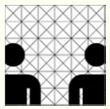
B Betriebssysteme



B4 Dateisysteme (a)
B4.1 Einführung

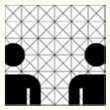
Def. Datei:

> Längerfristig zu speichernde Datenmenge (Datenobjekt) bestehend aus logisch zusammenhängender Menge von **(Daten-)Sätzen** bzw. **logischen Datenblöcken**, die ihrerseits auf **(physikalische) Blöcke** des Hintergrund-speichers abgebildet werden.

... oder *allgemeiner*: Einheit der Speicherung beliebiger Daten auf einem persistenten (dauerhaften) Speicher [NeS98].

Übliche Sichten von Dateien:

- *Großrechner* (kommerzielle DV):
  Datei aus Menge logischer Datenblöcke bestehend
- *Personal Computer*:
  Datei aus Bytesequenz bestehend (vgl. UNIX, Windows)



Typische Dateiinhalte

Beispiele von Dateiinhalten:

- Quellcode eines Programms in höherer Programmiersprache
- ausführbarer Programm-Code
- Ein-/Ausgabedaten für Programme
- Unternehmensdaten (z.B. Angestellten-, Kunden-, Lieferanden-Daten, etc)
- Audiodaten (z.B. MP3-codiert) oder Videosequenz (z.B. MPEG-codiert)
- Digitalbild, u.v.a.m.

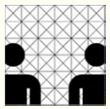
Beispiele für Hintergrundspeicher (vgl. Peripher- und Archivspeicher in Speicherhierarchie):

- Magnetplattenspeicher
- Disketten
- Magnetbänder

} *auf Magnetisierungs-zuständen basierend*

- (wieder-) beschreibbare CDs, DVDs,
- magneto-optische Plattenspeicher, etc

→ wesentlich für Hintergrundspeicher: **nicht-flüchtiger Speicher**
(wir sprechen auch von **dauerhafter** bzw. **persistenter Speicherung**)



Grundanforderungen an ein Dateisystem

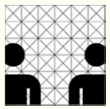
Unterstützung der Dateiverwaltung durch das Betriebssystem, insbes. durch das Dateisystem mit folgenden Aufgaben:

- Erstellen / Löschen von Dateien
- Unterstützung des Zugriffs (Lesen/ Schreiben) auf Dateien
- Verwaltung des Hintergrundspeichers
- Datenschutz (Verhinderung unberechtigter Zugriffe auf Daten)
- Datensicherung (Maßnahmen gegen phys. Verlust/Zerstörung von Dateien)

→ ergo: Dateisystem kann als *Ordnungs- und Zugriffssystem für Dateien* gesehen werden.

Zugriffe auf Dateien:

- i.d.R. über ihren (Datei-) Namen
- unter Berücksichtigung wohl definierter Zugriffsrechte (z.B. für Lesen, Schreiben, Ausführen)
- unter Verwendung unterschiedlicher Lese-/Schreibgranulate (z.B. blockorientierter Zugriff auf Teile der Datei)



Charakteristika der Dateinutzung

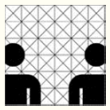
Empirische Untersuchungen zeigen :

- Dateien sind zumeist klein (wenige KByte)
- Dateien werden häufiger gelesen, seltener geschrieben und noch seltener gelöscht
- sequentieller Zugriffe ist dominant (z.B. Zugriff auf Dateiinhalte in serieller Weise)
- Dateien werden selten von mehreren Programmen/Prozessen oder Personen gleichzeitig benutzt.

nota bene: Dateisysteme im allg. optimiert für o.g. Nutzungsverhalten.

ABER: Nutzungsverhalten ändert sich über der Zeit, beispielsweise

- benötigen Audiodaten (bei CD-Qualität) pro Minute Ton ca. 10 MByte
- benötigt eine unkomprimierte Videoaufzeichnung (Format: 1024 x 768, 3 Byte pro Pixel, 50 Bilder pro sec) pro Minute ca. 6.5 GByte



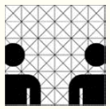
B4.2 Operationen auf Dateien

Wir haben gesehen:

Dateien sind “Behälter für (nahezu beliebige) Informationen” mit NAMEN und ATTRIBUTEN

→ Beispiele für Dateiattribute :

- Typ (Datei oder Verzeichnis, s.u.)
- Speicherort (z.B. welche Platte und an welcher Position)
- Größe
- Erzeugungs- sowie letzter Zugriffs- und Änderungszeitpunkt
- Besitzrechte
- Zugriffsrechte (z.B. für lesenden, schreibenden oder ausführenden Zugriff)

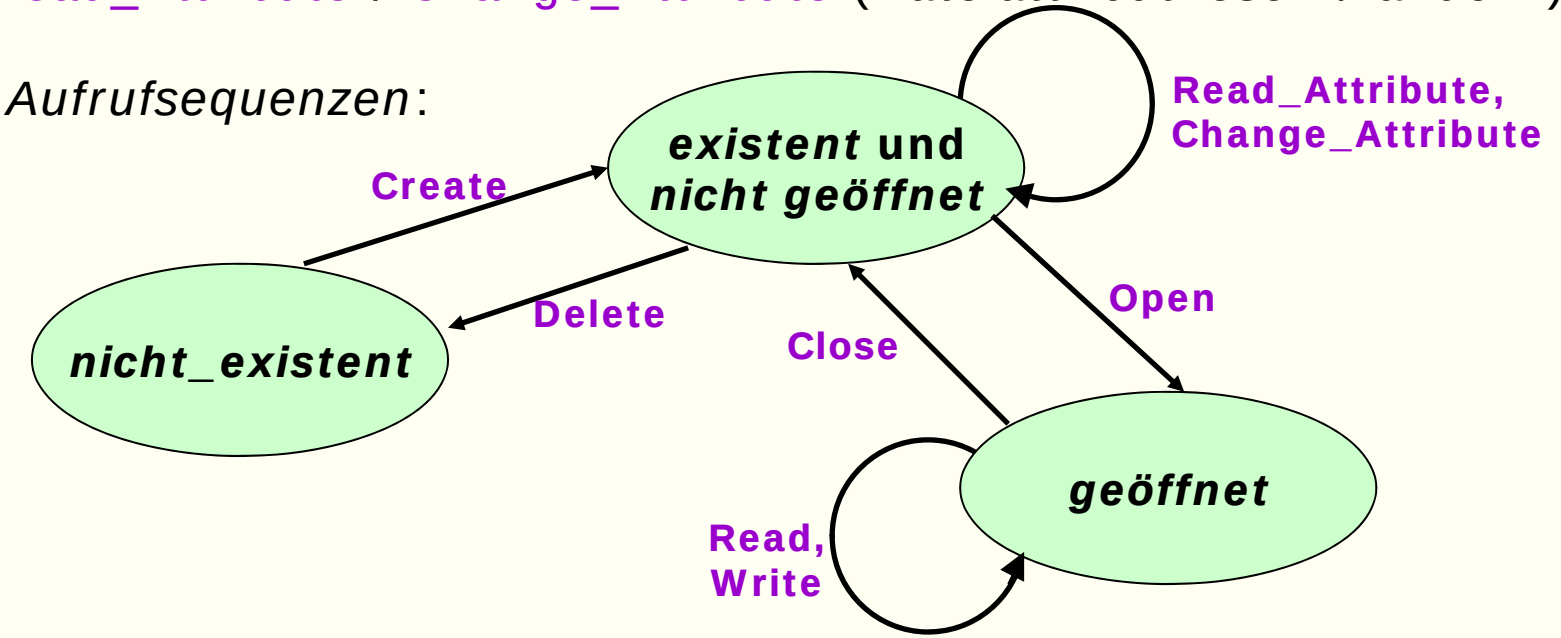


Liste und mögliche Aufrufsequenzen von Operationen auf Dateien

Die wesentlichen Operationen auf Dateien sind :

- Create / Delete (*Erzeugen / Löschen*)
- Open / Close (*Öffnen / Schließen*)
- Read / Write (*Lesen / Schreiben von Dateiinhalten*)
- Read_Attribute / Change_Attribute (*Dateiattribut lesen / ändern*).

Mögliche Aufrufsequenzen:



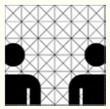
Öffnen einer Datei

Gründe für die Notwendigkeit des expliziten Öffnens:

- Zugriffsberechtigung des öffnenden Prozesses ist zu überprüfen
  (Art des gewünschten Zugriffs bereits beim Öffnen spezifiziert !)
- MUTEX-Problem zu lösen bei Auftreten der "*Reader-Writer*"-Problematik
- Existenz der zu öffnenden Datei ist sicherzustellen (ansonsten evtl. Erzeugung der Datei, d.h. CREATE und OPEN kann in existierenden Dateisystemen evtl. zu einer Operation zusammengefasst werden)
- Dateien auf externem Speicher durch Dateisystem zu lokalisieren und interne Datenpuffer für anschließenden Dateizugriff zu initialisieren

***nota bene*:**

Rückmeldungen an den aufrufenden Prozess geben Information über den Erfolg der Operation und bei Misserfolg (Fehlermeldung) evtl. auch über die Gründe.



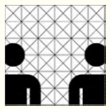
Schließen einer Datei

Explizites Schließen der Datei nach Beendigung der entsprechenden Dateizugriffe (seitens des zugreifenden Prozesses P) ist sehr wünschenswert, da sodann :

- evtl. andere Prozesse auf die Datei zugreifen können (siehe MUTEX-Problem)
- das Dateisystem reservierte Ressourcen (z.B. Datenpuffer) wieder freigeben kann.

GLEICHWOHL:

Schließt Prozess P die Datei nicht, so erledigt dies das Betriebssystem sobald P terminiert.



Lesen / Schreiben einer Datei

Nota bene: Lesen / Schreiben nur auf bereits geöffneten Dateien

Bei *Leseoperation*: Puffer zu spezifizieren, in den eingelesene Daten zu kopieren

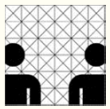
Bei *Schreiboperation*: Adresse zu spezifizieren, unter der die zu schreibenden Daten lokalisiert werden können.

→ bei Lese- und Schreiboperationen überdies:

Menge der zu lesenden/schreibenden Zeichen (Byte) zu spezifizieren; Lese-/ Schreiboperationen implizit auf aktuelle Dateiposition bezogen, wobei Dateiposition beim Öffnen der Datei auf Dateianfang initialisiert wird.

Bem.:

Dateisystem ist – in Verbindung mit dem E/A-System des Rechners – für die Realisierung der konkreten Zugriffe und Datentransfers vom / zum physikalischen (Peripher-) Speicher zuständig.



Zugriffsrechte auf Dateien (Beispiel UNIX)

Benutzerklassen:

- **u**ser
- **g**roup
- **o**ther

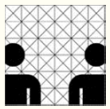
Für jede dieser Benutzerklassen (**u**ser, **g**roup und **o**ther) gibt es jeweils drei verschiedene Zugriffsrechte:

- Lesen (**r** - **r**ead)
- Schreiben (**w** - **w**rite)
- Ausführen (**x** - e**x**ecute)

Insgesamt können somit neun Zugriffsrechte *(protection bits)* für jede Datei gesetzt werden.

Bsp: rwx rwx rwx
rw- r-- r--
rwx r-x r-x

Anm.: Zu Details vgl. ”Security-Teil” der GSS-Vorlesung



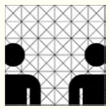
Arten von Dateien

Beispiel: File Extensions unter Windows

- File_name.**doc** : Word-Datei
- File_name.**exe** : ausführbare Datei
- File_name.**jpg** : JPEG-codierte Festbild-Datei
- File_name.**mpg** : MPEG-codierte Video-Datei
- File_name.**txt** : Text-Datei
- File_name.**ppt** : PowerPoint-Datei
- File_name.**ps** : PostScript-Datei
- File_name.**pdf** : PDF-Datei
- File_name.**mht** bzw. *.**mhtml** : Webseite in einer Datei
- File_name.**xml** : Word XML-Dokument

  u.v.a.m.

nota bene: ähnliche “Extensions” in anderen Betriebssystemen; Angaben schränken die Art der Dateibenutzung bzw. ihrer Weiterverarbeitung ein!



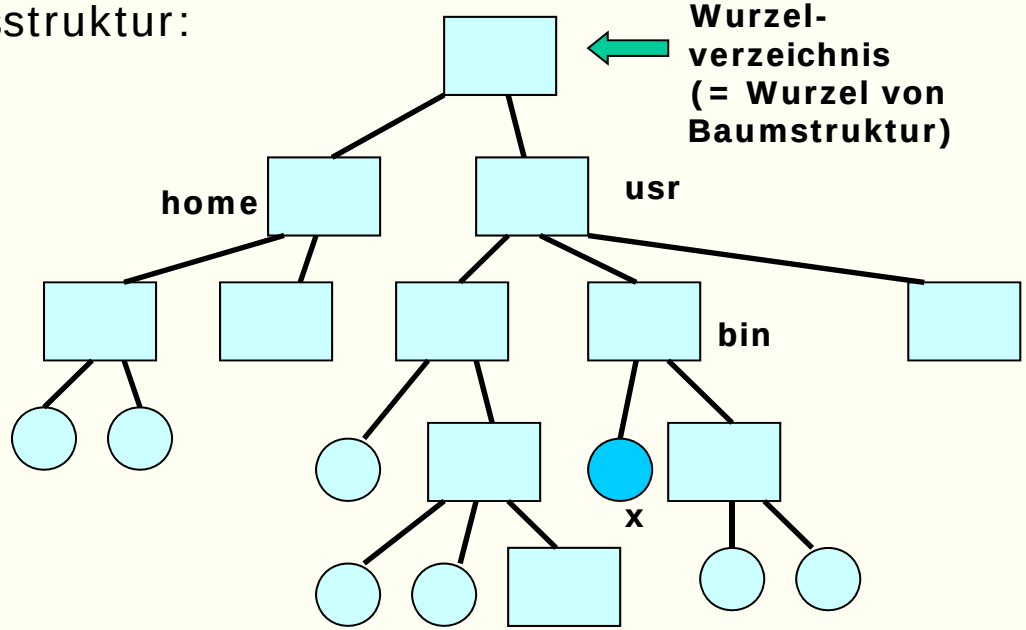
B4.3 Verzeichnisse

Def. Verzeichnis (*directory*) oder auch Katalog (*catalog*):

> Einheit in einem Dateisystem, die aus einer Gruppierung von Dateien und/oder anderen Verzeichnissen (sog. Unterverzeichnissen – *subdirectories*) besteht.

Verzeichnisse erlauben eine hierarchische Strukturierung des externen Speichers.

Beispiel für Verzeichnisstruktur:

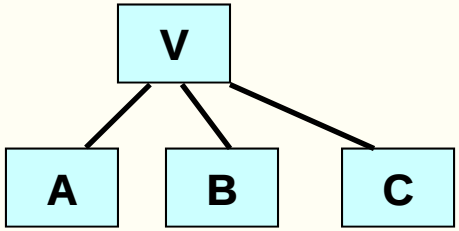


Notation:
: Verzeichnis
: Datei

Absoluter Name der mit x benannten Datei: **/usr/bin/x**

Spezielle Verzeichnisse

- Vaterverzeichnis (*parent directory*) :



Bsp.: V ist Vaterverzeichnis (= direkt übergeordnetes Verzeichnis) für die Unterverzeichnisse A, B und C

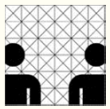
- Wurzelverzeichnis (*root directory*) :

Das Verzeichnis an der Wurzel des Baumes, das als einziges kein übergeordnetes Vaterverzeichnis besitzt.

- Arbeitsverzeichnis (*working directory*) :

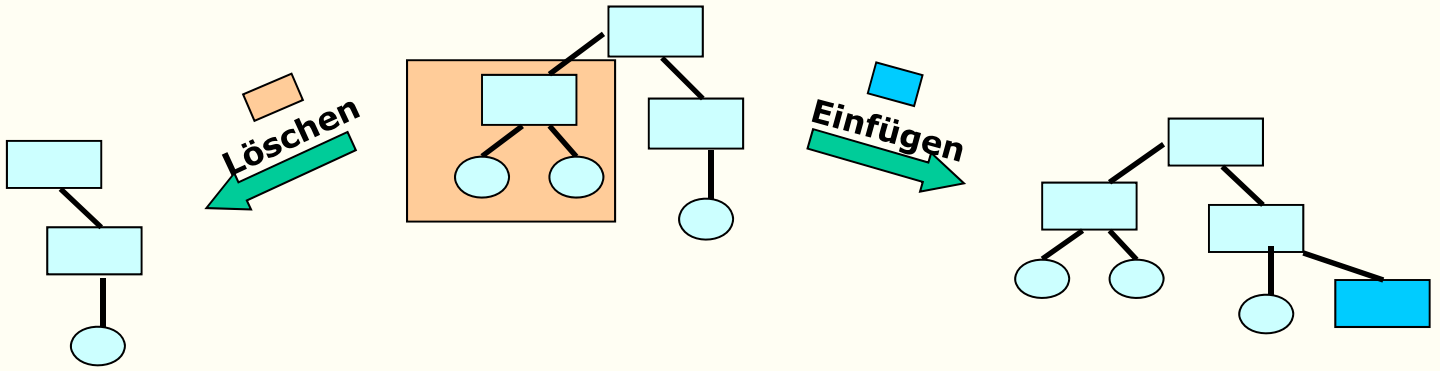
Um Dateien einfacher ansprechen zu können, führt das Dateisystem für jeden Prozess während seiner Ausführung ein sog. Arbeitsverzeichnis.

Bsp.: Ist Arbeitsverzeichnis
/home/Lehre/GSS
so genügt die Angabe von **KapB4/Version25April07.ppt**
um die Datei mit dem absoluten Dateinamen
/home/Lehre/GSS/KapB4/Version25April07.ppt
eindeutig zu benennen.

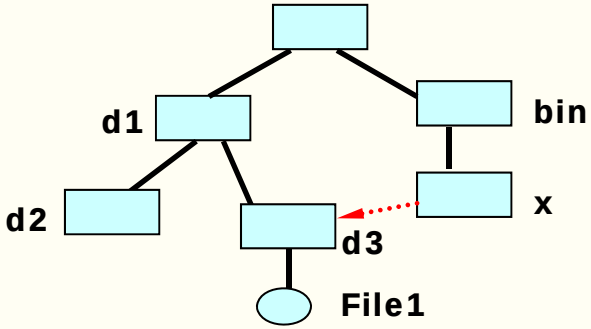


Operationen auf Verzeichnissen

- **Einfügen** / **Löschen** von Dateien bzw. Unterverzeichnissen in/aus existierendes/m Verzeichnis



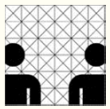
- **Umbenennen** von Dateien bzw. Verzeichnissen
- **Navigieren** durch ein Verzeichnis
- **Setzen von "Links"** (= Verweis auf andere Dateien oder Verzeichnis)



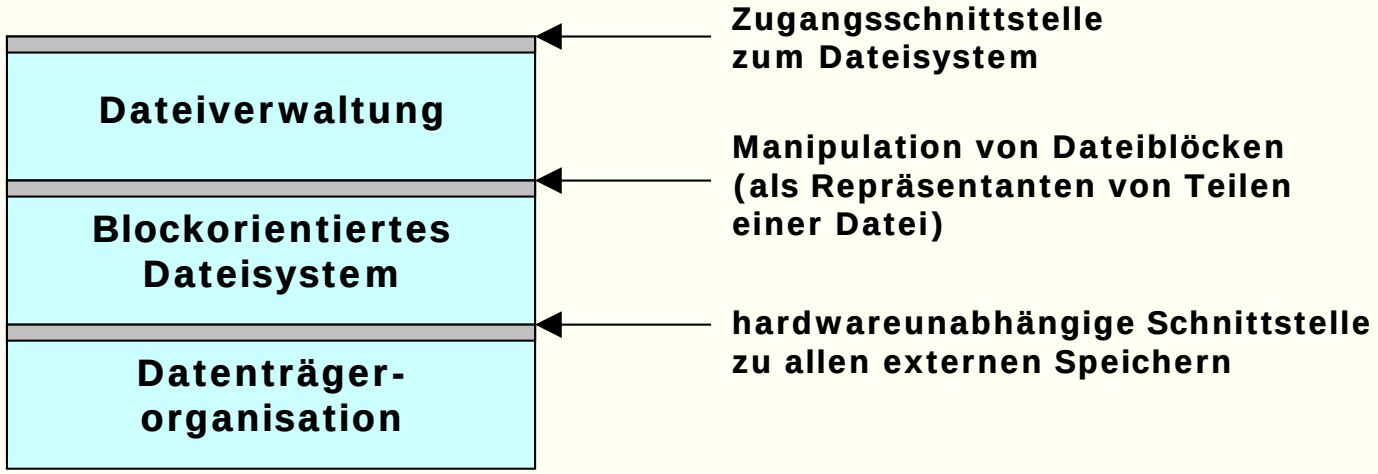
Notation:

Link (auf /d1/d3 verweisend)

File1 erreichbar über **/bin/x/File1** sowie **/d1/d3/File1**

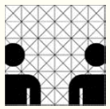


B4.4 Schichtenmodell für Dateisysteme



Grundfunktionen der Schichten:

- **Dateiverwaltung :**
  - Realisierung von Verzeichnissen und der daraus resultierenden hierarchische Strukturierung der externen Speicher
  - Bereitstellung von Dateizugriffsfunktionen (unabhängig vom Blockgranulat der tieferen Schichten und der verwendeten Datenträger)
- **Blockorientiertes Dateisystem :**
  - Aufteilung der Blöcke eines Datenträgers auf einzelne Dateien
- **Datenträgerorganisation :**
  - Organisation der physikalischen Datenträger (z.B. Festplatten, Disketten, ...)



Schicht **“Dateiverwaltung”**: Verfeinerung

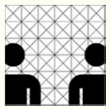
Aufgaben :

– Bereitstellung von Verzeichnissen zur hierarchischen Organisation des Datenträgers

→ Dateien hier identifiziert über Namen

– Abbildung der Dateizugriffe auf blockbasierte Lese- und Schreib-operationen

→ Festlegung der Zugriffsgranularität seitens der Anwendung



Schicht **“Blockorientiertes Dateisystem”**: Verfeinerung

Aufgaben :

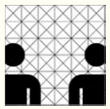
– Aufteilung des vorhandenen Speicherplatzes eines logisch durchnummerierten Datenträgers auf mehrere Dateien

→ nur interne Namen für Dateien in dieser Schicht, überdies keine hierarchische Verzeichnisstrukturen (d.h. Dateien in flacher Struktur über interne Dateinamen anzusprechen); Dateien aus Menge von Blöcken bestehend – relativ zum Dateianfang nummeriert.

– Realisierung folgender Zugriffsfunktionen auf Dateien :

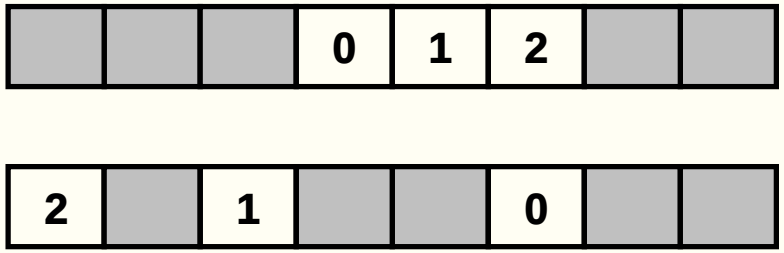
- *Erzeugen/Löschen*,
- *Vergrößern/Verkleinern*,
- *Öffnen/Schließen* sowie
- *Lesen/Schreiben*

von Blöcken

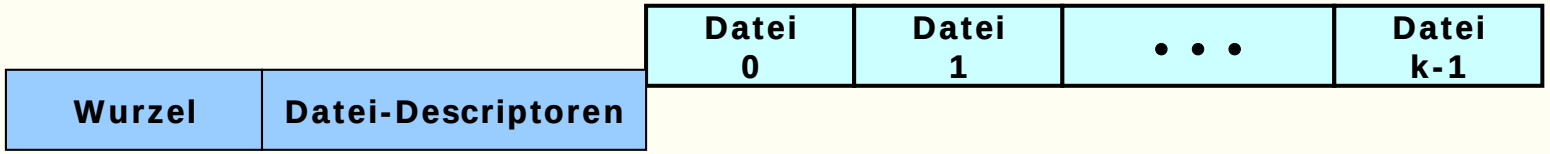


Schicht **“Blockorientiertes Dateisystem”**: Verfeinerung(Forts.)

- **Varianten zur Platzierung aufeinander folgender Dateiblöcke:**
  - ***zusammenhängend*** :
    → i.a. schnellere Zugriffe
  - ***verteilt*** :
    → i.a. bessere Speicher-ausnutzung

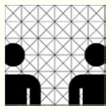


- **Aufbau eines Datenträgers (Sicht des blockorientierten Dateisystems):**



wobei:

- ***Dateidescriptoren*** genutzt werden zur Speicherung sämtl. relevanter Dateiattribute sowie zu Verweisen auf Datenstrukturen, zur Abb. “log. Block-Nr. (innerhalb einer Datei) → Block-Nr. des Datenträgers”
- ***Wurzel*** mit Verwaltungs-Info (z.B. reservierte Anzahl Descriptoren und Position Descriptortabelle)



Schicht **“Datenträgerorganisation”**: Verfeinerung

Aufgaben :

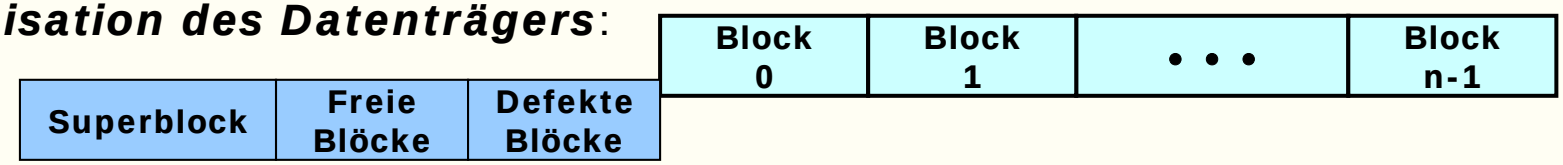
– Logische Durchnummerierung aller Blöcke eines Datenträgers

→ Vorteil der einheitlichen Adressierbarkeit der Blöcke eines externen Speichers unabhängig von techn. Realisierung des Datenträgers

– Bereitstellung von Lese-/Schreiboperationen basierend auf Block-nummerierung

– Formatierung eines Datenträgers (u.a. hier auch defekte Blöcke vor einer Verwendung geschützt).

→ ***Organisation des Datenträgers***:



wobei :

- ***Superblock*** : zur Verwaltung sämtl. essentieller Infos bzgl. Datenträger-aufbau (u.a. Größe des Datenträgers, Blockgröße, Positionen der Bitvektoren $(f_i)_{i=0,\ldots,n-1}$ für die freien und $(d_j)_{j=0,\ldots,n-1}$ für die defekten Blöcke)
  → Superblock mehrfach abgespeichert (Ausfallsicherheitsgründe !)
- ***Freie Blöcke*** : $f_0, f_1, f_2, \ldots f_{n-1}$ wobei $f_i = 1$ falls Block B_i frei und 0 sonst.
- ***Defekte Blöcke*** : $d_0, d_1, d_2, \ldots d_{n-1}$ wobei $d_j = 1$ falls Block B_j defekt und 0 sonst.