In diesem Abschnitt werden folgende Themen betrachtet:

- Einführung
- Dateisysteme
  - Definition des Dateisystems
  - Beispiele der Dateisysteme
  - Arten der Dateisysteme
  - Physische Struktur des Dateisystems
  - Logische Struktur des Dateisystems
  - Gängige Dateisysteme FAT, NTFS, ext
- Backup
- Administration
- Sicherheit

Eine der Aufgaben des Betriebssystems ist die Datenverwaltung. Der Benutzer verwendet in der alltäglichen Arbeit vielen Informationen unterschiedlicher Art:

- Textdokumente
- Bilder (Grafik)
- Musik
- Videos
- Präsentationen (Folien)
- Elektronische Tabellen
- Mails
- HTML-Dokumente

Um diese Informationen aufbewahren, wieder finden und verarbeiten zu können, gibt es prinzipiell zwei Arten der Speicherung:

- Dateisysteme
- Datenbanken

Beide Arten werden momentan in der Informationstechnik sehr intensiv verwendet, wobei die Datenbanken eine steigende Tendenz haben.

Der Unterschied zwischen Dateisystemen und Datenbanken liegt in der Zuständigkeit der Verwaltung der Informationen und in dem Funktionsumfang.

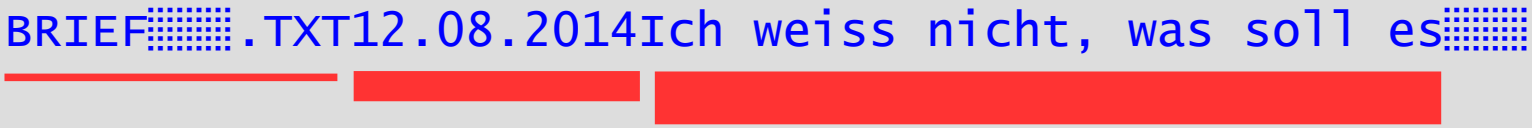
Im Falle der Dateisysteme sind das Betriebssystem und seine Dateisystem-Module für die Verwaltung der Informationen zuständig. Die Informationen sind als Dateien organisiert und befinden sich auf dem Datenträger. Jede einzelne Datei kann von dem Betriebssystem gespeichert, gelöscht, geöffnet werden.

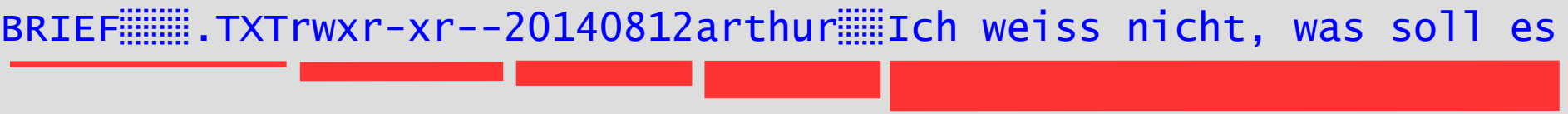
Definition: Das Dateisystem organisiert das Abspeichern von neuen und geänderten Dateien, die Suche von Dateien, sowie das Löschen und das Umbenennen der Dateien.

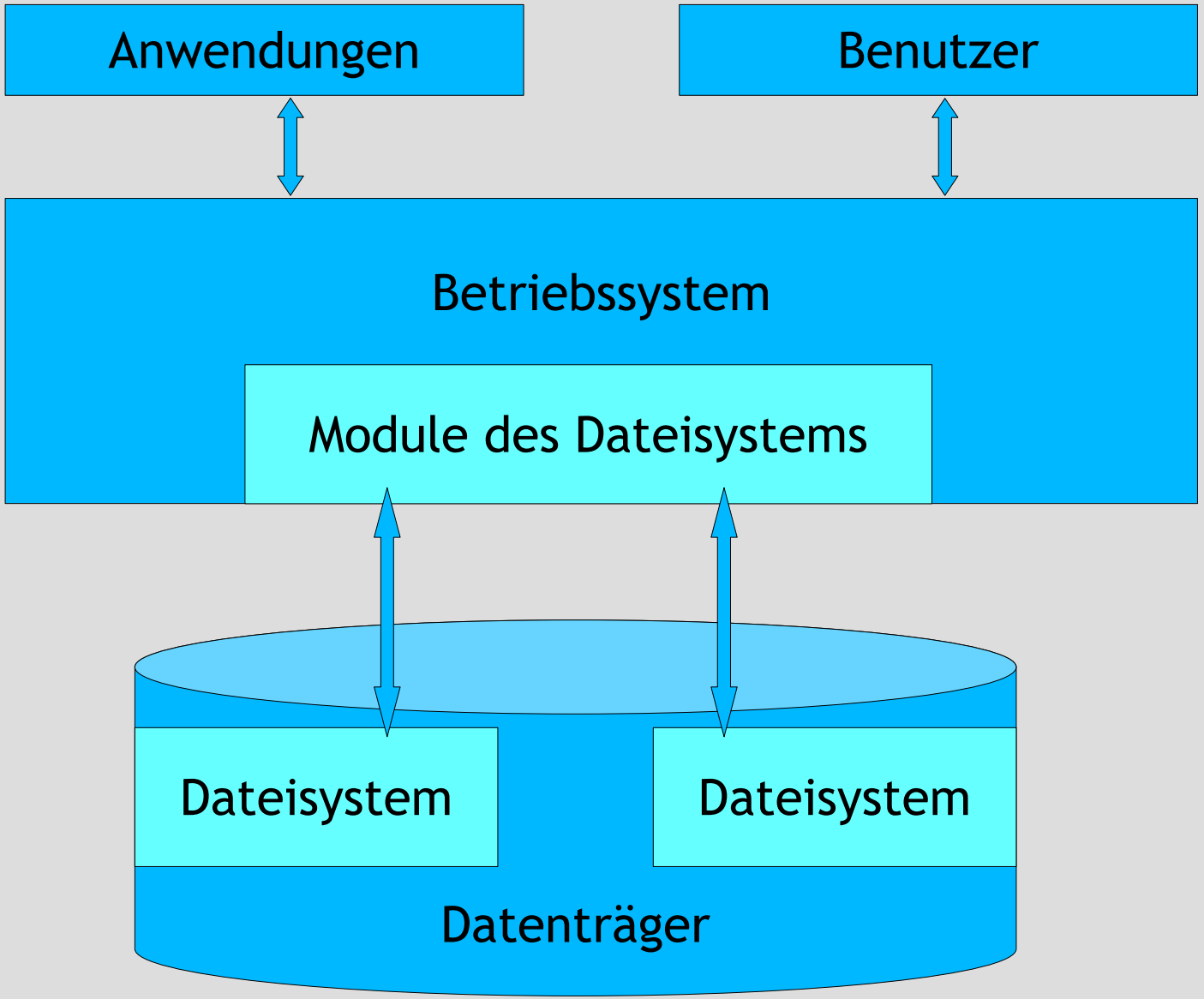
Definition: Das Dateisystem stellt die Regeln dar, wie die Daten gespeichert werden.

Für jeden Informationstyp gibt es einen Dateityp.

Beispiele der Organisation der Datenspeicherung als Dateien:







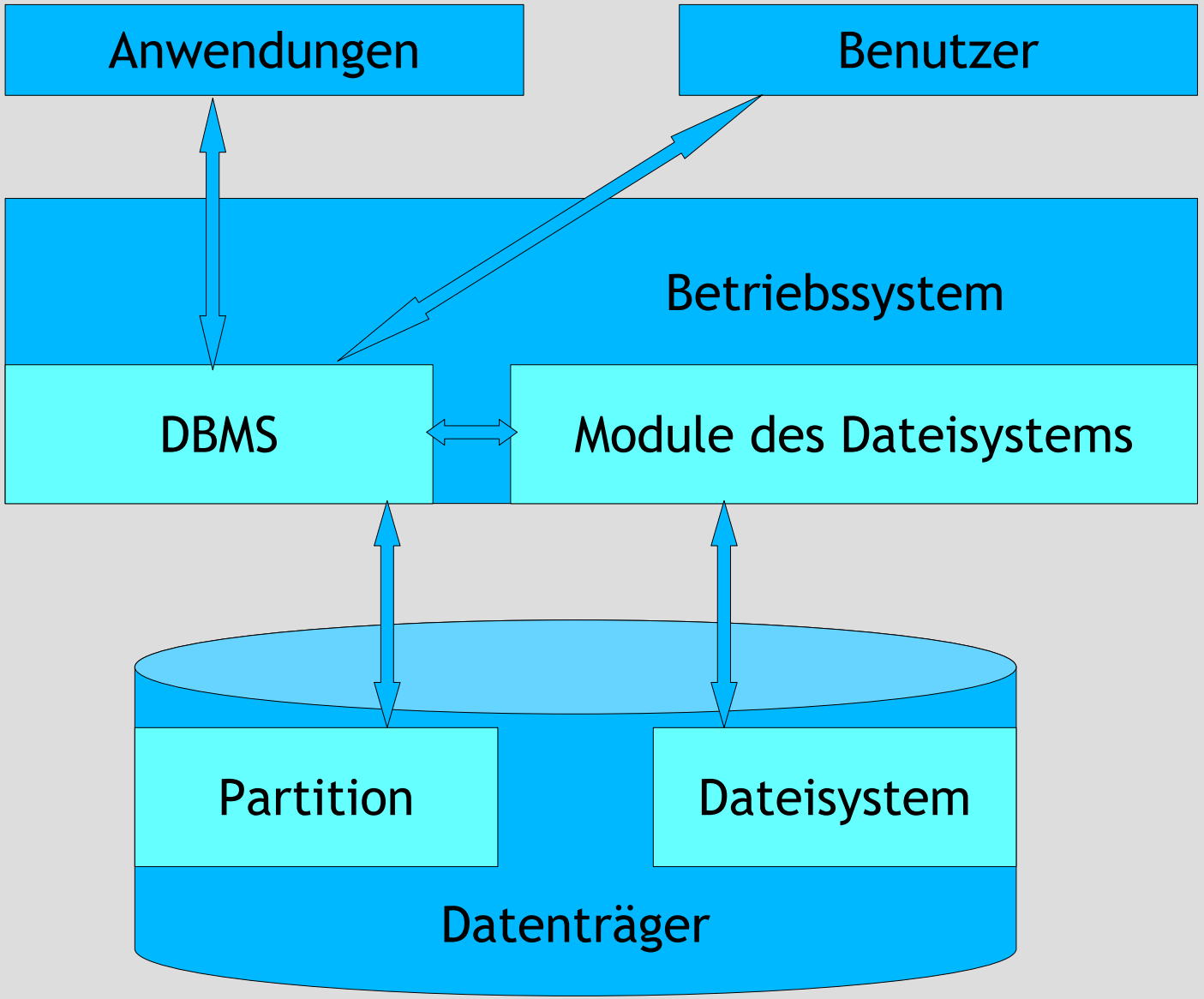
Art der Speicherung: Dateisystem

Im Falle der Datenbank ist ein Teil der Datenbank namens DBMS (Database Management System) für die Informationsverwaltung verantwortlich. Mehrere Informationseinheiten werden in einer Datei oder in mehreren Dateien nach den proprietären Standards gespeichert. Somit besteht die Datenbasis aus diesen Dateien, aber das Betriebssystem kann auf die Informationen in den Dateien nicht zugreifen.

Oft verwendet die Datenbank für die Speicherung der Informationen gar keine Dateien, sondern verwaltet den Datenträger selbst.
In dem Fall hat das Betriebssystem überhaupt keine Kontrolle über die gespeicherten Informationen.

Die Anwendung (oder Benutzer) haben eine Schnittstelle zur Datenbank und können somit die Informationen abrufen, auswerten und verarbeiten.

Die Datenbanken haben den Vorteil, dass sie die Daten ausfiltern, auswerten, verarbeiten und präsentieren können.



Art der Speicherung: Datenbank

Dateisystem	Betriebssystem	Rechte	Eigentum	Kontingent
FAT	MS Windows Linux	nein	nein	nein
NTFS	MS Windows Linux (lesen)	ja	ja	ja
ext	Linux	ja*	ja*	nein*
Reiser FS	Linux	ja*	ja*	nein*
xfs	Linux	ja*	ja*	nein*

* — Die Eigenschaft ist nicht vollständig oder fehlt, kann aber installiert werden.

Man kann folgende Arten der Dateisysteme unterscheiden:

- Lokale Dateisysteme. Diese Systeme befinden sich auf dem selben Rechner, wo das Betriebssystem läuft.
- Netzwerkdateisysteme. Diese Systeme befinden sich auf einem anderen Rechner. Lokales Betriebssystem kommuniziert mit dem Betriebssystem des anderen Rechners, um die Daten auszutauschen. Beispiel — Thin Clients.
- Verteilte Dateisysteme. Das ist eine Weiterentwicklung der Netzwerkdateisysteme. Der Client muss nicht wissen, wo sich das entfernte (remote) Dateisystem befindet.
- Virtuelle Dateisysteme. Es wird eine große leere Datei angelegt. Innerhalb dieser Datei wir ein Dateisystem erstellt, das vollständig die Charakteristiken eines lokalen Dateisystems hat. Ein virtuelles Dateisystem wird in dem System angebunden und ganz normal benutzt.

In Bezug auf Zugriffsart zeichnen sich folgende Dateisysteme aus:

- Dateisysteme mit dem linearen Zugriff (Bandlaufwerk, veraltet).
- Dateisysteme mit dem wahlfreien Zugriff (Festplatte, modern).

Sollte eine Datei in dem Dateisystem mit dem linearen Zugriff gelesen werden, dann muss man zuerst alle Dateien von Anfang an Byte für Byte (oder Block für Block) lesen.

Vorteil — sehr einfaches System.
Nachteil — sehr zeitaufwendiges System.

Das Auffinden einer Datei in dem Dateisystem mit dem wahlfreien Zugriff passiert durch die Angabe der physischen Adresse der Datei, z.B. Zylinder/Kopf/Sektor (CHS) oder logische Blockadressierung (LBA).

Vorteil — sehr schnelles System.
Nachteil — der Datenträger muss über eigenen Controller verfügen.

Die physische Struktur des Dateisystems bilden die Blöcke (Data Cluster) — die zusammenhängenden Bereiche des Datenträgers, die aus Bytes bestehen. Viele Datenträger können mehrere Dateisysteme beinhalten.

In einem Block wird entweder eine Datei (falls sie klein genug ist) oder ein Teil einer Datei (falls die Datei zu groß ist) platziert. Ausnahme — NetWare File System.

Die typischen Größen eines Blocks sind: 2, 4 bis 32 KiB. Viele Dateisysteme erlauben dem Benutzer, die Größe des Blocks vor der Formatierung festzulegen.

Die Größe des Blocks hat hat einen starken Einfluss auf die Performance des Dateisystems:

- Große Dateien + kleine Blöcke = Zeitaufwand.
- Kleine Dateien + große Blöcke = Platzverschwendung.

Die Blöcke des Datenträgers können im Laufe des Betriebs unbrauchbar werden. Es wird normalerweise nur dann festgestellt, wenn das Betriebssystem versucht, die Daten in einen fehlerhaften Block zu schreiben.

Die Software des Datenträgers markiert den fehlerhaften Block, sodass keine Daten künftig dahin geschrieben oder davon gelesen werden.

Alle Schreib- und Leseoperationen werden in einen fehlerfreien Block in dem speziell dafür vorgesehenen Bereich auf der Festplatte (Hot Fix) umgeleitet.

Das Betriebssystem und die Anwendungen kriegen nicht mit, dass derartige Fehler auf der Festplatte passieren.

Die Fragmentierung eines Datenträgers (oder einer Datei) kommt bei jedem Betriebssystem vor, wenn die Daten im Laufe der Zeit mehrmals geändert werden. Eine Datei besteht aus den Blöcken, die weit über die Festplatte verstreut sind, mit anderen Dateien oder Lücken dazwischen.

Die Fragmentierung hat im allgemeinen einen negativen Einfluss auf die Performance des Dateisystems, aber in allen modernen Dateisystemen hält sich in Grenzen dank den vorhandenen Gegenmechanismen. Außerdem laufen viele schreibende und lesende Prozesse in modernen Betriebssystemen gleichzeitig, deswegen wird keine Datei am Stück von der Festplatte gelesen oder auf die Festplatte geschrieben.

Jedes Betriebssystem hat eigene Methoden für Defragmentierung des entsprechenden Dateisystems.

Die Defragmentierungsmechanismen der Betriebssysteme:

- Das Dateisystem ext von Linux schreibt die Daten nicht in einzelnen Blöcke, sondern bildet die Gruppen der Blöcke. Am Ende jeder Gruppe lässt es genug Platz, um die künftige Änderungen zu speichern.
- Das Dateisystem xfs von Linux/UNIX versucht die ganze Datei vor dem Schreiben in dem Arbeitsspeicher zu konsolidieren, um deren Größe zu berechnen. Dadurch kann passender Platz für die Datei besser gefunden werden.
- Die Dateisysteme NTFS von MS Windows führt die Defragmentierung ab und zu automatisch durch.
- Jedes Betriebssystem/Dateisystem hat Befehle für die manuelle Defragmentierung.

Die logische Struktur des Dateisystems bilden die Dateien, Verzeichnisse und andere Speichereinheiten für spezielle Zwecke.

Grundlage des Dateisystems bilden die "normalen" Dateien, in denen die Dokumente, Bilder, Musik, Programme, Systembefehle u.s.w. abgespeichert sind.

In den ersten Betriebssystemen (CP/M, Apple DOS) befanden sich die Dateien alle zusammen in einem logischen Bereich, z.B. es existierten keine zwei Dateien mit dem selben Namen.

Der moderne Computer enthält viel mehr Dateien. Es gibt Notwendigkeit, die Dateien logisch von einander zu trennen, z.B. die Systemdateien gesondert von den Bildern oder Musik abzuspeichern.

Die modernen Dateisysteme verfügen über solche getrennte logische Bereiche — Verzeichnisse, Ordner. Jetzt können schon mehrere Dateien mit dem selben Namen existieren, aber in unterschiedlichen Verzeichnissen.

Eigentlich ist ein Verzeichnis nur eine besondere Art der Datei, die eine Liste der Dateien beinhaltet. Es kann aber nicht mit den Befehlen, die für "normalen" Dateien vorgesehen sind, bearbeitet werden.

Außerdem können die Verzeichnisse die weiteren Verzeichnisse (Unterverzeichnisse) enthalten.

Durch solche logisch getrennte Bereiche (Verzeichnisse) lässt sich eine baumartige Struktur aufbauen, wo die Daten hierarchisch abgespeichert sind.

Alle modernen Dateisysteme verfügen über solche hierarchische Strukturen, die teilweise vordefiniert und festgelegt sind, teilweise sich von dem Benutzer ändern lassen.

Einige Dateisysteme (NTFS) trennen die Speicherbereiche für Dateien und für Verzeichnisse physikalisch von einander, dadurch kann bessere Performance erreicht werden.

Noch eine besondere Art der Dateien sind die Links, die keine Daten sondern nur einen Verweis auf eine "normale" Datei enthalten. Die Links gibt es in allen Linux/UNIX-Betriebssystemen und in modernen MS Windows Betriebssystemen.

Die Links finden wichtige Einsätze in der Administration, im Sicherheitsbereich, in der Programmierung der Client-Server-Anwendungen sowie für die Kompatibilität der Anwendungen. In der täglichen Arbeit eines "normalen" Benutzers spielen sie kaum eine Rolle.

Der Sinn der Links ist die Flexibilität: ein und der selbe Inhalt kann unter verschiedenen Namen aufgerufen und geändert werden.

In den Linux/UNIX-Betriebssystemen/Dateisystemen werden die Links sehr intensiv eingesetzt.

Die logische Struktur des Dateisystems kann bei den Fehlern auf der Festplatte beschädigt werden. Dadurch können viele Dateien nicht mehr erreichbar werden oder das ganze Dateisystem nicht mehr funktionsfähig sein.

Das Journaling (Eigenschaft des Dateisystems) verhindert solche Verluste, aber nicht die Verlust einzelner Datei.

Das Prinzip des Journaling besteht darin, dass alle Änderungen an Meta-Daten der Datei zuerst in einem besonderen Bereich des Datenträgers beschrieben werden, und dann ausgeführt. Selten werden auch die Änderungen an dem Inhalt der Datei auch gespeichert (Full-Journaling) — das erlaubt die Datei vollständig wiederherzustellen, nimmt aber viel Zeit in Anspruch bei jeder Änderung der Datei.

Das Dateisystem FAT (File Allocation Table) ist das älteste und das einfachste und findet immer noch den Einsatz in unterschiedlichen Betriebssystemen.

Im Laufe der Zeit entstanden folgende Versionen von FAT:

- FAT12 auf den Disketten.
- FAT16 auf den Festplatten bis zu 4 GiB.
- FAT32 auf den Festplatten ab 4 GiB aufwärts.
- exFAT auf den Flash-Datenträgern, Festplatten bis 512 TiB.
- TFAT – Transactional FAT. Transaktion ist ein Satz von Aktionen, die entweder alle erfolgreich ausgeführt werden müssen, oder gar keine Aktion aus dem Satz.
- VFAT– Virtual FAT, ein Treiber, der die langen Dateinamen (bis 255 Symbole) neben den kurzen (8.3-Regel) auf den alten FAT-Versionen erlaubt.

Wichtige Eigenschaften von FAT:

- Datei wird durch den Namen eindeutig identifiziert.
- Keine Zugriffsberechtigungen auf Dateien/Verzeichnisse.
- Kein Besitz von Dateien/Verzeichnissen.
- Keine Kontingente in dem Dateisystem.
- Keine Protokollierung der Zugriffe.

Das Dateisystem FAT besteht aus:

- File Allocation Table (FAT, Zuordnungstabelle) — 0,58 %.
  Das ist wirklich eine Tabelle mit mehreren Einträgen. Anzahl der Einträge ist gleich der Anzahl der Blöcke in der Datenzone. Die Länge von einem Eintrag ist 12, 16 oder 32 Bits. Jeder Eintrag ist fest einem Block in der Datenzone zugeordnet. Inhalt jedes Eintrags informiert, ob der entsprechende Block in der Datenzone frei oder mit Daten gefüllt ist und wo der nächste Block zu finden ist.
- Hauptverzeichnis — 0,58 %. Hauptverzeichnis befindet sich in einem festgelegten Ort (physische Adresse).
- Datenzone — 98,79 %. Sie beinhaltet die Daten in den Blöcken.

Inhalt eines Eintrages im FAT-Dateisystem, hexadezimal		
FAT16	FAT32	Bedeutung
0	0	Entsprechender Block ist frei.
FFF7	FFFF FFF7	Entsprechender Block ist fehlerhaft.
FFF8-FFFF	FFFF FFF8 - FFFF FFFF	Der letzte Block der Datei.
Sonst	Sonst	Die Nummer des nächsten Blocks mit den Daten.

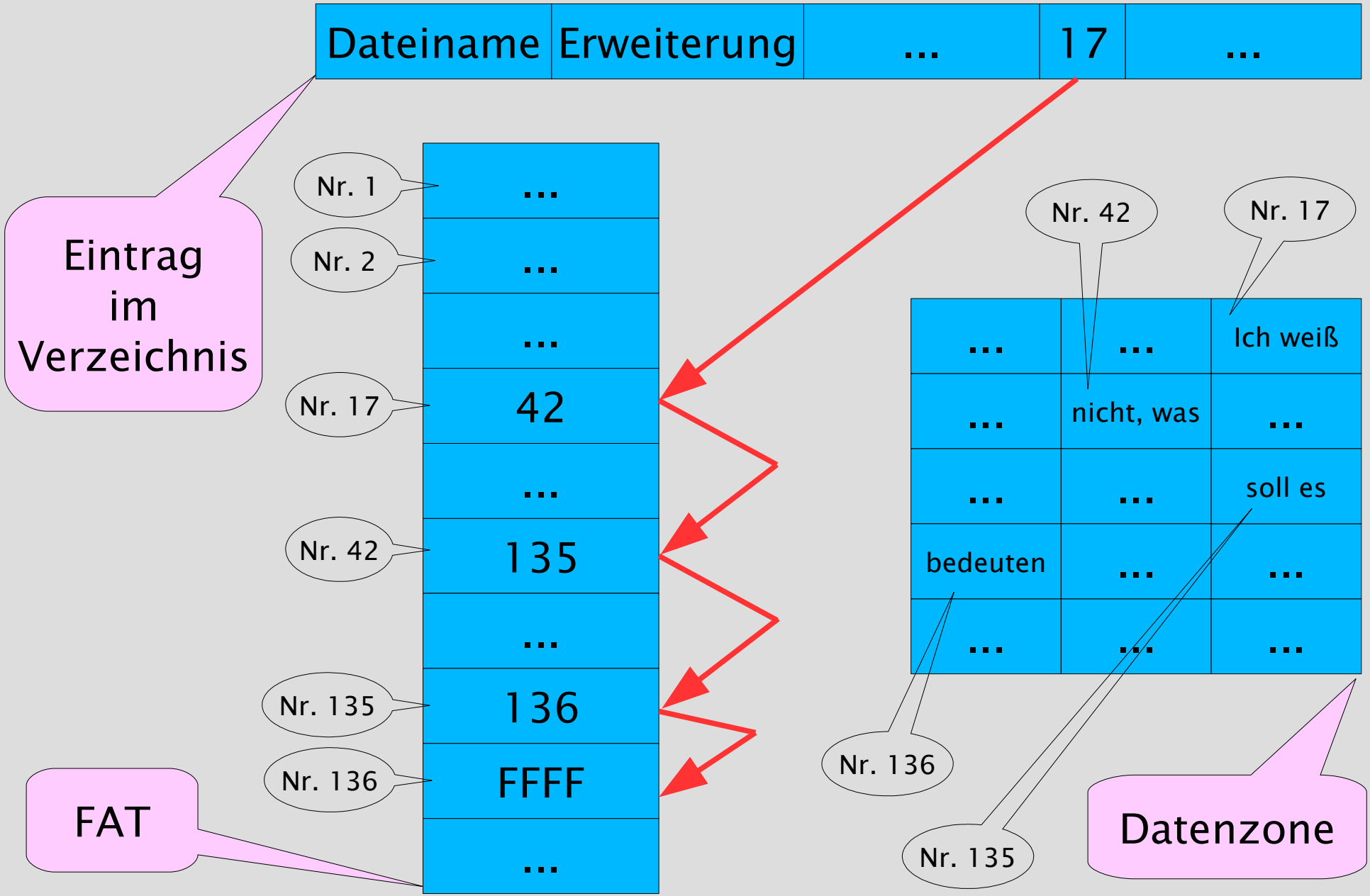
Verzeichnis ist auch eine Tabelle, jede Zeile ist 32 Byte lang:

- 8 Bytes — Dateiname.
- 3 Bytes — Erweiterung.
- 1 Byte — Attribute.
- 10 Bytes — Reserve.
- 2 Bytes — Uhrzeit.
- 2 Bytes — Datum der letzten Änderung oder Erstellung.
- 2 Byte — Nummer des ersten Clusters mit den Daten der Datei.
- 4 Bytes — Größe der Datei.

Hauptverzeichnis hat feste Position auf der Festplatte und begrenzte Anzahl von Einträgen (511). Die Unterverzeichnisse werden wie Dateien behandelt.

Attribut-Byte:

- Bit 0 = 1 readonly.
- Bit 1 = 1 hidden.
- Bit 2 = 1 system.
- Bit 3 = 1 dieser Eintrag ist der Name des Datenträgers.
- Bit 4 = 1 dieser Eintrag ist ein Unterverzeichnis.
- Bit 5 = 1 diese Datei wurde seit der letzten Sicherung geändert (archiv).


Dateiname
Erweiterung
...
17
...
Eintrag im Verzeichnis
Nr. 1
Nr. 2
Nr. 17
Nr. 42
Nr. 135
Nr. 136
...
42
135
136
FFFF
FAT
Nr. 42
Nr. 17
Ich weiß
nicht, was
soll es
bedeuten
Nr. 136
Nr. 135
Datenzone

Das Dateisystem NTFS (New Technology File System) ist das proprietäre Produkt der MS-Windows-NT-Reihe.

Es gibt folgende Versionen:

- NTFS 1.x
- NTFS 2.x
- NTFS 3.x
- Transactional NTFS

Wichtige Eigenschaften von NTFS:

- Datei wird durch eine File Reference Number (FRN) eindeutig identifiziert, nicht durch den Namen.
- Ausgereifte Zugriffsberechtigungen auf Dateien/Verzeichnisse (Access Control List, ACL) inkl. Protokollierung der Zugriffe.
- Ausgereiftes Besitzer-Konzept für Dateien/Verzeichnissen.
- Kontingentverwaltung.
- Dateiverschlüsselung mit EFS (Encrypted File System).
- Komprimierung der Dateien.
- Datenträger können mit einem Verzeichnis verknüpft werden.
- Journaling gewährleistet die Konsistenz des Dateisystems.

Das Dateisystem NTFS besteht aus:

- Master File Table, MFT — zentrale Verwaltungsstelle für Dateien und Verzeichnisse. Hier gibt es einen Eintrag für jede Datei und jedes Verzeichnis.
- Cluster Bitmap File — Information über belegte Datenblöcke.
- Bad Cluster File — Information über fehlerhafte Datenblöcke.
- Volume File — Information über das logische Laufwerk.
- Log File — Information über die Vorgänge, die für Wiederherstellung der logischen Struktur des Dateisystems im Falle eines Absturzes notwendig ist.
- Datenzone.

Aus der Sicht des Dateisystems NTFS besteht jede Datei aus Attributen:

- Name (bis 255 Symbole lang).
- Kurzer Name für Kompatibilität zu FAT16.
- Inhalt.
- ACL, Zugriffsberechtigungen.
- Besitzer.
- Zeitstempel.

Die FRN basiert auf der Nummer des Datensatzes in MFT, der den Dateneintrag enthält. Das erlaubt es, mehrere Namen für einen Inhalt zu haben (harte Links). Die Datei kann einen Verweis auf eine andere Datei enthalten (symbolische Links).

Ein Verzeichnis enthält die Liste der zugeordneten FRNs.

Das Dateisystem ext ist ein Standard in den UNIX/Linux-Systemen. Es ist mit vielen anderen Dateisystemen wie ReiserFS, xfs, minix, Next3 verwandt.

Es gibt folgende Versionen:

- ext2
- ext3
- ext4

Wichtige Eigenschaften von ext:

- Datei wird durch eine Inode-Nummer eindeutig identifiziert, nicht durch den Namen.
- POSIX-kompatible Zugriffsberechtigungen auf Dateien/Verzeichnisse und Besitzer-Konzept, also weniger Flexibilität und Funktionsumfang als NTFS, aber dafür kompakter. ACL von NTFS kann installiert werden.
- Kontingentverwaltung kann installiert werden.
- Dateiverschlüsselung kann installiert werden.
- Komprimierung der Dateien kann installiert werden.
- Datenträger werden mit einem Verzeichnis verknüpft.
- Journaling gewährleistet die Konsistenz des Dateisystems, kann bis zu Full-Journaling erweitert werden.

Aufbau des Dateisystems ext:

- Superblock. Hier werden folgende Informationen gespeichert: Typ des Betriebssystems, Größe des Dateisystems (Größe der Inode-Liste + Anzahl der Datenblöcke).
- I-Bitmap (>=1024 Bytes). Jedem Bit in diesem Bereich entspricht ein Eintrag in der Inode-Liste.
  Bit=1 — Eintrag wird benutzt, Bit=0 — Eintrag ist frei.
- D-Bitmap (>=1024 Bytes). Jedem Bit entspricht ein Datenblock.
  Bit=1 — Datenblock wird benutzt, Bit=0 — Datenblock ist frei.
- Inode-Liste (inode, information node). Jeder Eintrag in dieser Liste beschreibt eine Datei. Die Größe eines Eintrages ist 128 Bytes (oder flexibel mehr in ext4).
- Datenzone.

Prinzipieller Aufbau eines Eintrages in der Inode-Liste:

- UID, Besitzer der Datei.
- GID, Gruppe, zu der die Datei gehört.
- Anzahl der Links (Verweiszähler).
- Größe der Datei.
- Typ der Datei (*-, d, l, b, c*).
- Rechte (rwxrwxrwx) laut POSIX.
- Datum, Uhrzeit laut POSIX.
- Direkte Verweisfelder 1-12, sie enthalten die Nummer des Datenblocks, wo sich die Daten der Datei befinden.

- Indirektes Verweisfeld des 1. Grades, es enthält die Nummer des Datenblocks, wo sich die Verweise auf Datenblöcke der Datei befinden.
- Indirektes Verweisfeld des 2. Grades, es enthält die Nummer des Datenblocks, wo sich die Verweise auf die Datenblöcke befinden, die auf die Daten der Datei zeigen.
- Indirektes Verweisfeld des 3. Grades, es hat die Nummer des Datenblocks, wo sich die Verweise auf die Datenblöcke befinden, wo sich wiederum die Verweise auf die Datenblöcke befinden, die auf die Daten der Datei zeigen.

Dank diesem Aufbauprinzip kann eine Datei unter den Dateisystemen ext3/ext4 die maximale Größe von 256 TiB bis zu 256 PiB haben.

Das Dateisystem selbst erreicht die Größe von 256 PiB bis zu 1 YiB.

Das Verzeichnis ist eine Datei vom Typ *d* (eigentlich, einfach eine Liste), die Einträge über entsprechende zugehörige Dateien hat. Nur bestimmte Programme (Befehle), die diese Struktur kennen, können auf den Inhalt des Verzeichnisses zugreifen.

Jeder Eintrag beinhaltet:

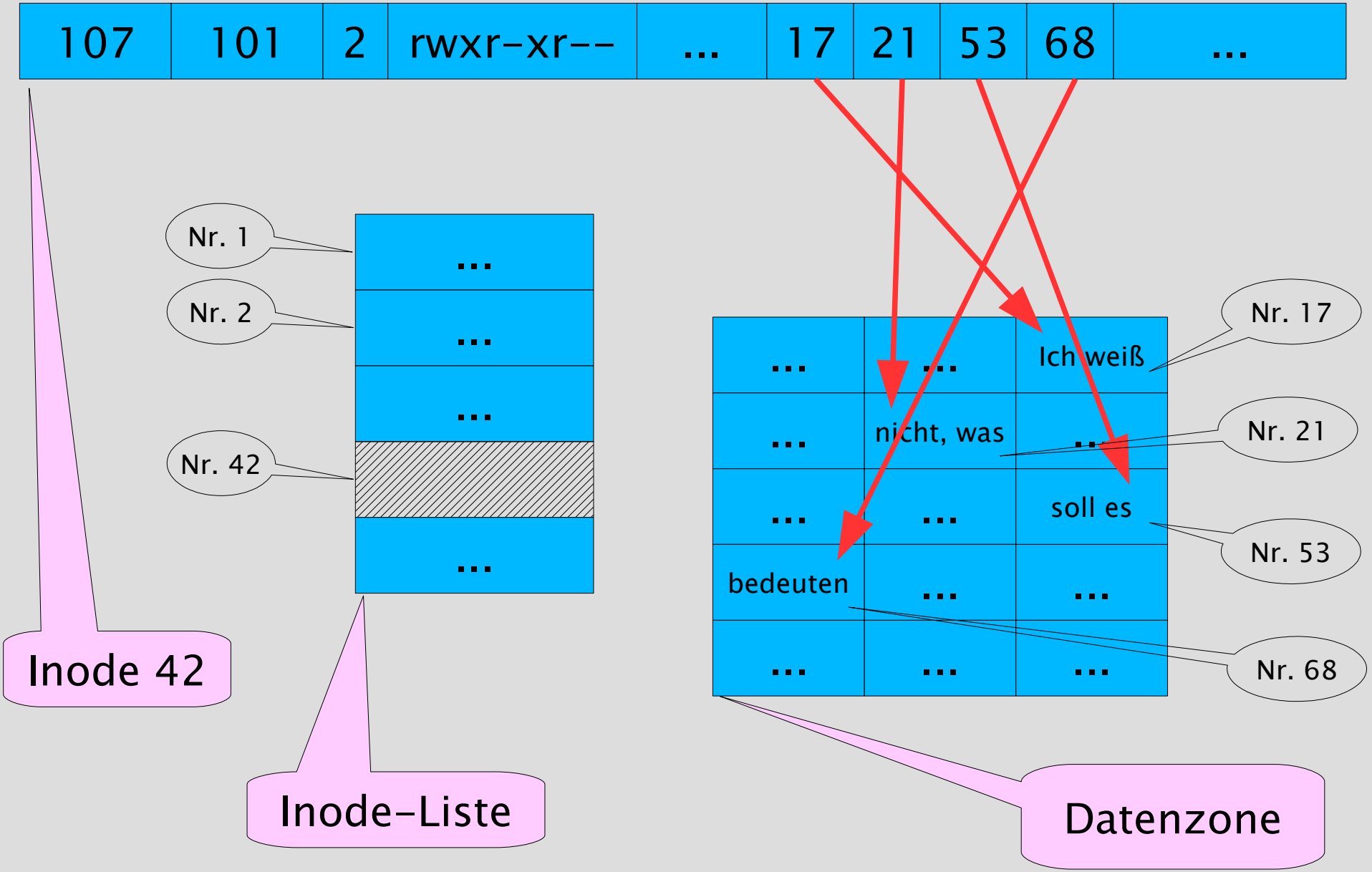
- Dateiname.
- Inode-Nummer (die Reihenfolgenummer des entsprechenden Eintrages in der Inode-Liste).

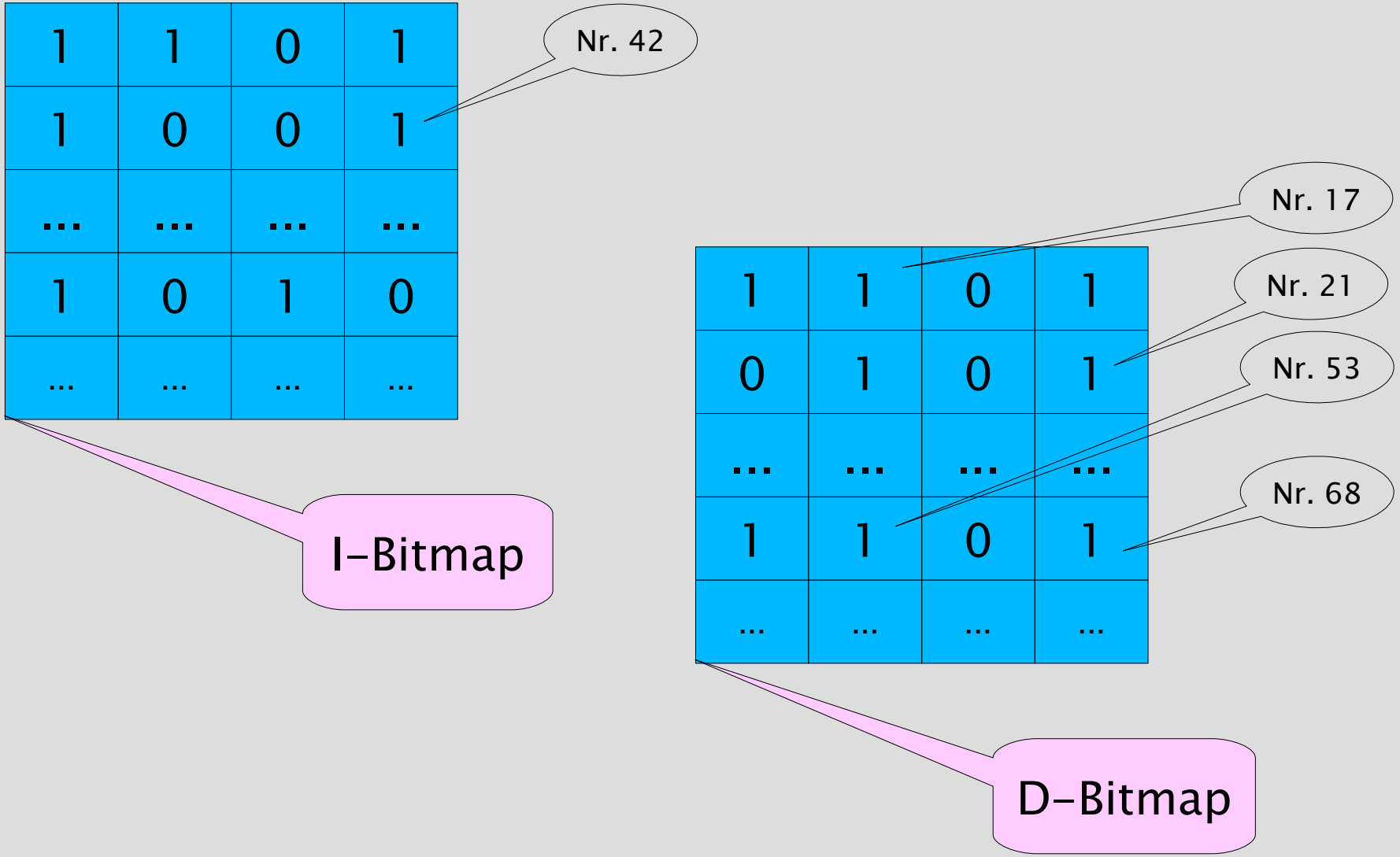
Wenn eine neue Datei zu erstellen ist, ermittelt das Betriebssystem anhand der I-Bitmap und D-Bitmap freie Inode-Nummer und Datenblöcke. Wird Datei ohne Fehler angelegt, so werden die Bits in I-Bitmap und D-Bitmap auf 1 gesetzt und wird entsprechender Eintrag im Verzeichnis gemacht.

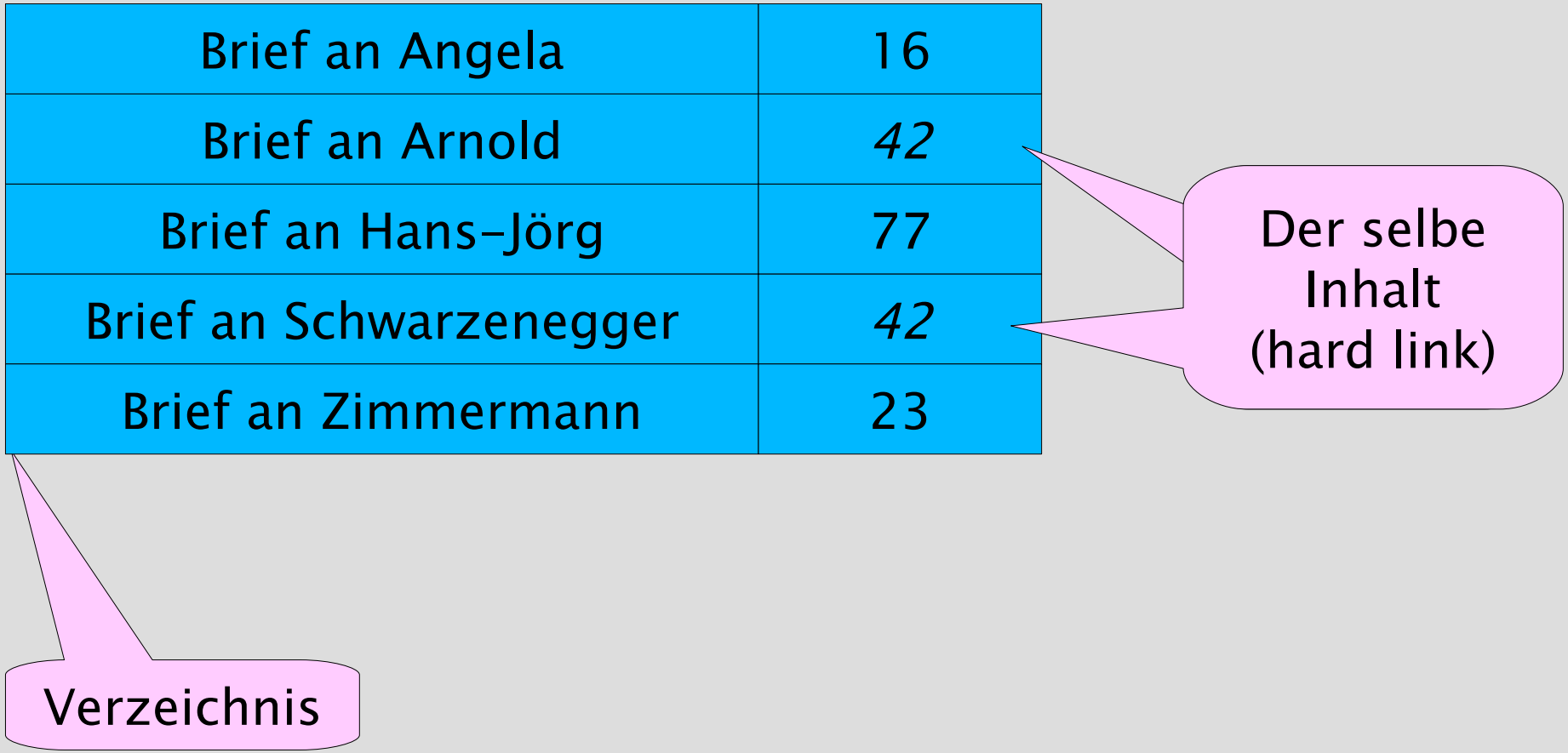
Wenn eine Datei mit einem bestimmten Namen zu finden ist, wird zuerst der Eintrag aus dem Verzeichnis gelesen. Dadurch wird die Inode-Nummer der Datei ermittelt. Dann wird die Zeile mit dieser Nummer aus der Inode-Liste gelesen, und ab jetzt kann der Inhalt der Datei gelesen werden.

Ist eine Datei, für die die Anzahl der Links = 1 ist, zu löschen, dann werden die entsprechenden Bits in I-Bitmap und D-Bitmap auf 0 gesetzt und somit wird der Eintrag in Inode-Liste frei (so dass er für eine neue Datei vergeben werden kann). Werden die Daten in Datenblöcken tatsächlich gelöscht oder nicht ist von Sicherheitslinien abhängig.

Ist beim Löschen die Anzahl der Links > 1, dann wird die Anzahl der Links auf 1 vermindert, so dass die Inhalte des I-Bitmap, des D-Bitmap und der Datei unverändert bleiben.


107
101
2
rwxr-xr--
...
17
21
53
68
...
Nr. 1
Nr. 2
Nr. 42
Nr. 17
Nr. 21
Nr. 53
Nr. 68
Ich weiß
nicht, was
soll es
bedeuten
Inode 42
Inode-Liste
Datenzone


1 1 0 1
1 0 0 1
... ...
1 0 1 0
... ...
Nr. 42
I-Bitmap
1 1 0 1
0 1 0 1
... ...
1 1 0 1
...
Nr. 17
Nr. 21
Nr. 53
Nr. 68
D-Bitmap


Brief an Angela
16
Brief an Arnold
42
Brief an Hans-Jörg
77
Brief an Schwarzenegger
42
Brief an Zimmermann
23
Der selbe Inhalt (hard link)
Verzeichnis