# 6.2 FAT32 Dateisystem

- Dateisystem für **Windows 98**
- einige Unterschiede zum Linux-Dateisystem EXT2:
  - **keine Benutzeridentifikation** für Dateien und Verzeichnisse!
  - Partitionen werden durch Laufwerke repräsentiert, die durch Buchstaben dargestellt werden, z.B.: `A:` (Floppy), `C:` (Platte), `D:` (DVD)
  - jedem Windows-Programm ist ein **aktuelles Laufwerk** und ein **aktuelles Verzeichnis** aus Dateibaum zugeordnet
  - es gibt keine Inodes: die Speicherung aller Attribute einer Datei erfolgt im Verzeichnis
  - es gibt keine *Hard Links*
  - die kleinste adressierbare Einheit heißt **Cluster** und ist ein Block mit einer Zweierpotenz von 1 bis 128 Sektoren (bei Formatierung wählbar)
  - die Blockadressierung erfolgt über eine Tabelle (**FAT** = *File Allocation Table*), in der die Verkettung der Cluster aller Dateien gespeichert ist



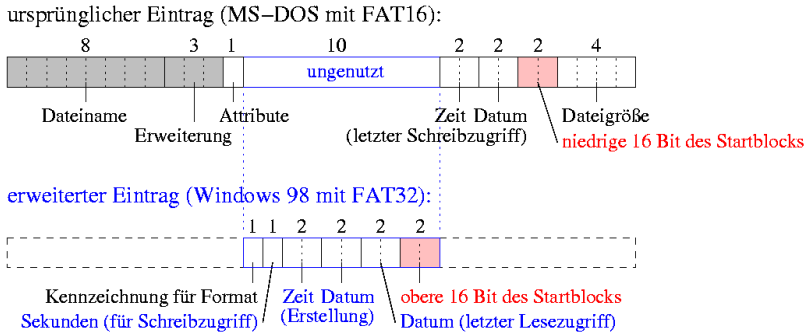
# 6.2 FAT32 Dateisystem (2)

- Attribute einer **FAT32-Datei**:
  - **Name**
    - im MS-DOS Modus: 8 Zeichen Name + 3 Zeichen Erweiterung (z.B.: „`AUTOEXEC.BAT`“)
    - im Windows 98 Modus: 255 Zeichen inklusive Sonderzeichen (z.B.: „`Eigene Dateien`“)
  - **Dateilänge**
  - **Typ**: Verzeichnis, versteckte Datei (*hidden*), Systemdatei (*system*), zu archivierende Datei (*archive*)
  - nur **zwei Zugriffsrechte**: „nur lesbar“ und „schreib- und lesbar“
  - **Ortsinformation**: Nummer des ersten Clusters einer Datei
  - **Zeitstempel**:
    - zunächst nur Datum und Uhrzeit des letzten Schreibzugriffs
    - bei Windows 98 zusätzlich Datum und Uhrzeit der Erstellung sowie Datum des letzten Lesezugriffs



# 6.2 FAT32 Dateisystem (3)

- Aufbau eines **FAT32-Verzeichnis**:
  - unsortierte 32-Byte Einträge werden hintereinander in Liste gespeichert




  - aus langen Dateinamen (bis zu 255 Zeichen) wird ein neuer eindeutiger Name aus 8+3 Zeichen generiert und eingetragen; der vollständige Name wird in zusätzlichen vorangestellten 32-Byte Feldern gespeichert



# 6.2 FAT32 Dateisystem (4)

- die **Verkettung** der Cluster wird in der **FAT** festgehalten:
  - enthält Eintrag für jedes Cluster auf der Festplatte
  - für jedes Cluster einer Datei ist die Nummer des nachfolgenden Clusters als 32-Bit Zahl eingetragen
    - die Nummer des Startblocks kann dem Verzeichnis entnommen werden
    - Dateiende wird durch Eintrag –1 markiert
  - freie Cluster werden durch Eintrag 0 markiert

Auszug aus einer FAT:

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| – | – | 0 | 8 | 5 | 6 | 20 | 0 | 9 | 15 | 11 | 17 | 0 | 0 | 0 | 16 | 18 | 4 | –1 | 0 | –1 | 0 | ... |

Plattenblöcke (Cluster) für Datei A:

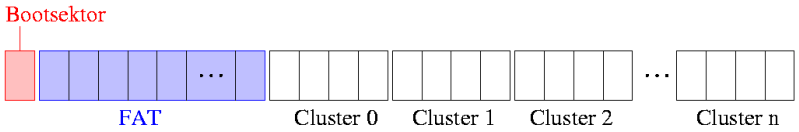
| 10 | 11 | 17 | 4 | 5 | 6 | 20 |
|---|---|---|---|---|---|---|

Plattenblöcke (Cluster) für Datei B:

| 3 | 8 | 9 | 15 | 16 | 18 |
|---|---|---|---|---|---|

## 6.2 FAT32 Dateisystem (5)

- Blockorganisation einer Partition mit FAT32:




  - **Bootsektor** enthält neben dem *Bootloader* noch einige Angaben über das Dateisystem, z,B:
    - Gesamtanzahl der Sektoren (4 Byte)
    - Bytes je Sektor (2 Byte)
    - Sektoren je Cluster (1 Byte, nur Zweierpotenzen von 1 bis 128 erlaubt)
    - Startposition des Hauptverzeichnisses (4 Byte)
    - Label (10 Byte) und Serien-Nummer (4 Byte)
    - Anzahl FATs (1 Byte) und Sektoren je FAT (4 Byte)
  - **FAT** kann zur Erhöhung der Sicherheit auch mehrfach auf Festplatte gespeichert sein



## 6.2 FAT32 Dateisystem (6)

- einige **Nachteile** von FAT32:
  - **umständliche** Datenstrukturen (wegen Kompatibilität zu MS-DOS)
  - **sehr große FAT** bei modernen Festplatten hoher Kapazität
  - Positionieren eines Dateizeigers bei großen Dateien sehr zeitaufwendig
  - für jeden Dateizugriff muss mindestens ein Plattenblock mit einem Teil der FAT von der Festplatte geladen werden
  - FAT enthält Verkettungen für alle Dateien ⇒ es werden stets auch viele nicht benötigte Verkettungsinformationen geladen
  - langsame Suche nach freien Clustern
  - **sehr viele Kopfbewegungen**, wenn Cluster einer Datei verstreut sind
    (⇒ regelmäßiger Aufruf eines Defragmentierungsprogramms sinnvoll; es versucht die Cluster jeder Datei zusammenhängend anzuordnen)
- FAT32 wird nicht mehr weiterentwickelt!



## 6.3 NTFS Dateisystem

- Dateisystem für **Windows NT**, optional auch für **Windows XP**
- einige Unterschiede zum FAT32 Dateisystem:
  - Unterstützung mehrerer **Benutzer** und **Gruppen** mit umfangreichen Zugriffsrechten
  - jede Partition wird als **Volume** bezeichnet und besteht aus einer linearen Sequenz von **Clustern** (mit z. Zt. 512, 1024, 2048 oder 4096 Byte)
  - Adressierung eines Clusters erfolgt über **64-Bit Cluster-Nummern**
    (⇒ sehr große Dateien möglich)
  - zentrales Element der Dateiorganisation ist die *Master File Table* (**MFT**), die für jede Datei einen Eintrag enthält
  - Unterstützung von *Hard Links*
  - Dateien können automatisch komprimiert abgespeichert werden
  - Konsistenzüberprüfung mittels Journal-Datei



## 6.3 NTFS Dateisystem (2)

- Zugriffsrechte einer **NTFS-Datei**:
  - *no access* : kein Zugriff (`---`)
  - *list* : Anzeige von Verzeichnisinhalt erlaubt (`r--` )
  - *read* : Lesen und Ausführen von Dateien erlaubt (`rw-`)
  - *add* : Hinzufügen von Einträgen in einem Verzeichnis erlaubt (`-wx`)
  - *change* : Ändern und Löschen von Dateien erlaubt (`rwx`)
  - *full* : zusätzlich Ändern von Eigentümer und Zugriffsrechten erlaubt
- jede Datei wird eindeutig durch eine **64-Bit Dateireferenz** (*File reference*) bezeichnet; sie besteht aus:
  - **48-Bit Dateinummer** (*File ID*), die einen eindeutigen Index in der MFT darstellt
  - **16-Bit Folgenummer** (*Sequence ID*), die bei jeder Wiederverwendung der Dateinummer hochgezählt wird

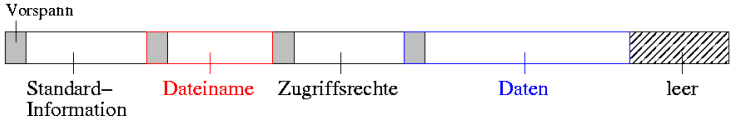
# 6.3 NTFS Dateisystem (3)

- jede Datei besteht aus mehreren **Strömen**, z.B.:
  - **Standard-Information** (zu MS-DOS kompatibler Dateiname sowie klassische MS-DOS Attribute wie Dateilänge, Zeitstempel, Typ, ...)
  - **Dateiname** (in Unicode mit 16-Bit Zeichen)
  - **Dateireferenz** (64-Bit Wert)
  - **Sicherheits-Beschreibung** (enthält **Eigentümer** und **Zugriffsrechte**)
  - **eigentliche Daten**
- Dateiorganisation erfolgt mittels *Master File Table* (**MFT**):
  - enthält **für jede Datei genau einen Eintrag**
  - Größe jedes Eintrags entspricht der Cluster-Größe
  - **Index** in Tabelle wird durch die **Datei-Nummer** festgelegt
  - Eintrag in Bootsektor verweist auf Beginn der MFT
  - ein Eintrag besteht aus **Hintereinanderreihung mehrerer Ströme**, die jeweils durch einen kurzen Vorspann (mit Länge, ...) eingeleitet werden

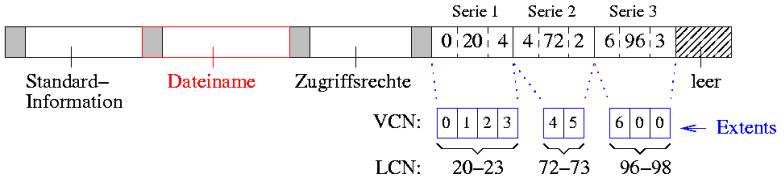


# 6.3 NTFS Dateisystem (4)

- Eintrag in MFT für eine **kurze Datei**:




- Eintrag in MFT für eine **lange Datei** (Beispiel):

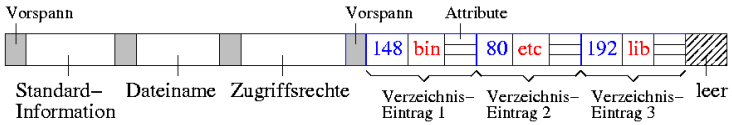



  - Daten befinden sich in Serien aus zusammengehörigen Clustern (**Extents**)
  - Zuordnung von virtuellen Cluster-Nummern (VCN) zu logischen Cluster-Nummern (LCN) wird als weiterer Strom gespeichert



# 6.3 NTFS Dateisystem (5)

- Eintrag in MFT für ein **kurzes Verzeichnis** (Beispiel):




  - Inhalt eines Verzeichnisses wird als eigener Strom gespeichert
  - jeder Verzeichniseintrag enthält **Dateireferenz**, **Dateiname** und einige ausgewählte **Attribute** (z.B. Dateilänge, Datum der letzten Modifikation)
  - Sortierung in lexikographischer Reihenfolge
- Eintrag in MFT für ein **langes Verzeichnis**:
  - Verzeichniseinträge werden nicht in MFT, sondern in separaten **Extents** gespeichert
  - Organisation als **$B^+$-Baum** ermöglicht eine schnelle Suche in großen Verzeichnissen



# 6.3 NTFS Dateisystem (6)

- die ersten 16 Dateien in der MFT sind **Metadateien**, die für das System reserviert sind:

| Index | Bedeutung |
|---|---|
| 0 | **MFT** |
| 1 | **Kopie der MFT** |
| 2 | **Journal-Datei** (protokolliert die Änderungen am Dateisystem) |
| 3 | **Volume-Informationen** (z.B. Name, Größe des Volumes) |
| 4 | **Attribut-Tabelle** (definiert erlaubte Ströme in den Einträgen) |
| 5 | Wurzelverzeichnis |
| 6 | **Cluster-Bitmap** (kennzeichnet alle freien und belegten Cluster) |
| 7 | *Bootloader* |
| 8 | *Bad Cluster List* (enthält die Indizes aller fehlerhaften Cluster) |
| 9 bis 15 | ... (reserviert für weitere Systemdateien) |
| 16 | erste Benutzerdatei |

# 7 Lernziele

- **Begriffe**: Zone Bit Recording, Hostadapter, Inode, Bitmap, Symbolic/Hard Link, Journaling, FAT, MFT, ...
- Aufbau einer **Festplatte**:
  - Organisation der Daten in Zylinder, Spuren und Sektoren
  - physikalische/logische Adressierung von Sektoren
  - Leistungsmerkmale und ihre Auswirkung beim Plattenzugriff
  - Unterschiede zu CD-ROM, DVD und Floppy
- Verständnis der Abläufe auf einem **Festplattenbus** (IDE, SCSI)
- Arbeitsweise und Vorteile von RAID-Systemen
- prinzipielle Arbeitsweise von **Dateisystemen** am Beispiel von **EXT2**, **FAT32** und **NTFS**, z.B.:
  - wie werden die Blöcke einer Datei adressiert?
  - wie erfolgt der Zugriff auf eine Datei mit gegebenem Pfadnamen?
  - wie werden freie Blöcke ermittelt?