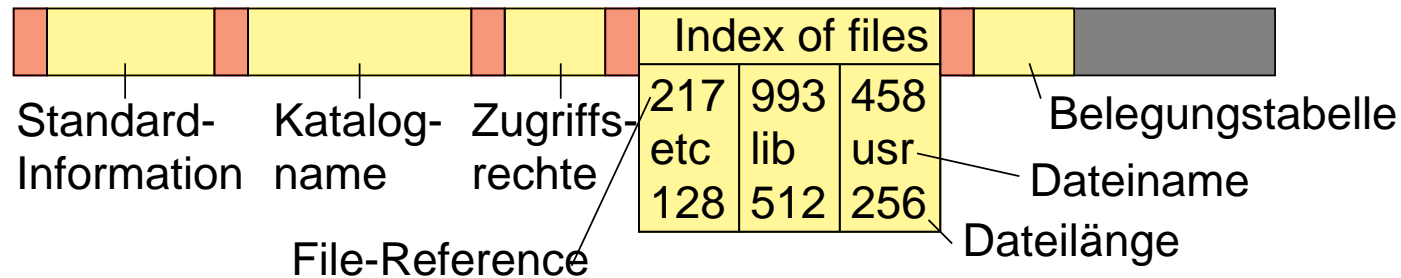
- ◆ **Extents** werden außerhalb der MFT in aufeinanderfolgenden Clustern gespeichert
- ◆ Lokalisierungsinformationen werden in einem eigenen Stream gespeichert

6.2 Master-File-Table (4)

- Mögliche weitere Streams (*Attributes*)
 - ◆ Index
 - Index über einen Attributsschlüssel (z.B. Dateinamen) implementiert Katalog
 - ◆ Indexbelegungstabelle
 - Belegung der Struktur eines Index
 - ◆ Attributliste (immer in der MFT)
 - wird benötigt, falls nicht alle Streams in einen MFT Eintrag passen
 - referenzieren weitere MFT Einträge und deren Inhalt
 - ◆ Streams mit beliebigen Daten
 - wird gerne zum Verstecken von Viren genutzt, da viele Standard-Werkzeuge von Windows nicht auf die Bearbeitung mehrerer Streams eingestellt sind (arbeiten nur mit dem unbenannten Stream)

6.2 Master File Table (5)

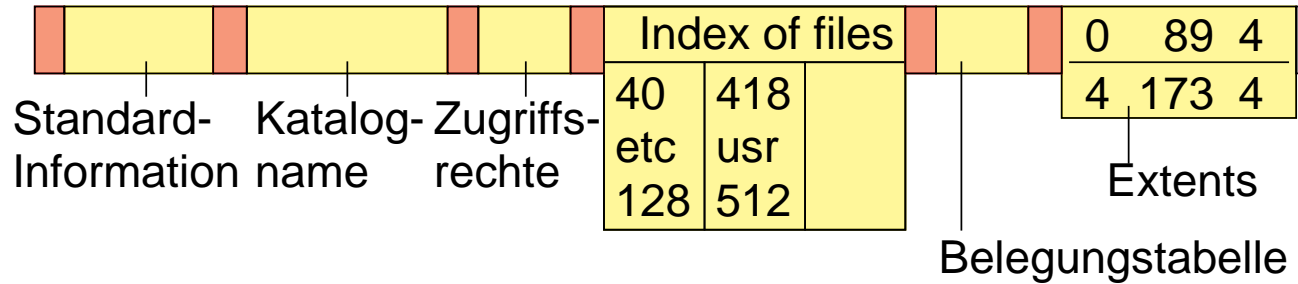
■ Eintrag für einen kurzen Katalog



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

6.2 Master File Table (6)

Eintrag für einen längeren Katalog



Daten-Extents

VCN	0	1	2	3	4	5	6	7	
	918	773	473		873	910		10	File reference
	cd	cs	doc		lib	news		tmp	Dateiname
	128	2781	128		512	1024		128	Dateilänge
LCN	89	90	91	92	173	174	175	176	

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