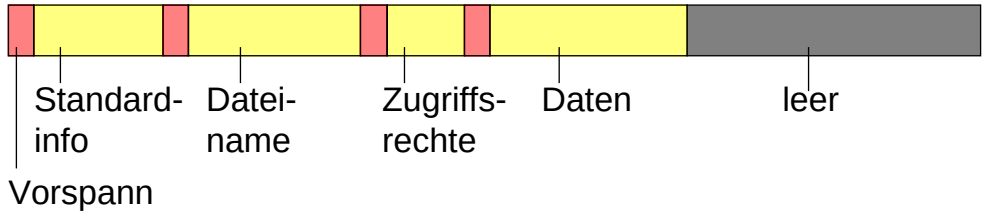
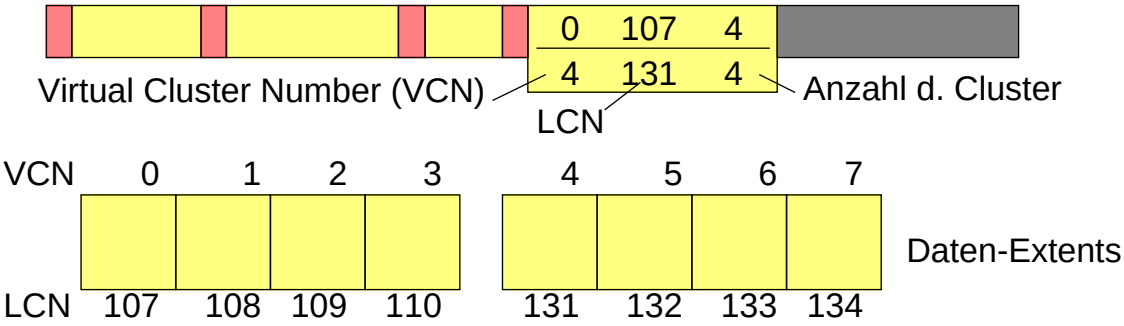
3 Master File Table

- Eintrag für eine kurze Datei



- Eintrag für eine längere Datei



  - ◆ Extents werden außerhalb der MFT gespeichert

3 Master File Table (2)

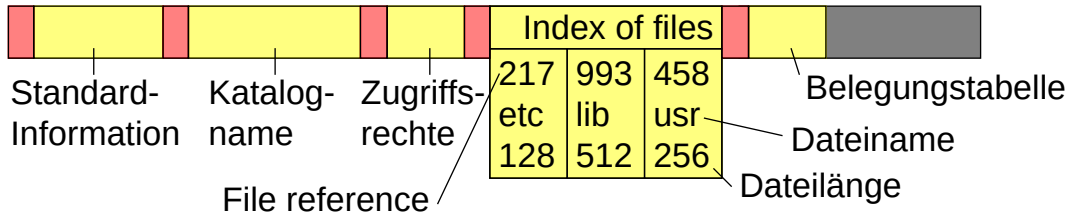
- Mögliche Ströme *(Attributes)*
  - ◆ Standard Information (immer in der MFT)
    - enthält Länge, MS-DOS Attribute, Zeitstempel, Anzahl der Hard links
  - ◆ Dateiname (immer in der MFT)
    - kann mehrfach vorkommen (Hard links, MS-DOS Name)
  - ◆ Zugriffsrechte
    - *Security descriptor*
  - ◆ Daten
    - die eigentlichen Daten
  - ◆ Index
    - Index über einen Attributschlüssel (z.B. Dateinamen) implementiert Katalog
  - ◆ Indexbelegungstabelle

3 Master File Table (3)

◆ Attributliste (immer in der MFT)

- wird benötigt, falls nicht alle Ströme in einen MFT Eintrag passen
- referenzieren weitere MFT Einträge und deren Inhalt

■ Eintrag für einen kurzen Katalog

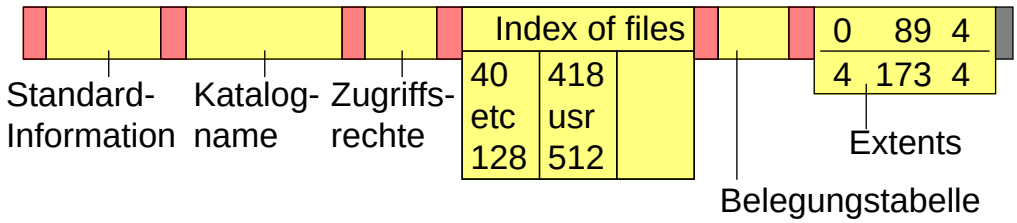


◆ Dateien des Katalogs werden mit File references benannt

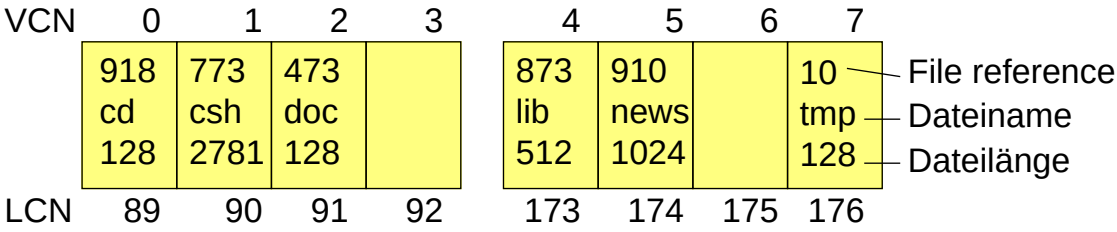
◆ Name und Länge der im Katalog enthaltenen Dateien und Kataloge werden auch im Index gespeichert
(doppelter Aufwand beim Update; schnellerer Zugriff beim Kataloglisten)

3 Master File Table (4)

■ Eintrag für einen längeren Katalog



Daten-Extents

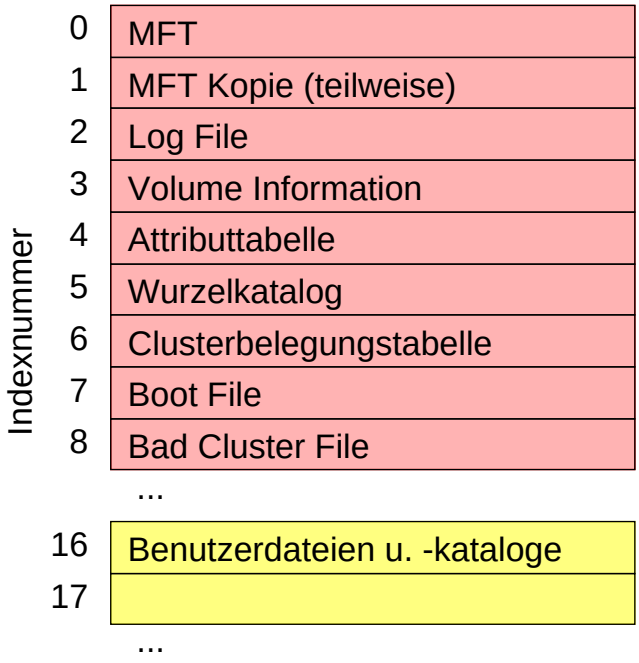


◆ Speicherung als B^+-Baum (sortiert, schneller Zugriff)

◆ in einen Cluster passen zwischen 3 und 15 Dateien (im Bild nur eine)

4 Metadaten

- Alle Metadaten werden in Dateien gehalten



Feste Dateien in der MFT

4 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

5 Fehlererholung

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

5 Fehlererholung (2)

- ★ Log structured file system
  - ◆ Protokollieren aller Änderungen am Dateisystem in einer Protokolldatei *(Log file)*
  - ◆ Änderungen treten als Teil von Transaktionen auf
    - Erzeugen, löschen, erweitern, verkürzen von Dateien
    - Dateiattribute verändern
    - Datei umbenennen
  - ◆ Schreiben der Logdatei bevor die Änderungen auf Platte geschrieben werden (wurde etwas auf Platte geändert steht auch das Protokoll auf Platte)
  - ◆ Beim Bootvorgang wird überprüft, ob die protokollierten Änderungen vorhanden sind:
    - Transaktion kann evtl. wiederholt bzw. abgeschlossen werden *(Redo)*
    - angefangene aber nicht beendete Transaktionen werden rückgängig gemacht *(Undo)*

5 Fehlererholung (3)

- 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*

5 Fehlererholung (4)

- 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: rückwärts 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

5 Fehlererholung (5)

★ 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

F.7 Limitierung der Plattennutzung

■ Mehrbenutzersysteme

- ◆ einzelnen Benutzern sollen verschieden große 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)

F.8 Fehlerhafte Plattenblöcke

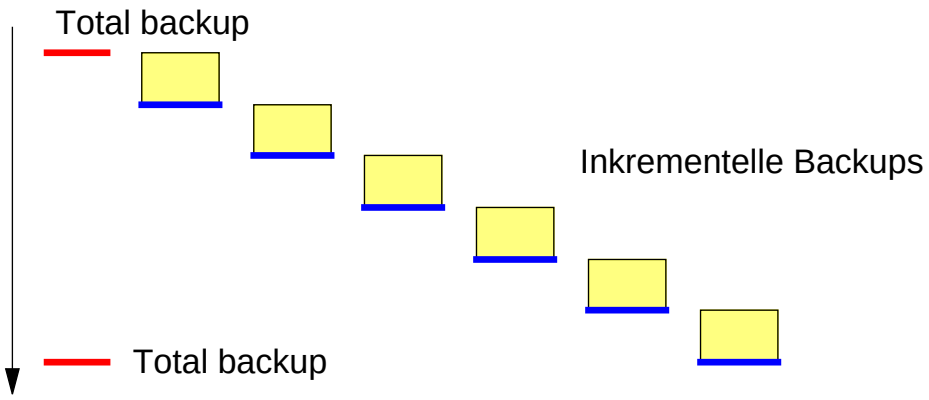
- Blöcke, die beim Lesen Fehlermeldungen erzeugen
  - z.B. Checksummenfehler
- Hardwarelösung
  - Platte und Plattencontroller bemerken selbst fehlerhafte Blöcke und maskieren diese aus
  - Zugriff auf den Block wird vom Controller automatisch auf einen „gesunden“ Block umgeleitet
- Softwarelösung
 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

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

1 Beispiele für Backup Scheduling

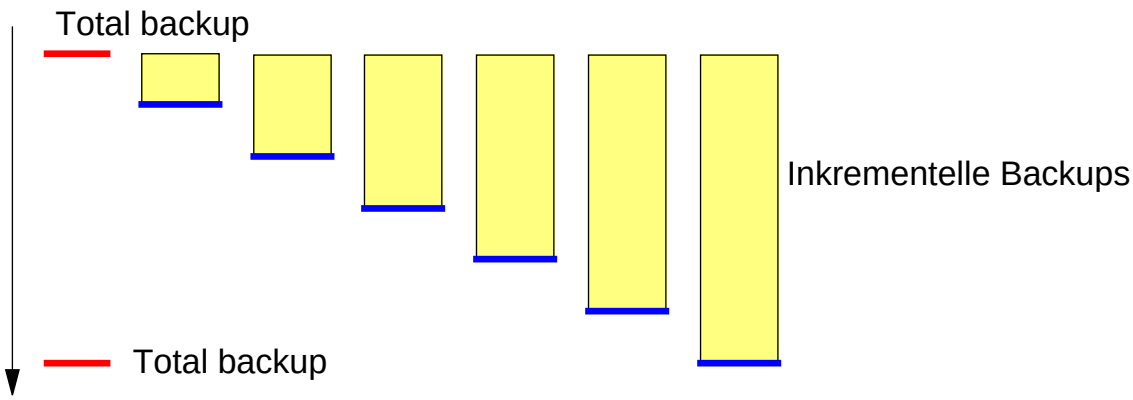
- Gestaffelte inkrementelle Backups



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

1 Beispiele für Backup Scheduling (2)

- Gestaffelte inkrementelle Backups zum letzten Total backup

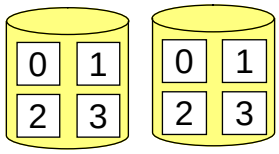


  - z.B. alle Woche ein Total backup und jeden Tag ein inkrementelles Backup zum letzten Total backup: maximal 2 Backups müssen eingespielt werden

- Hierarchie von Backupläufen
  - mehrstufige inkrementelle Backups zum Backup der nächst höheren Stufe
  - optimiert Archivmaterial und Restaurierungszeit

2 Einsatz mehrere redundanter Platten

- Gespiegelte Platten (*Mirroring*; RAID 0)
  - Daten werden auf zwei Platten gleichzeitig gespeichert

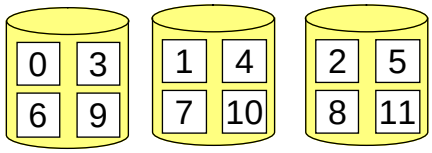


  - Implementierung durch Software (File system, Plattentreiber) oder Hardware (spez. Controller)
  - eine Platte kann ausfallen
  - schnelleres Lesen (da zwei Platten unabhängig voneinander beauftragt werden können)

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

2 Einsatz mehrere redundanter Platten (2)

- Gestreifte Platten (*Striping*; RAID 1)
  - Daten werden über mehrere Platten gespeichert



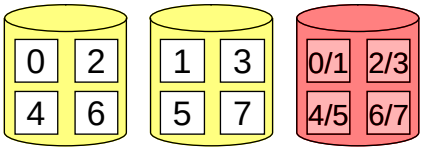
  - Datentransfers sind nun schneller, da mehrere Platten gleichzeitig angesprochen werden können

- Nachteil
  - keinerlei Datensicherung: Ausfall einer Platte läßt Gesamtsystem ausfallen

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

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)

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: $P_{neu} = P_{alt} \oplus B_{alt} \oplus B_{neu}$ (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])