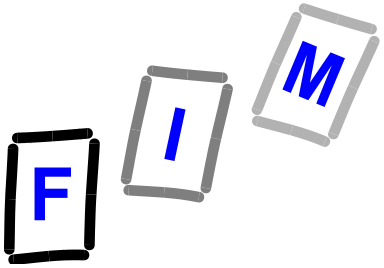
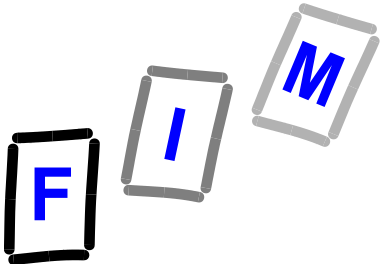


Betriebssysteme

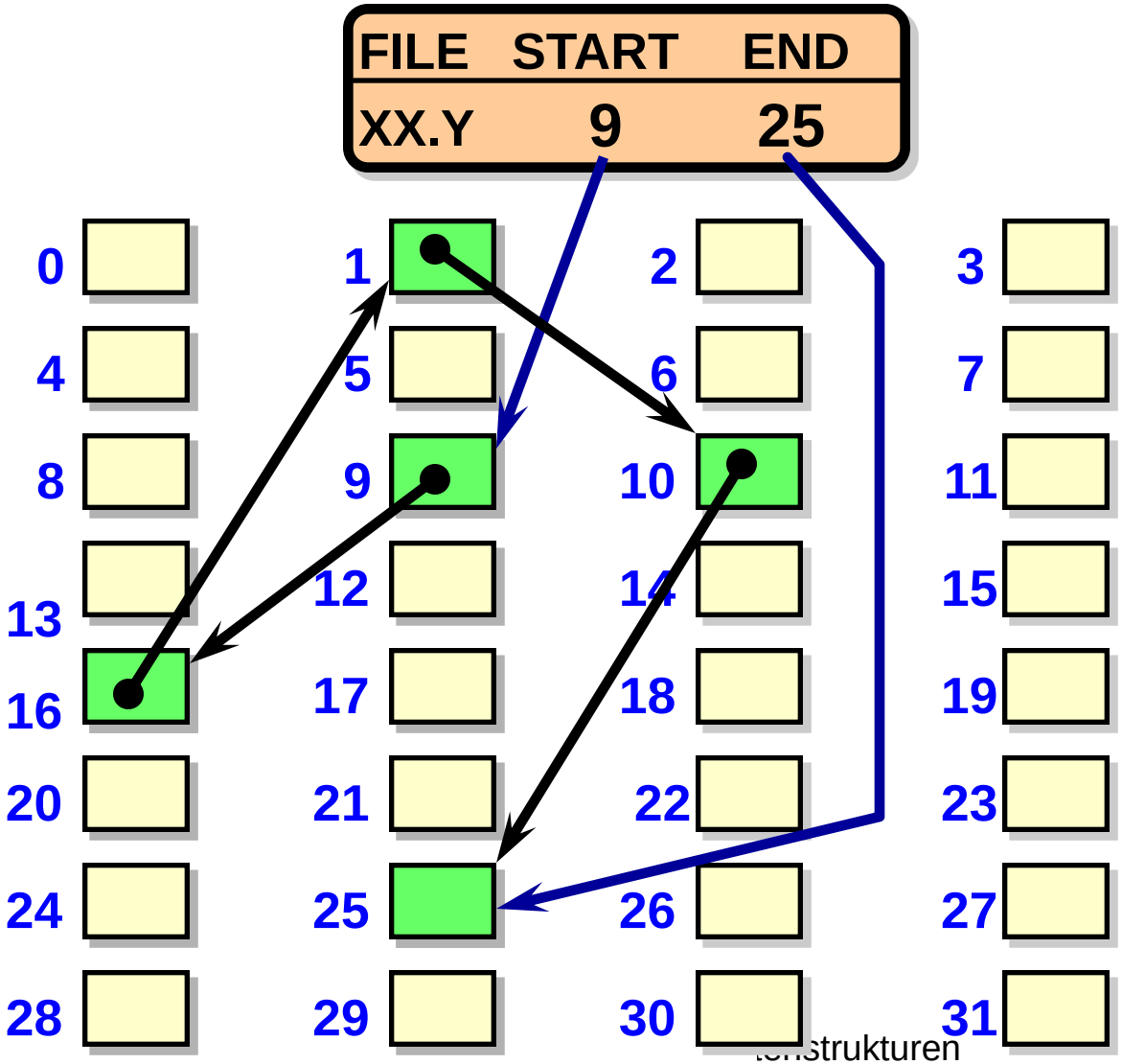
K_Kap11B: Files, Filesysteme Datenstrukturen

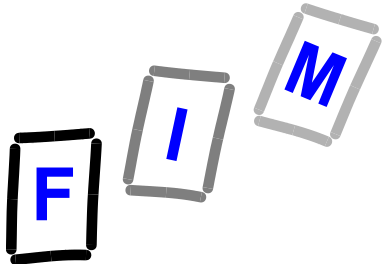


Files als lineare Liste

- File angeordnet als verkette Liste von Blöcken
  - ➔Jeder Block enthält Zeiger zum Nachfolger
  - ➔Zeiger = Adresse des Blocks
- Directory enthält Zeiger auf ersten Block
  - ➔Allenfalls auch auf letzten Block
- Fazit:
  Klassiche Implementierung einer Liste

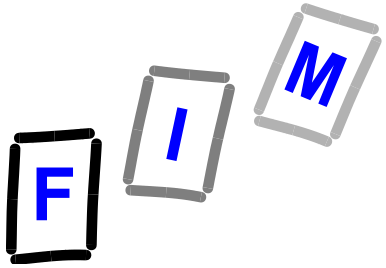
Files als lineare Liste





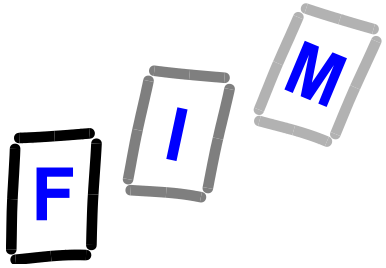
Files als lineare Liste Einzelschritte

- **create:**
  - ➔ **Neuen Eintrag im Directory eröffen**
  - ➔ **Zeiger auf NIL setzen**
  - ➔ **File-Anfangsgrösse: 0**
- **write**
  - ➔ **Finde eine freien Block auf Disk**
    - » **Dazu: Komponente des BS**
  - ➔ **Setze Zeiger dorthin**
  - ➔ **Schreibe in den bereitgestellten Block**



File Allocation Table (FAT)

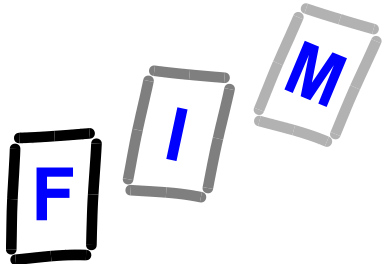
- **FAT Konzept: Einfache und effiziente Umsetzung von verketteten Listen**
- **Eingeführt bei MS-DOS und (wegen Rückwärtskompatibilität) in der Windows Familie und praktisch alles anderen Betriebssystemen**
- **Idee: Ergänze das Directory um ein gesondertes Verzeichnis (FAT) für jede Partition**
  - ➔ **Enthält die Nummern der zu einem File gehörigen Cluster**
  - ➔ **Zugriffsoptimierung: „cache FAT“ !**



File Allocation Table (FAT)

- FAT entspricht verketteter Liste

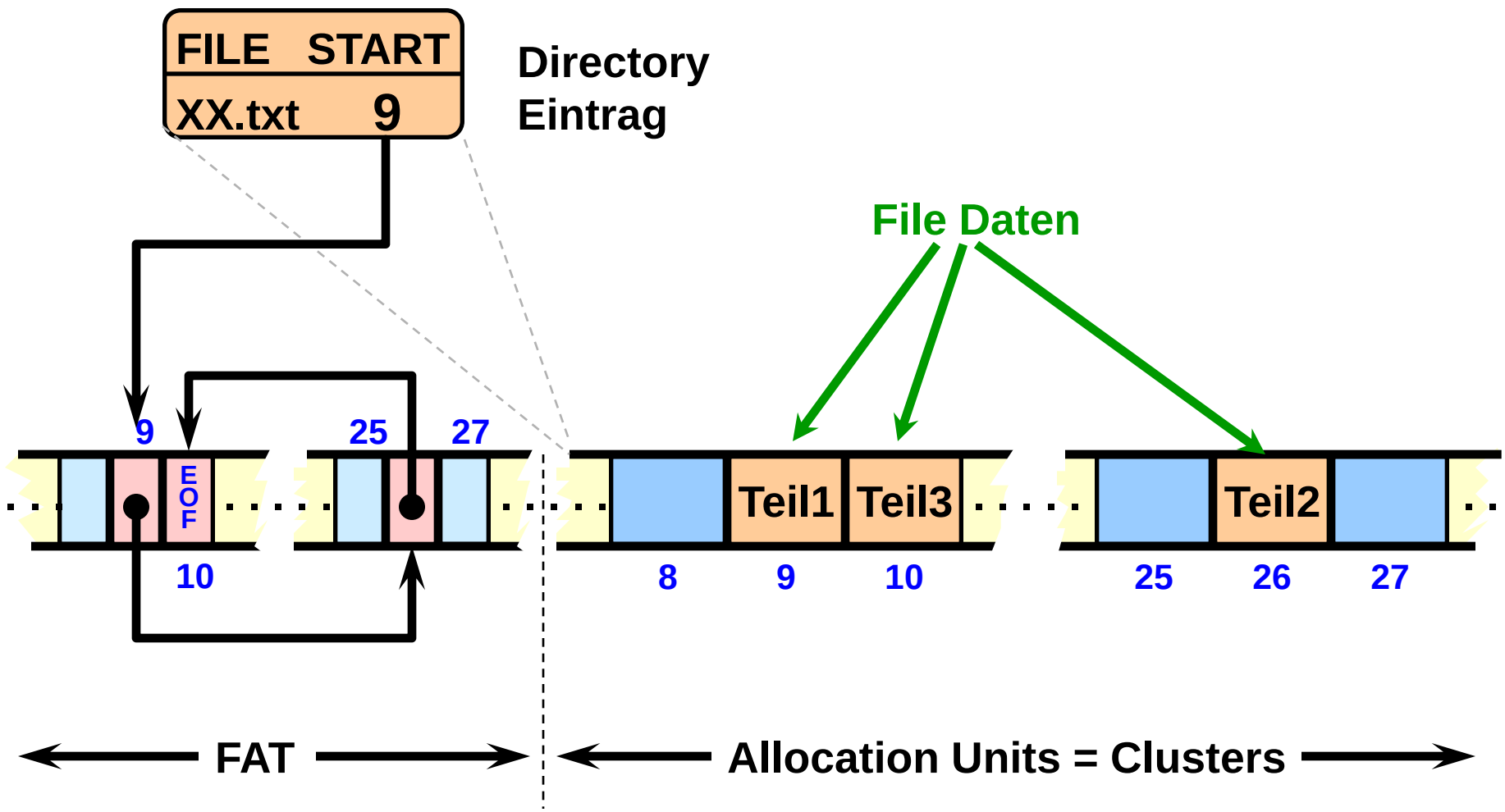
- FAT für jeden Cluster einen Eintrag
  - ➔ Frei: “0”
  - ➔ Letzter eines Files: “EOF” (z.B. $FFF8_{16}$)
  - ➔ Defekt: „BAD“(z.B. $FFF7_{16}$)
  - ➔ Ansonsten: Nummer des Folgecluster

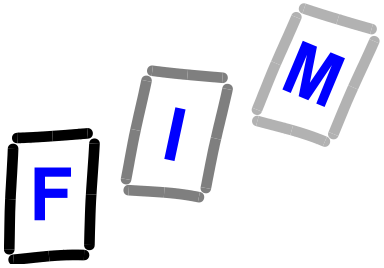


FAT: Directory-Einträge

- Jeder Directory Eintrag enthält Clusternummer des Startcluster des jeweiligen Files
- Außerdem Metadaten
  - ➔ Dateiname
  - ➔ Attribute wie read only, hidden, system-file
  - ➔ Zeitstempel
  - ➔ Dateigröße

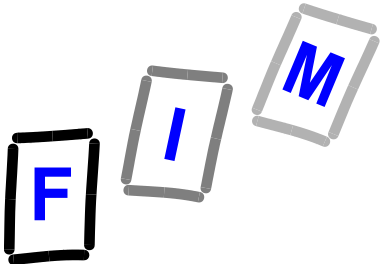
File Allocation Table (FAT)





FAT Versionen
(Auszug)

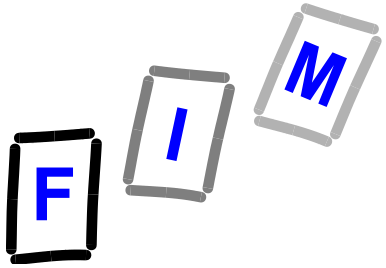
- **FAT 12**
  - ➔ **„8.3“ Dateinamen**
  - ➔ **12-Bit Clusternummern -> 2^{12} = 4096 Cluster**
  - ➔ **Clustergröße 512 bis 4096 Bytes**
  - ➔ **Partitionen bis 16 MB möglich**
  - ➔ **Typisch für 3,5 Zoll Disketten**
- **FAT 16**
  - ➔ **„8.3“ Dateinamen**
  - ➔ **2^{16} - 12 = 65524 Cluster (12 speziell reserv.)**
  - ➔ **Clustergröße 512 bis 32 KByte**
  - ➔ **Partitionen bis 2GB (Windows-NT bis 4GB)**



FAT Versionen

(Auszug)

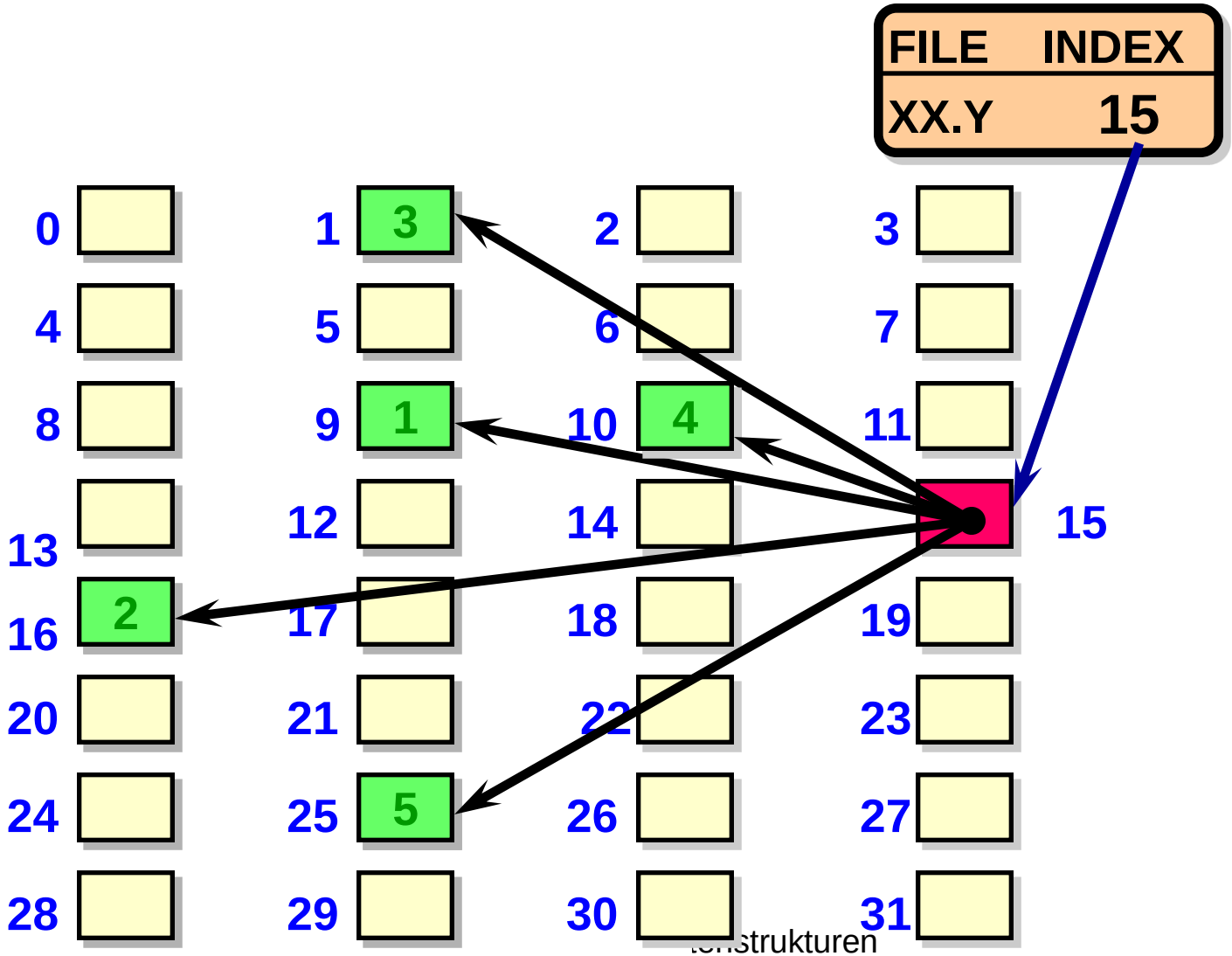
- VFAT (Virtual File Allocation Table)
  - ➔ Erweiterng von FAT 16
  - ➔ Lange Dateinamen
  - ➔ Unicode
- FAT 32
  - ➔ 32 Bit (28 verwendet; 2^{18} Cluster), 268.435.456 Cluster
  - ➔ Clustergröße 512Bytes bis 32KB
  - ➔ Dateigröße max 4GB (-1 Byte)
  - ➔ Partitionsgröße bis ca 8TB (im Prinzip)
    - » Format unterstützt nur 32GB Dateisysteme
    - » Gesonderte Tools für größere erforderlich
  - ➔ Rootverzeichnis kann variabel groß sein
- exFAT für Flashspeicher
  - ➔ Unter Windows CE 6.0 eingeführt
  - ➔ VISTA unterstützt exFAT

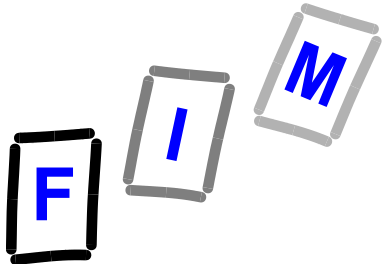


Indizierte Anordnung: Grundidee

- Anstatt einer blockweisen Verkettung werden alle Zeiger in einem **Index** gehalten
  - ➔ Jeder File hat eigenen Index
  - ➔ Index selbst ist ein Block (Cluster) auf dem Massenspeicher
- Zeiger vom Directory zum Indexblock
- Der j-te Eintrag im Index-Block zeigt auf den j-ten Block (Cluster) des Files

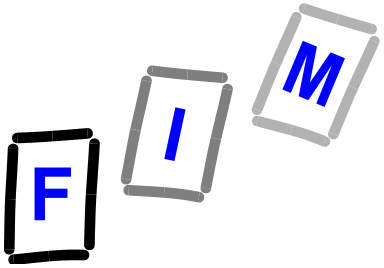
Speicherung mit Index





Index
Spezielle Probleme

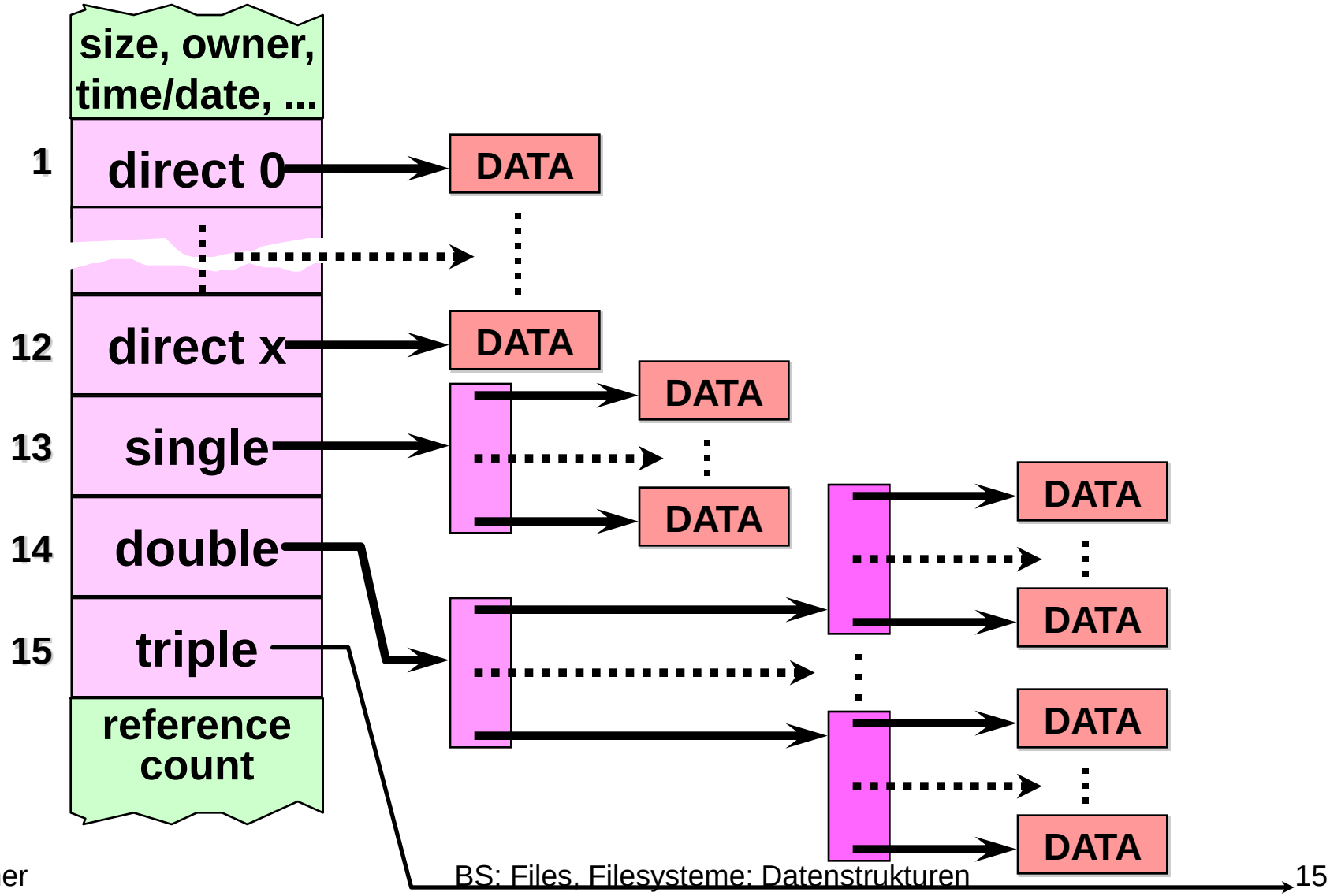
- **Wird ein File groß, enthält er viele Blöcke**
  - **➔Blöcke haben feste Größe**
  - **➔Auch der Indexblock selbst**
- **Großer Indexblock:**
  - **» Platzverschwendung für kleine Files**
- **Kleiner Block:**
  - **➔BS kann keine großen Files verwalten**

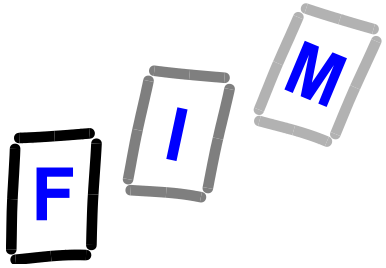


UNIX
Verwendet “Inodes”

- **Jedem File ein spezieller Indexblock zugeordnet: Kombination der Index-Idee**
  - ➔ ***Inode***
    - » ***Inodes als Liste angeordnet***
    - » ***Länge der Liste steht im „Superblock“***
      vgl.: Partition Boot Record PBR
  - ➔ **Enthält Metadaten**
  - ➔ **Enthält Zeiger auf File-Blöcke und Zeiger auf indirekte Blöcke**
    - » **(typisch) 15 Zeiger in einem Inode**
    - » **12 davon zeigen direkt auf Blöcke des Files,**
      Gut geeignet für kleine Files
    - » **die nächsten 3 Zeiger verweisen auf Indexblöcke (“Indirekte Blöcke”)**
      der erste auf einen “single indirect block”
      der zweite auf einen “double indirect block”
      der dritte auf einen “ triple indirect block”

UNIX/Linux INODE





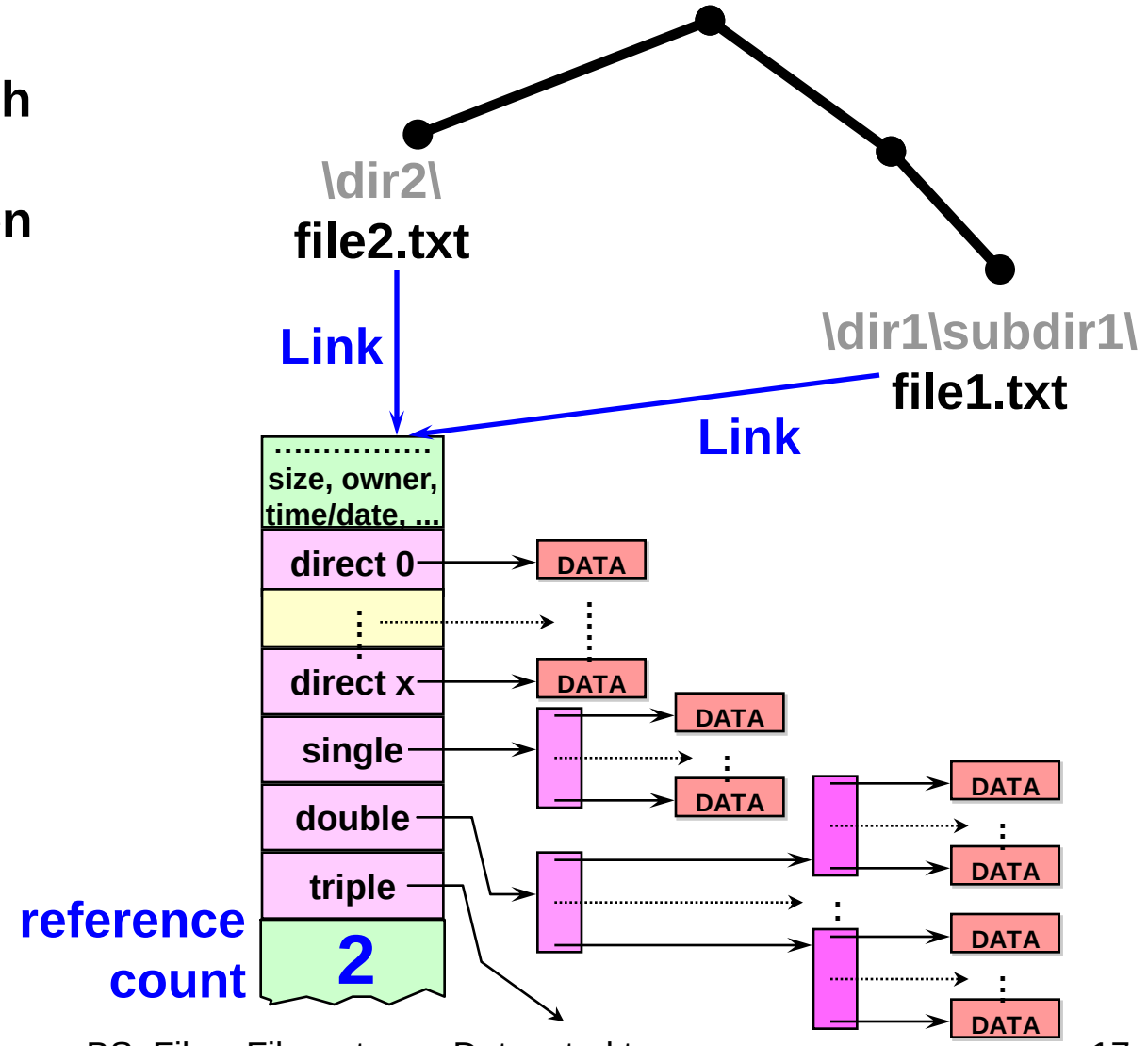
Verzeichnis in UNIX/Linux

- Datei mit festem Format
- Einträge:
  - ➔Dateiname
  - ➔Länge der Datei
  - ➔Inode- Nummer (4 Bytes)
  - ➔Zeiger auf nächsten Eintrag

Links
Reference counter

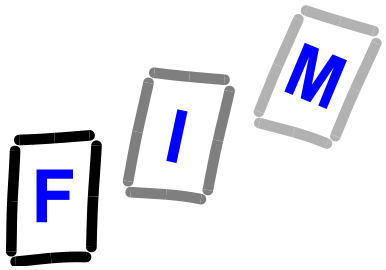
n File – Einträge (auch aus verschiedenen Directories) verweisen auf den selben File (link auf Inode)

Erst wenn referencecount =0 kann File wirklich gelöscht werden



Globale Struktur eines UNIX Filesystems

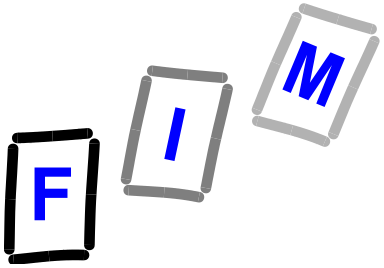
- **Eingebettet in eine aktive Partition folgender logischer Aufbau:**
  - ➔ **Bootblock: Startroutine des BS**
  - ➔ **Superblock: Allgemeine Verwaltungsinformation**
  - ➔ **Inode Bitmap: Benutzt/nicht benutzt**
  - ➔ **Cluster-Bitmap: Für jeden Block „frei“/“belegt“**
  - ➔ **Inode Tabelle: Alle Inodes**
  - ➔ **Cluster: Eigentliche Nutzdaten**
    - » **Dateien und Verzeichnisse**



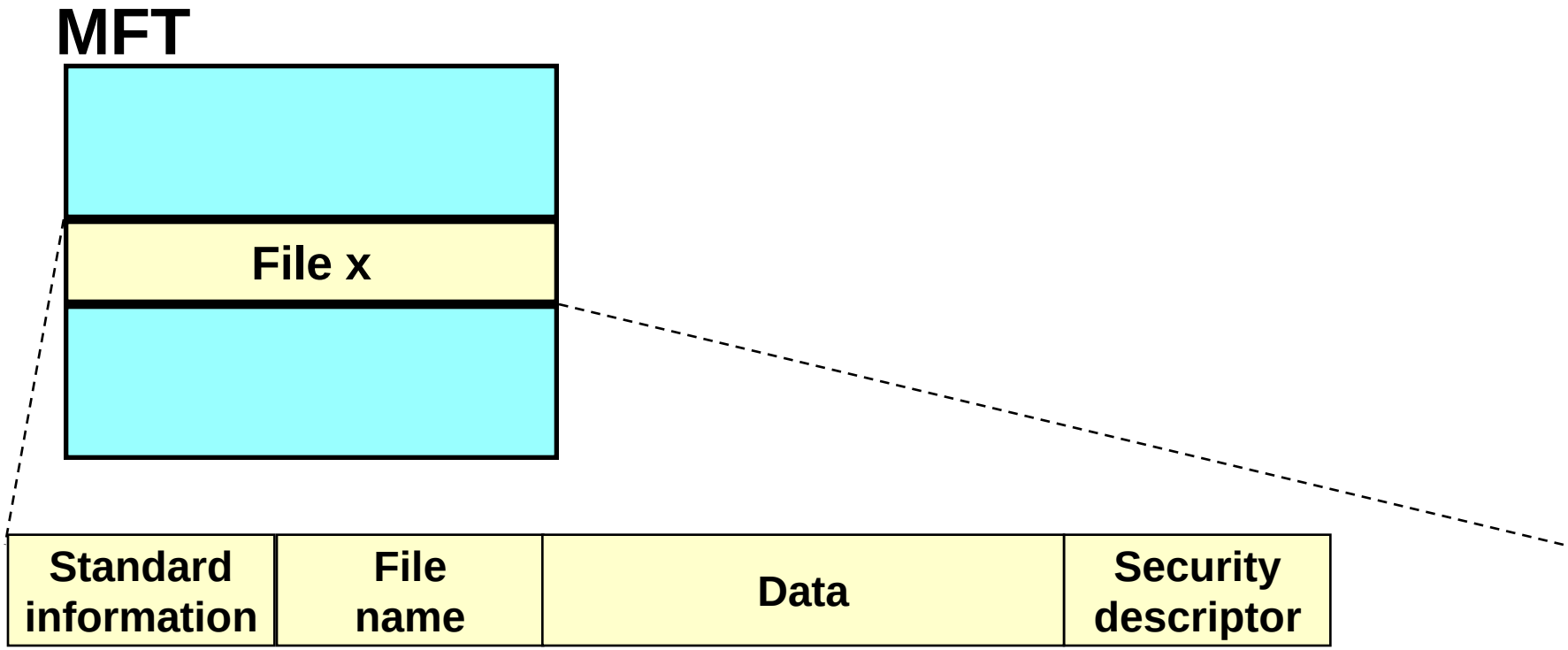
NTFS-Modell MFT

- **Verwaltung der Disk (Volume) erfolgt über eine relationale Datenbank:**
  Master File Table (MFT)

- **MFT: Zeilen von Eintragungen**
  - ➔ **Eine Zeile pro File oder Directory Eintrag**
  - ➔ **Für MFT selbst ist Zeile 0 vorgesehen**
  - ➔ **Die Spalten sind die Attribute des File**
    - » **Die (bisher gewohnten Daten) sind nur ein spezielles Attribut**
  - ➔ **Größe je Eintrag: 2KB**

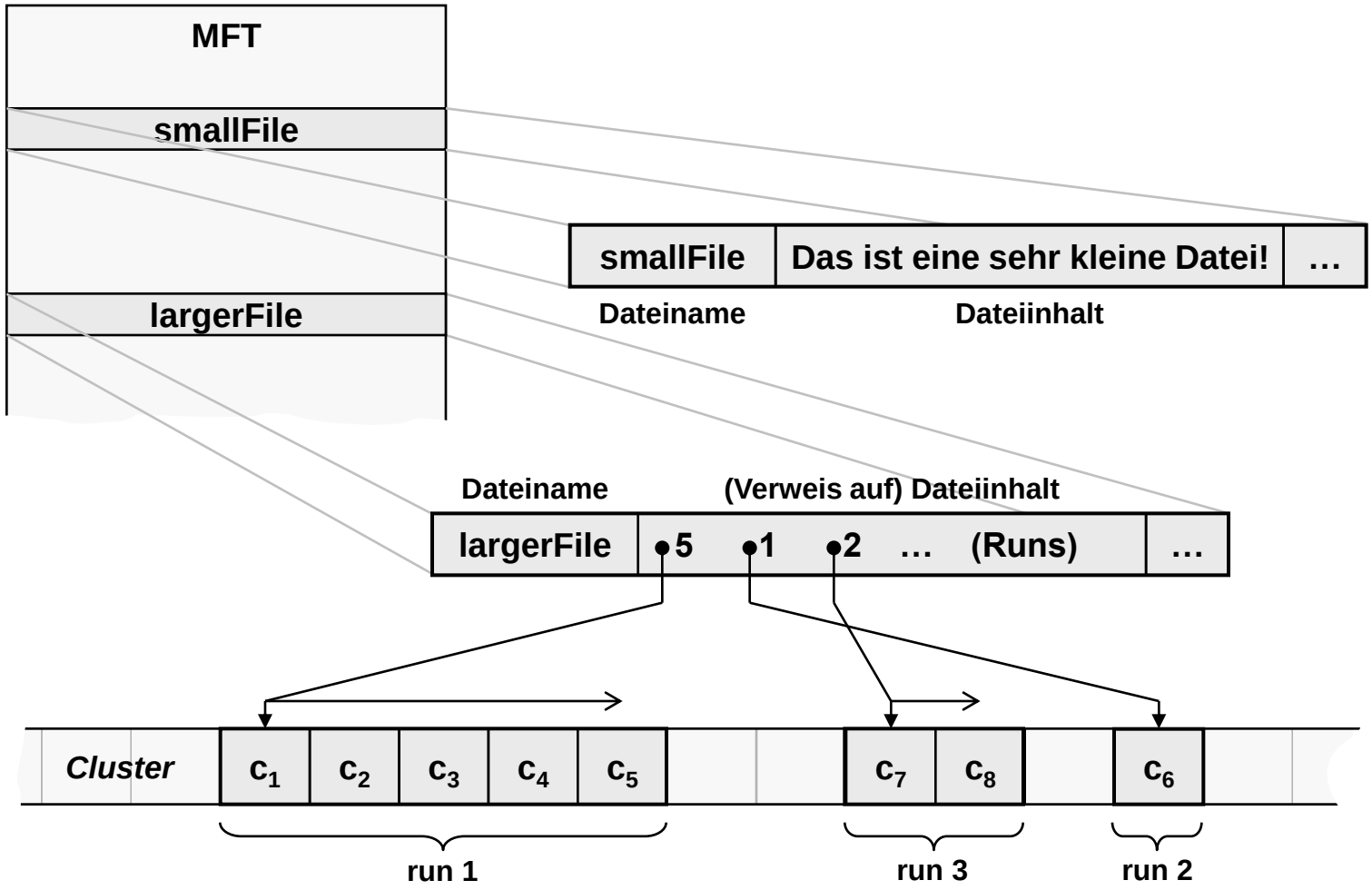


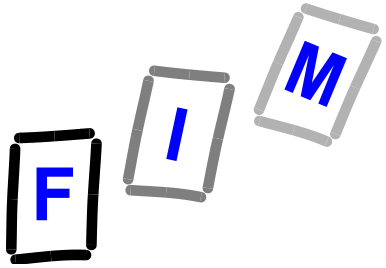
MFT Struktur allgemein



EAs = Extended Attributes

Kleine und große Files

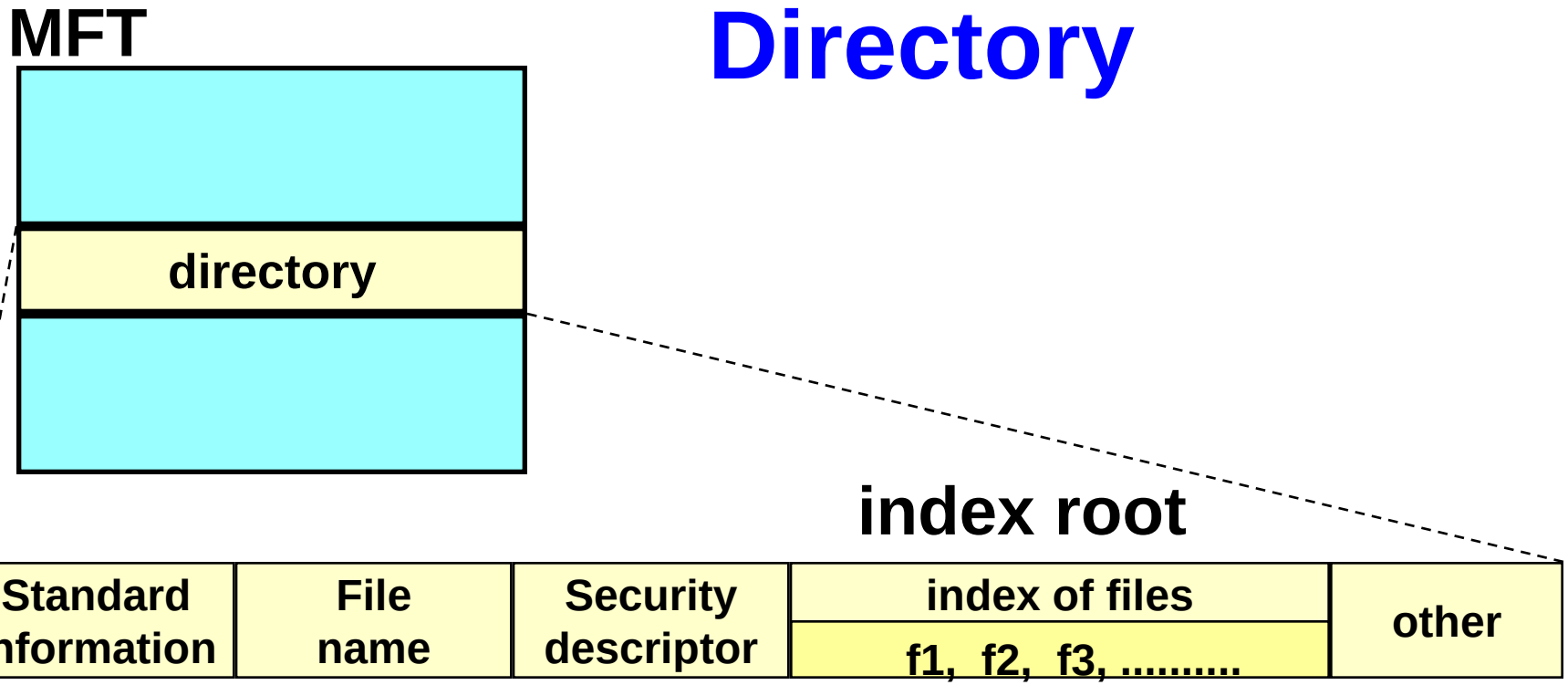




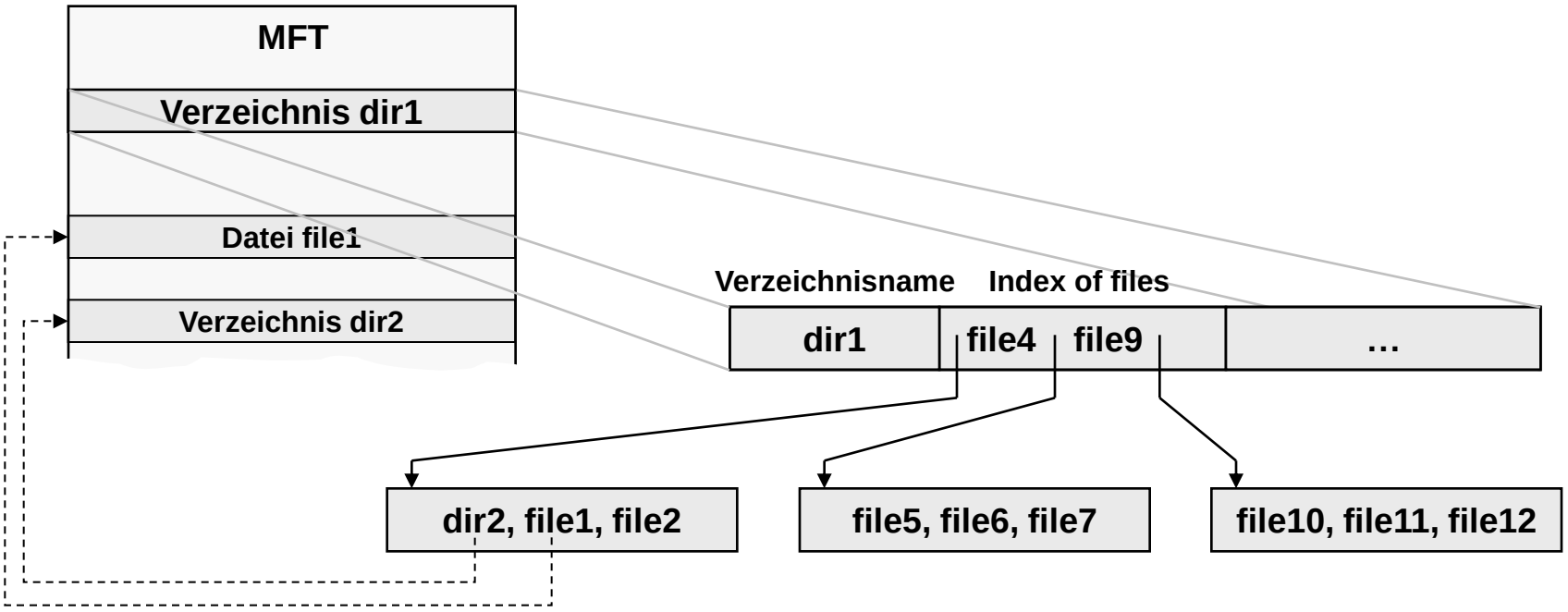
NTFS-Modell MFT

- File kann mehrere Namen haben:
  - ➔long Name, MS-DOS name
- MFT enthält auch die Einträge für directories
  - ➔Demnach entspricht einem (sub-) directory wieder eine Zeile im MFT

MFT Struktur Eintrag für kleines Directory



Großes directory mit Unterverzeichnis



Organisation als B-Baum