

# Systemprogrammierung

Grundlage von Betriebssystemen

## Teil C – XIII. Dateisysteme

Jürgen Kleinöder

20. Januar 2016

28. Januar 2016



## Agenda

Medien

Speicherung von Dateien

Freispeicherverwaltung

Beispiele: Dateisysteme unter UNIX und Windows

Dateisysteme mit Fehlererholung

Datensicherung

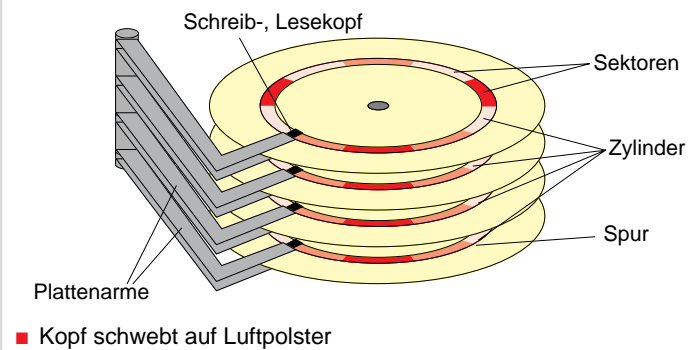


## Medien

### Festplatten

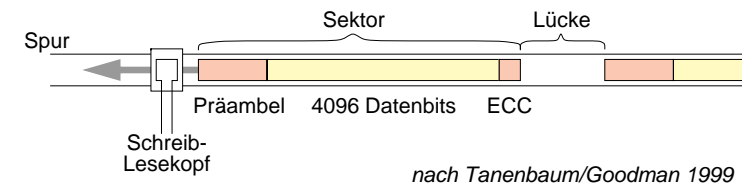
#### ■ Häufigstes Medium zum Speichern von Dateien

##### ■ Aufbau einer Festplatte



## Festplatten (2)

#### ■ Sektoraufbau



- Breite der Spur: 5–10  $\mu\text{m}$
- Spuren pro Zentimeter: 800–2000
- Breite einzelner Bits: 0,1–0,2  $\mu\text{m}$

#### ■ Zonen

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



## Festplatten (3)

### ■ Datenblätter von drei (alten) Beispielpplatten

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
Positionierungszeiten	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

06.2015: Kapazität bis 10TB bei 7.200 U/min oder 0,6 - 2 TB bei 15.000 U/min, Zugriffszeit ab 2,7 ms, Transferrate bis 200 MB/s

SSD: Kapazität bis 16 TB, Zugriffszeit ca. 0,1 ms, Transferrate bis 3 GB/s



## 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

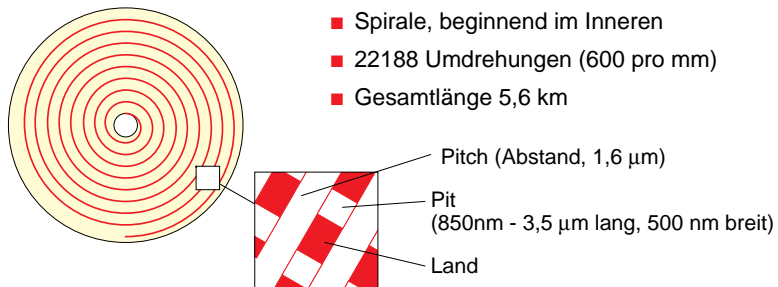
### ■ Blöcke sind üblicherweise nummeriert

- früher getrennte Nummerierung: Zylindernummer, Sektornummer
- heute durchgehende Nummerierung der Blöcke



## CD-ROM / DVD

### ■ Aufbau einer CD



- **Pit:** Vertiefung, wird von Laser (780 nm Wellenlänge) abgetastet

### ■ DVD

- gleiches Grundkonzept, Wellenlänge des Lasers 650 nm
- Pits und Spurbabstand weniger als halb so groß



## CD-ROM / DVD (2)

### ■ Kodierung einer CD

- **Symbol:** ein Byte wird mit 14 Bits kodiert (kann bereits bis zu zwei Bitfehler korrigieren)
- **Frame:** 42 Symbole (192 Datenbits, 396 Fehlerkorrekturbits)
- **Sektor:** 98 Frames werden zusammengefasst (16 Bytes Präambel, 2048 Datenbytes, 288 Bytes Fehlerkorrektur)
- **Effizienz:** 7203 Bytes transportieren 2048 Nutzbytes (28,4 %)

### ■ Kodierung einer DVD

- Codierung mit Reed-Solomon-Product-Code, 8/16-Bit-Modulation, 43,2 % Nutzdaten

### ■ Transferrate

- CD-Single-Speed-Laufwerk: 75 Sektoren/Sek. (153.600 Bytes/Sek.)
- CD-72-fach-Laufwerk: 11,06 MB/Sek.
- DVD 1-fach: 1.3 MB/sec, 24-fach: 33.2 MB/sec



## CD-ROM / DVD (3)

- Kapazität
  - CD: ca. 650 MB
  - DVD single layer: 4.7 GB
  - DVD dual layer: 8.5 GB, beidseitig: 17 GB
- Varianten
  - **DVD/CD-R** (Recordable): einmal beschreibbar
  - **DVD/CD-RW** (Rewritable): mehrfach beschreibbar



## Speicherung von Dateien

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

### 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



## 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



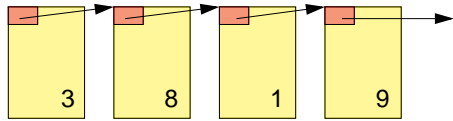
## Kontinuierliche Speicherung (3)

- 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)



## 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



## Verkettete Speicherung (2)

### ▲ Probleme

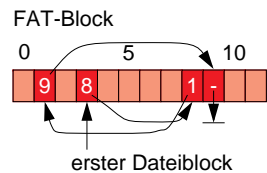
- Speicher für Verzeigerung geht von den Nutzdaten im Block ab (ungünstig im Zusammenhang mit Paging: Seite würde immer aus Teilen von zwei Plattenblöcken bestehen)
- Fehleranfälligkeit: Datei ist nicht restaurierbar, falls einmal Verzeigerung fehlerhaft
- schlechter direkter Zugriff auf bestimmte Dateiposition
- häufiges Positionieren des Schreib-, Lesekopfs bei verstreuten Datenblöcken



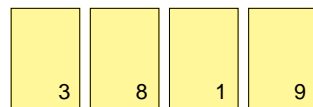
## Verkettete Speicherung (3)

### ■ Verkettung wird in speziellen Plattenblocks gespeichert

#### ■ FAT-Ansatz (FAT: File Allocation Table), z. B. MS-DOS, Windows 95



Blöcke der Datei: 3, 8, 1, 9



#### ★ Vorteile

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



## Verkettete Speicherung (4)

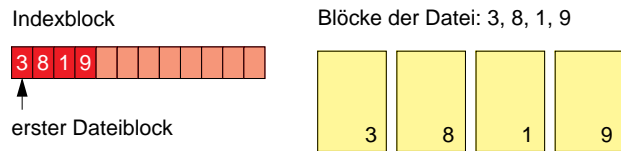
### ▲ Probleme

- 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



## Indiziertes Speichern

- Spezieller Plattenblock enthält Blocknummern der Datenblocks einer Datei



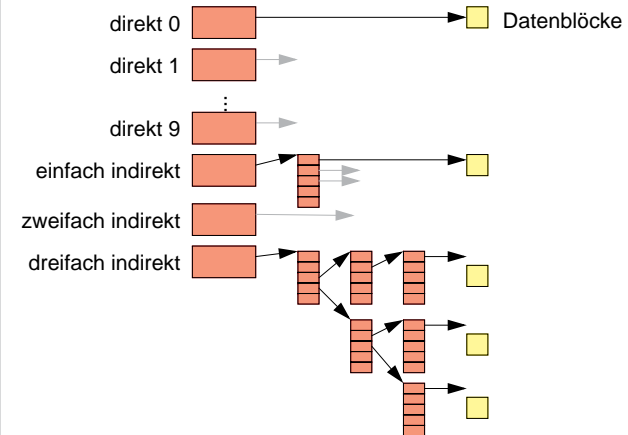
### ▲ Problem

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



## Indiziertes Speichern (2)

- Beispiel UNIX Inode



## 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

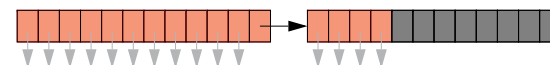
- mehrere Blöcke müssen geladen werden (nur bei langen Dateien)



## Freispeicherverwaltung

- Prinzipiell ähnlich wie Verwaltung von freiem Hauptspeicher

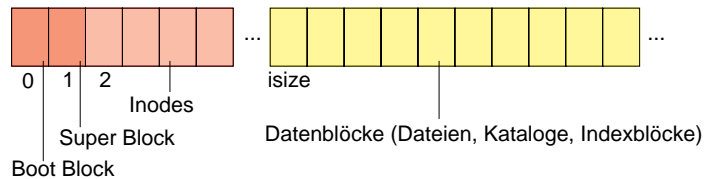
- Bitvektoren zeigen für jeden Block Belegung an
- verkettete Listen repräsentieren freie Blöcke
  - Verkettung kann in den freien Blöcken vorgenommen werden
  - Optimierung: aufeinanderfolgende Blöcke werden nicht einzeln aufgenommen, sondern als Stück verwaltet
  - Optimierung: ein freier Block enthält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



## Beispiel: UNIX Dateisysteme

### System V File System

#### ■ Blockorganisation

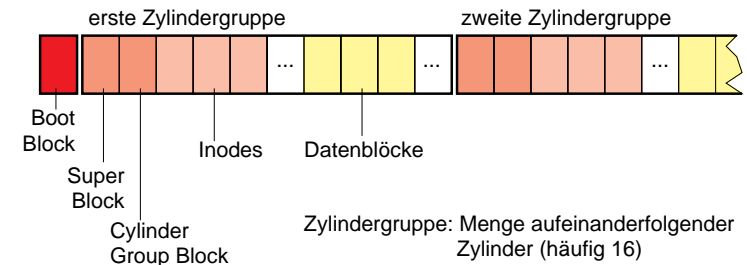


- Boot Block enthält Informationen zum Laden eines initialen Programms
- Super Block enthält Verwaltungsinformation für ein Dateisystem
  - Anzahl der Blöcke, Anzahl der Inodes
  - Anzahl und Liste freier Blöcke und freier Inodes
  - Attribute (z.B. *Modified flag*)



## BSD 4.2 (Berkeley Fast File System)

#### ■ Blockorganisation



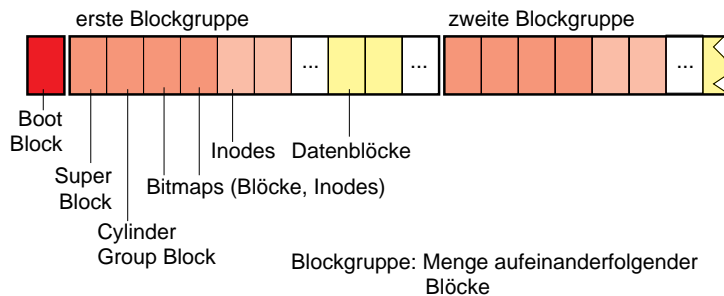
- Kopie des Super Blocks in jeder Zylindergruppe
- freie Inodes u. freie Datenblöcke werden im *Cylinder Group Block* gehalten
- eine Datei wird möglichst innerhalb einer Zylindergruppe gespeichert

★ Vorteil: kürzere Positionierungszeiten



## Linux EXT2/3/4

#### ■ EXT2: Blockorganisation



- Ähnliches Layout wie BSD FFS
- Blockgruppen unabhängig von Zylindern
- EXT3: Erweiterung als Journaling-File-System (siehe Abschnitt 7.2)
- EXT4: Einführung von Extents (siehe NTFS) und viele neue Details



## Beispiel: Windows NTFS

#### ■ Dateisystem für Windows-Systeme (seit Windows NT 3.1, 1993)

#### ■ 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



## Dateiverwaltung

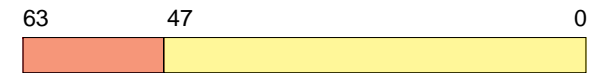
- Basiseinheit „Cluster“
  - 512 Bytes bis 4 Kilobytes (beim Formatieren festgelegt)
  - wird auf eine Menge von hintereinanderfolgenden Blöcken abgebildet
  - logische Cluster-Nummer als Adresse (LCN)
- Basiseinheit „Strom“
  - jede Datei kann mehrere (Daten-)Ströme speichern
  - einer der Ströme wird für die eigentlichen Daten verwendet
  - Dateiname, MS-DOS Dateiname, Zugriffsrechte, Attribute und Zeitstempel werden jeweils in eigenen Datenströmen gespeichert (leichte Erweiterbarkeit des Systems)



## Dateiverwaltung (2)

### File-Reference

- Bezeichnet eindeutig eine Datei oder einen Katalog



Sequenznummer

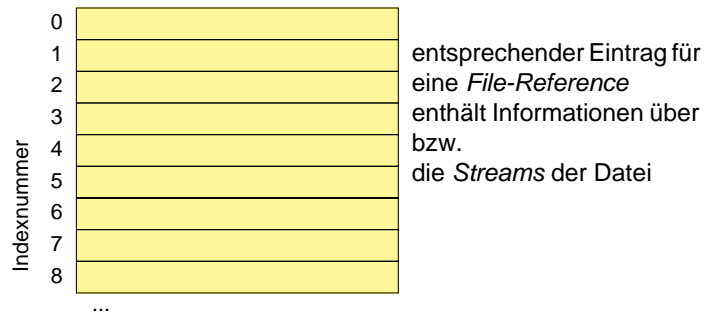
Dateinummer

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



## 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

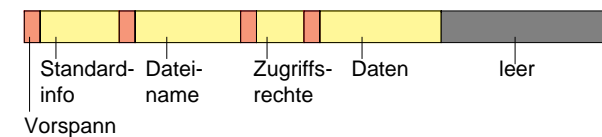


- Index in die Tabelle ist Teil der *File-Reference*



## Master-File-Table (2)

### Eintrag für eine kurze Datei



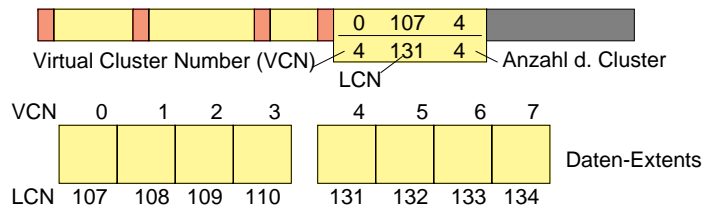
### Streams

- 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



## 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



## Master-File-Table (4)

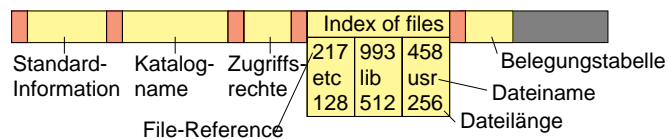
### Mögliche weitere Streams (*Attributes*)

- Index
  - Index über einen Attributschlüssel (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)



## Master File Table (5)

### Eintrag für einen kurzen Katalog

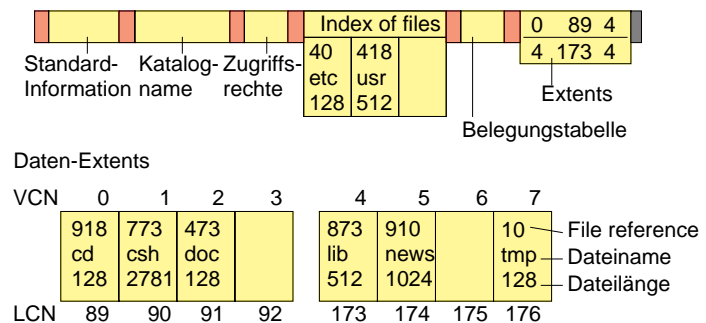


- Dateien des Katalogs werden mit File-References benannt
- Name und Standard-Attribute (z.B. Länge) der im Katalog enthaltenen Dateien und Kataloge werden auch im Index gespeichert (doppelter Aufwand beim Update; schnellerer Zugriff beim Kataloglisten)



## Master File Table (6)

### Eintrag für einen längeren Katalog



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





## Metadaten

### ■ Alle Metadaten werden in Dateien gehalten

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



## 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
- Attributtabelle: 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



## Fehlererholung

### ■ NTFS ist ein Journaling-File-System

- Änderungen an der MFT und an Dateien werden protokolliert.
- Konsistenz der Daten und Metadaten kann nach einem Systemausfall durch Abgleich des Protokolls mit den Daten wieder hergestellt werden.

### ▲ Nachteile

- etwas ineffizienter
- nur für Volumes >400 MB geeignet



## Dateisysteme mit Fehlererholung

### ■ Metadaten und aktuell genutzte Datenblöcke geöffneter Dateien werden im Hauptspeicher gehalten (Dateisystem-Cache)

- effizienter Zugriff
- Konsistenz zwischen Cache und Platte muss regelmäßig hergestellt werden
  - synchrone Änderungen: Operation kehrt erst zurück, wenn Änderungen auf der Platte gespeichert wurden
  - asynchrone Änderungen: Änderungen erfolgen nur im Cache, Operation kehrt danach sofort zurück, Synchronisation mit der Platte erfolgt später

### ■ Mögliche Fehlerursachen

- Stromausfall (dummer Benutzer schaltet einfach Rechner aus)
- Systemabsturz



## Konsistenzprobleme

- Fehlerursachen & Auswirkungen auf das Dateisystem
  - Cache-Inhalte und aktuelle E/A-Operationen gehen verloren
  - inkonsistente Metadaten
    - z. B. Katalogeintrag fehlt zur Datei oder umgekehrt
    - z. B. Block ist benutzt aber nicht als belegt markiert
- ★ Reparaturprogramme
  - Programme wie **chkdsk**, **scandisk** oder **fsck** können inkonsistente Metadaten reparieren
- ▲ Datenverluste bei Reparatur möglich
- ▲ Große Platten bedeuten lange Laufzeiten der Reparaturprogramme



## Journaling-File-Systems

- Zusätzlich zum Schreiben der Daten und Meta-Daten (z. B. Inodes) wird ein Protokoll der Änderungen geführt
  - Grundidee: Log-based Recovery bei Datenbanken
  - 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.



## Journaling-File-Systems (2)

- Protokollierung
  - für jeden Einzelschritt 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
- 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*).



## Journaling-File-Systems (3)

- Beispiel: Löschen einer Datei im NTFS
  - 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*

