# Kapitel 4: Festplatten und Dateisysteme

## Basiswissen Mathematik für Informatiker*innen: Zweierpotenzen AUSWENDIG LERNEN

$1Mi = 1Ki \times 1Ki = 2^{10} \times 2^{10} = 2^{20}$
$1Gi = 1Mi \times 1Ki = 2^{20} \times 2^{10} = 2^{30}$
$1Ti = 1Gi \times 1Ki = 2^{30} \times 2^{10} = 2^{40}$

- Ein Byte besteht aus 8 Bit! Es kann $2^8$ verschiedene Werte annehmen.
- Es gibt $x^n$ viele verschiedene Zahlen mit $n$ Stellen und $x$ vielen verschiedenen Ziffern. Beispiele:
  - 3 Stellen und 10 Ziffern: $10^3$ viele Zahlen
  - 32 Stellen und 2 Ziffern: $2^{32}$ viele Zahlen
- Ki ist das Symbol für kibi (motiviert durch "kilobinary"), analog sind Mi, Gi und Ti die Symbole für mebi, gibi und tibi.

| | | |
|---|---|---|
| $2^0 = 1$ | $2^6 = 64$ | $2^{12} = 4096$ |
| $2^1 = 2$ | $2^7 = 128$ | $2^{13} = 8192$ |
| $2^2 = 4$ | $2^8 = 256$ | $2^{14} = 16384$ |
| $2^3 = 8$ | $2^9 = 512$ | $2^{15} = 32768$ |
| $2^4 = 16$ | $2^{10} = 1024$ (1K) | $2^{16} = 65536$ |
| $2^5 = 32$ | $2^{11} = 2048$ | $2^{32} = 4294967296$ |

## Aufbau von Festplatten

### Mechanik von Festplattenlaufwerken

- **Spur** (engl. track) ist eine kreisförmige Bahn auf der Oberfläche.
- **Zylinder** (engl. cylinder) bezeichnet übereinander liegende Spuren (auf verschiedenen Oberflächen).
- **Sektor** (engl. sector) ist ein Ausschnitt fester Länge aus einer Spur.
- **Block** ist der Nutzinhalt eines Sektors.
- **Cluster** ist eine Gruppe von Blöcken. Größere Verwaltungseinheit.

### Adressierung von Blöcken

**Angabe der drei Koordinaten:**

- **(1) Zylinder, (2) Kopf (engl. head) und (3) Sektor (CHS-Adressierung).**
  - Implementierung dieses Verfahrens im BIOS von PCs.
  - Typische IDE Festplatten

**Angabe einer logischen Blocknummer (LBA = Logical Block Address)**

- Implementierung im Controller der Festplatte
- Controller kann auch die „alte“ Adressierung umrechnen.
- Entspricht einer Nummerierung der Blöcke beginnend mit Null
- Wird bei SSDs benutzt.

### Parameter von Festplatten

- **Blockgröße**: Typischerweise eine 2er Potenz, 1 KB, 2 KB, 4 KB, . . .
- **Umdrehungsgeschwindigkeit**: 5600 UpM, 7200 UpM bis ca. 15000 UpM bei Serverplatten
- **Positionierzeit**: 5 bis 10 ms
- **Maximale Datenübertragungsrate**: bis ca. 200 MB/s bei rotierenden Platten, bis ca. 3200 MB/s bei SSDs (Angaben ändern sich laufend)
- **Wartezeit beim Festplattenzugriff:**

  Positionierzeit + Latenzzeit + Übertragungszeit
- **Beschleunigung** der Zugriffe durch einen **Cache** auf der Platte ist möglich.
- **Solid State Disks (SSD)** erlauben durch den Einsatz von Halbleiter-Chips noch schnellere persistente Speicher.

### Partitionieren von Festplatten

**Idee**: Teile eine **physische (reale) Festplatte** logisch in Teile (Partitionen), die sich wie eine (virtuelle) Festplatte verhalten. *(physiche Dateneinheiten!)*

- Ganze Zylinder werden zu Partitionen zusammengefasst.

**Im Master Boot Record (MBR)** einer Festplatte steht u. a. die Partitionstabelle.

- Dort ist Platz für **maximal 4 primäre Partitionen**.
- Die letzte primäre Partition kann als **extended Partition** konfiguriert werden und dann wiederum sogenannte **logische Partitionen enthalten**. *(mehr als 4)*

**Zweck**: Installation mehrerer Betriebssysteme auf einer Platte, Trennung von wichtigen Datenbereichen, vorhalten einer „Recovery“-Partition zur Wiederherstellung des Betriebssystems, usw.
*(In Unix Systemen nutzen wir Partitionierung um verschiedene Verzeichnisse zu ordnen)*

Beispiel einer Partitionierung

- In Linux-Systemen werden Festplatten über Gerätedateien im Verzeichnis /dev/ angesprochen. Beispiel: /dev/sda

```
root@seclab05:/home/stefan# partx -s /dev/sda
NR  START      END  SECTORS SIZE NAME UUID
 1   2048     4095     2048   1M      d3d6a653-f1b8-496a-b95b-8a3bdcab5195
 2   4096   503807   499712 244M      635b3f01-f32e-4039-af81-49b8157acac0
 3 503808 9762240511 9761736704 4.6T      468810d1-0d29-4be9-baec-951cbaaf1e57
```

- Die Spalte NR zeigt die Nummer der Partition.
- Die Spalten START und END geben den ersten und letzten Sektor der Partition an.

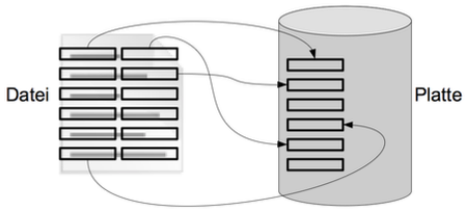
## Das Datei- und Verzeichniskonzept

### *Zweck von Dateien*

- **Abstraktionsschicht** für den Benutzer (oder das Anwendungsprogramm) auf den logischen Blocknummern einer Partition. *(human readable)*
- **Dateien** (engl. files) bezeichnen zusammengehörige Informationen. *(Datei -> Blocknummer)*
  Adressierung erfolgt durch Datei-Name statt (Block-)Nummern.
- **Verzeichnisse** (engl. directories) erlauben die Gruppierung von Dateien.
- Schütze Informationen in Dateien durch den Einsatz von **Zugriffskontrollen** (engl. access control).

### Dateien vs. Blöcke

- Dateien sind Folgen von Zeichen (Bytes)!
- Jede Datei wird in Bereiche gleicher Größe aufgeteilt. Diese Bereiche werden dann in Blöcke auf der Festplatte geschrieben.




### Verzeichnisse

- Dateien werden in **Verzeichnissen** (engl. directory) zusammengefasst.
- Ein **Dateisystem** hat ein „oberstes“ Verzeichnis, das **Wurzel-Verzeichnis** (engl. root directory).
  Name des Wurzelverzeichnisses ist in UNIX: / MS Windows: C:\
- Verzeichnisse können rekursiv weitere (Unter-)Verzeichnisse enthalten.
- Jedes Verzeichnis (außer der Wurzel) hat ein übergeordnetes Verzeichnis genannt „**Elternverzeichnis**“ (engl. parent directory).
- Es ergibt sich ein **eindeutiger Pfad von der Wurzel** zu jeder Datei. /home/wohlfeil/OSN1/WS1617/unix-intro.pdf

### Pfadnamen und Arbeitsverzeichnis

Das Wurzelverzeichnis hat den Namen / (Schrägstrich).

**Absoluter Pfadname**: Pfad von Wurzel durch Verzeichnisse zur Datei

- Pfadbestandteile werden durch / getrennt (Windows: \) Beispiel:/log/messages

**Relativer Pfadname**: Beschreibt den Weg vom aktuellen Verzeichnis zur Datei

- Beginnt mit anderem Zeichen als / Beispiel:./log/messages

- **Kurzname für das aktuelle Verzeichnis: . (Punkt)**
- **Kurzname für das übergeordnete Verzeichnis: .. (zwei Punkte)**
- Das **Oberverzeichnis** von / ist / selbst.
- Jeder Prozess ist immer in einem **Arbeitsverzeichnis (engl. working directory)**.

## Kommandos zum Umgang mit Verzeichnissen

`ls`: Dateien im Arbeitsverzeichnis auflisten
`cat <Datei>`: Inhalt einer Datei ausgeben
`less <Datei>`: Dateiinhalt seitenweise ausgeben (Blättern u. Suchen möglich)
`cp <alte Datei> <neue Datei/Verzeichnis>`: Datei kopieren
`mv <alte Datei> <neue Datei/Verzeichnis>`: Verschieben/umbenennen rm <Datei>: Datei löschen
`mkdir <Verzeichnis>`: Verzeichnis anlegen
`rmdir <Verzeichnis>`: Verzeichnis löschen

- Verzeichnis muss leer sein; falls nicht: rm –R löscht rekursiv ein Verzeichnis und alles darin (Vorsicht!)

`pwd`: zeigt akktuelles Arbeitsverzeichnis an
`cd`: wechselt Verzeichnisse
`touch`: erstellt eine neue Datei
`chmod`: gibt Dateien rechte
`ln`: links werden erzeugt

Allgemeinwissen

Lern das bullshit

## Verweise (engl. link)

Es gibt zwei „Arten“ von Verweisen:

(1) **1 Feste Verweise (engl. hard link):** Hierbei wird im zweiten Verzeichnis einfach dieselbe Inode-Nr. eingetragen. (siehe Folie 4-33). Hard Links sind nur innerhalb einer Partition möglich!

(2) **2 Symbolische Verweise (engl. soft link)**: Hierbei wird im zweiten Verzeichnis eine spezielle Datei angelegt, in der nur der Pfadname zur „Originaldatei“ im ersten Verzeichnis steht. Soft links können überall in den Verzeichnisbaum zeigen.

- Wird ein Link geöffnet, so sorgt das Betriebssystem dafür, dass die „Originaldatei“ geöffnet wird.
- In der Praxis werden häufiger soft links benutzt.

Beispiel:

```
ln beispiel.txt hard-beispiel.txt
ln -s beispiel.txt soft-beispiel.txt
```

```
mcwohlfeiltd:test wohlfeil$ ls -li
total 24
31254650 -rw-r--r-- 2 wohlfeil staff 67 24 Aug 11:15 beispiel.txt
31254650 -rw-r--r-- 2 wohlfeil staff 67 24 Aug 11:15 hard-beispiel.txt
31254653 lrwxr-xr-x 1 wohlfeil staff 12 24 Aug 11:16 soft-beispiel.txt@ -> beispiel.txt
```

- Die Option -i zeigt die Inode-Nr. der Dateien mit an.
- beispiel.txt und hard-beispiel.txt haben dieselbe Inode-Nr. 31254650 Vor dem Benutzernamen steht 2, denn es gibt 2 hard links auf diese Datei.
- Der Dateityp für soft links ist durch das l in der ersten Spalte angezeigt. Der soft link hat eine Größe von 12 Bytes, denn der Name auf den verwiesen wird ist 12 Bytes lang.

### *Verweise wieder löschen*

- Bei **hard links** wird der **Dateiname gelöscht** und die Zahl der **hard links auf diese Datei um 1** reduziert.
  - Falls die Zahl der hard links jetzt 0 ist, dann werden auch alle belegten Blöcke der Datei freigegeben.
  - Falls die Zahl der hard links größer 0 ist, dann gibt es diese Datei noch in anderen Verzeichnissen und die belegten Blöcke werden noch gebraucht.
- Bei **soft links** wird die spezielle Datei mit dem Namen der „Originaldatei“ gelöscht.
  - der soft link wird nicht geloescht falls die „Originaldatei“ gelöscht wird.

## Gerätedateien

- In UNIX werden auch Geräte als Dateien modelliert.
- Der Zugriff erfolgt über die normale Dateischnittstelle.
  - Dateien liegen in /dev (Zugriff erfordert root-Rechte)
  - Zugriffe werden an Gerätetreiber weitergeleitet
- Gerätedateien sind entweder blockorientiert (wahlfreier Zugriff auf jedes beliebige Byte) oder zeichenorientiert (nur sequenzieller Zugriff: „Byte-Strom“)
  - **Blockorientiert**: Festplatte, CD/DVD, USB-Stick, . . .
  - **Zeichenorientiert**: Drucker, Maus, Tastatur, Text-Terminal, . . .

## Pipe

- Endpunkt einer Kommunikationsverbindung zwischen Prozessen
  - **„Rohr“** unter bekanntem Namen, in das Daten hineingeschoben werden und aus dem Daten herausgeholt werden können
  - Auch **FIFO** genannt **(„first in, first out“: Warteschlange für Daten**)
- Kann unabhängig von Prozessen als Datei angelegt werden
  - Normaler Pfadname, z. B. /home/wohlfeil/my-fifo
  - Anlegen mit speziellem Kommando: **`mkfifo my-fifo`**
- Kann von Prozessen wie Datei geöffnet, gelesen, geschrieben werden
  - Nur **sequentieller Zugriff** möglich
  - Interner Datenpuffer begrenzter Größe; Schreiber blockiert, sobald Puffer voll; Leser blockiert, solange Puffer leer
  - Mehrere Leser-Prozesse parallel möglich (ebenso Schreiber) Synchronisierungsprobleme, deshalb Mehrfachzugriff besser vermeiden

## Benutzer und Gruppen

- Jeder **Benutzer** eines Systems gehört (mindestens) einer **Gruppe** an.
  - Über Gruppen werden gleiche Zugriffsrechte für viele Benutzer einfach realisiert.
- Der Systemverwalter (**Benutzer root**) legt die Namen der Gruppen fest und weist den Benutzern ihre Gruppe(n) zu
  - Die Login-Gruppe eines Benutzers ist in seinem /etc/passwd-Eintrag definiert. Für normale Benutzer ist das meistens die **Gruppe users**
  - Kommando groups oder id gibt Gruppe(n) des aktuellen Benutzers aus
- Falls ein Benutzer zu mehreren Gruppen gehört, ist eine davon seine **Hauptgruppe** (engl. primary group)
- Nach dem Login entspricht die Hauptgruppe zunächst der Login-Gruppe Hauptgruppe ändern durch Kommando `newgrp <gruppe>`

## Datei-Besitzer (engl. owner) und Datei-Gruppe

- Jede Datei hat **genau einen Benutzer als Besitzer**
- Der **Besitzer** (engl. owner) einer Datei ist der Benutzer, der sie erstellt hat.
  - Ändern darf die Besitzverhältnisse nur **root** mit dem Kommando `chown <neuer Besitzer> <datei>`
- **Jede Datei hat genau eine ihr zugeordnete Benutzergruppe**.
  - Normalerweise ist das die Hauptgruppe des Dateibesitzers beim Anlegen der Datei.
  - Besitzer darf Datei-Gruppe ändern mit Kommando `chgrp <neue Gruppe> <datei>`
- **Die Datei-Gruppe ist eine Gruppe von Benutzern (nicht etwa von Dateien!)**
- Besitzer und Gruppe einer Datei sind Attribute der Datei. Sie spielen die zentrale Rolle beim Schutz des Zugriffs auf diese Datei.

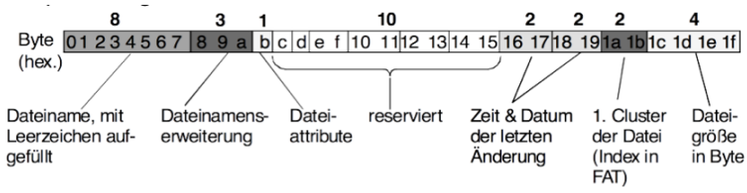
## Zugriffsrechte in UNIX

- Auf jeder Datei sind Berechtigungen definiert. Damit kann der Besitzer unerwünschten Zugriffes anderer Benutzer verhindern.
- **Ein Zugriffsrecht regelt, welche Subjekte welche Operationen auf dem Objekt (also der Datei) ausführen dürfen**.
- Subjekte sind einer von drei „Kreise von Benutzern“:
  - **user**: Der Besitzer der Datei.
  - **group**: Alle Benutzer, die Mitglied der Datei-Gruppe sind.
  - **other**: Alle anderen Benutzer.
- Operationen werden in drei „Operationsarten“ gruppiert.
  - **read**: Dateiinhalt oder Dateiattribute lesen.
  - **write**: Dateiinhalt ändern
  - **execute**: Programmdatei ausführen, Verzeichnis durchlaufen.
- Implementiert durch 9 Bits im Dateiattribut.
- `ls -l` zeigt die Zugriffsrechtebits in Form von 9 Buchstaben
  - Je 3 Bits/Buchstaben für user/group/others
  - `ls -l myfile.txt`
  - -rwxr-xr-- 1 stefan users 0 31. Okt 16:58 myfile.txt Besitzer
  - stefan darf lesen/schreiben/ausführen
  - Gruppe users darf lesen/ausführen (aber nicht schreiben)
  - Andere dürfen lesen (aber nicht schreiben oder ausführen)
- Mit chmod was datei können die Zugriffsbits geändert werden.
  - Dabei ist was eine Folge von drei Zeichen(gruppen):
    (1) u für user, g für group und o für other;
    (2) + für Recht geben oder - für Recht entziehen und
    (3) r für read, w für write oder x für execute.
- Beispiel: `chmod u+w file1 oder chmod go-x file2`

# Implementierung von Dateisystemen

## Aufgaben eines Dateisystems

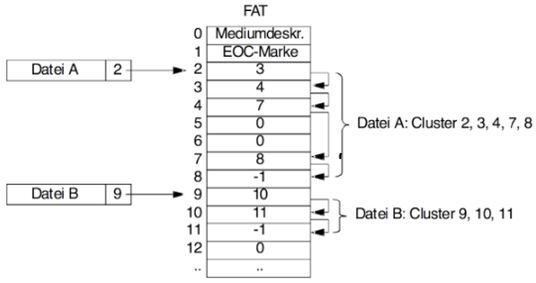
- **Verwalte die Namen der Dateien in einem Verzeichnis**.
  - **Verzeichnisse sind „spezielle“ Dateien, in denen fest definierte Datensätze stehen**.
  - Beispieleintrag aus **FAT-Verzeichnis**




- Verwalte, in welchen Blöcken der Inhalt einer Datei gespeichert ist.
- Verwalte die unbelegten Blöcke der Festplatte.

## File Allocation Table

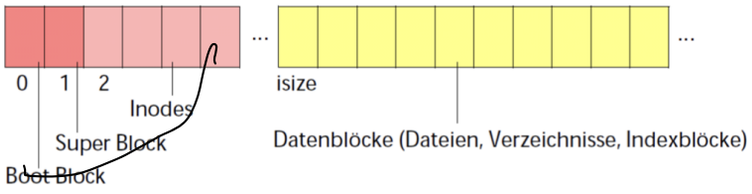
- Jede Platte (Partition) hat genau eine **FAT**.
- Bei n Blöcken (Clustern) hat die FAT n + 2 Einträge.
- In jedem Eintrag steht eine Blocknummer, die des Folgeblocks.




### *Eigenschaften von FAT*

- Einfache Datenstruktur, **sehr leicht zu implementieren**.
- Zugriff auf letzten Block einer großen Datei **benötigt sequentiellen Durchlauf** durch die FAT. Ineffizient bei großen Platten/Dateien.
- ~~Beschränkungen bei den Dateinamen (nur ASCII erlaubt; Länge 8 + 3 Zeichen); in neueren Versionen (VFAT, exFAT) teilweise aufgehoben.~~
- Keine Implementierung von Zugriffskontrollen
- FAT und Wurzelverzeichnis (root directory) sind an fester Stelle auf der Platte gespeichert. Wurzelverzeichnis hat feste Größe, alle anderen Verzeichnisse werden wie „spezielle“ Dateien behandelt.
- ~~**Wird auf USB-Sticks und zum Dateiaustausch immer noch benutzt, da alle Betriebssysteme es implementiert haben.**~~

## Organisation der Blöcke einer Platte/Partition innerhalb Unix und Unix-Derivaten




**Superblock (MBR):** Informationen zum zu startenden System

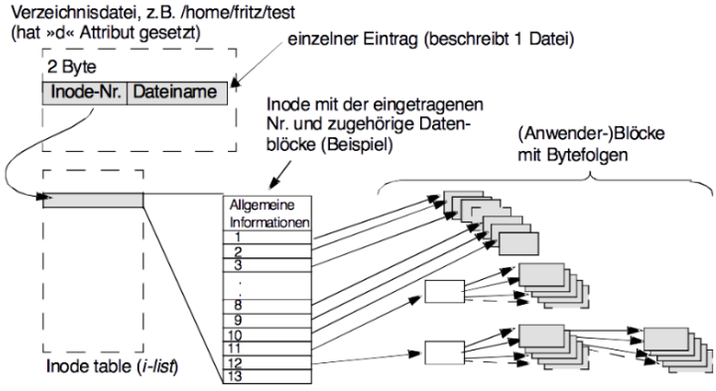
**Superblock**: Verwaltungsinformationen des Dateisystems

- Anzahl der Blöcke, Anzahl der inodes
- Freie Blöcke, freie inodes
- Dateisystemattribute (wie clean oder active)
- Dateisystem-Label, Pfadname des letztet mount point.

## Indexbasierte Dateisysteme

Die Platte (Partition) wird in vier Bezirke eingeteilt:

1. **Boot Block**
2. **Super Block**
3. **iNodes**
4. **(Daten-)Blöcke**



### *Eigenschaften von UNIX-Dateisystemen*

- **Baum-Orientierte Datenstruktur verbessert Zugriffszeiten.**
- Vorhandene Beschränkungen bei Dateinamen *(Länge 255 Zeichen, kein / im Namen)* sind in der Praxis irrelevant.
- **Zugriffskontrollen sind möglich.**
- Komplexe Wiederherstellung nach Absturz erforderlich (File System Check, fsck).
- Superblock enthält Basisinformationen (Name der Platte, Anzahl iNodes, . . . ) und an fester Stelle steht eine iNode-Liste.

## Journaling File Systems

**Zweck**: Vereinfache und beschleunige die Wiederherstellung des Dateisystems nach einem (Rechner-)Absturz.
**Idee**: Halte in einem Journal fest, welche Änderungen anstehen und lösche sie nach Änderungsdurchführung.

*Beispiele für Journaling File Systems:*
*Ext3 (Linux, erweitert ext2), Ext4 (Linux), ReiserFS (Linux), Btrfs (Linux)*
*APFS (Apple), HFS+ (Apple), NTFS (Microsoft), XFS (SGI), JFS (IBM),...*
*ZFS (Sun, integriert RAID und LVM ins Dateisystem)*

## Redundant Array of Inexpensive Disks (RAID)

**Ziele:**

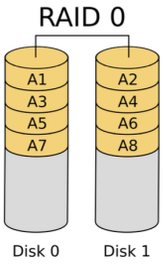
- Beschleunigung des Zugriffs durch Einsatz von **Parallelität**.
- Erhöhung der Zuverlässigkeit durch Einführung weiterer Sicherungs-Codes. Vergrößerung der Kapazität durch Zusammenfassen mehrerer realer Platten.

**Idee:**

- Kombiniere geschickt mehrere einfache Platten. Architektur = Anordnung und Steuerung von Platten
- Jede Architektur wurde Raid Level genannt.

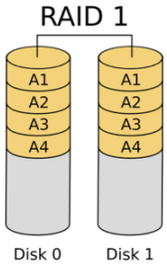
**Die Steuerung der RAID-Systeme erfolgt durch:**

- Einen Controller, sogenanntes **Hardware-RAID** oder **Software-RAID**, also einen Bestandteil des Betriebssystems



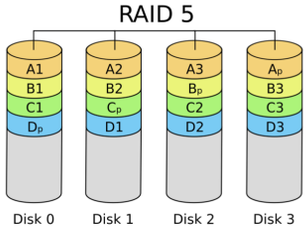

**RAID 0 (Striping)**

- **Keine Redundanz**, also eigentlich kein RAID.
- **Verteile Daten auf zwei Platten**.
- **Größere Geschwindigkeit**, da paralleler Zugriff.
- Verschlechterte Ausfallsicherheit.




**RAID 1 (Mirroring)**

- **Schließe zwei Platten an und benutze eine immer nur als Sicherungskopie**.
- „Verschwendet“ die Hälfte der Kapazität.
- Der Ausfall einer Platte kann kompensiert werden.
- Keine Änderung bei Schreib-/Lese-Geschwindigkeit.




**RAID 5 (Striping with distributed Parity)**

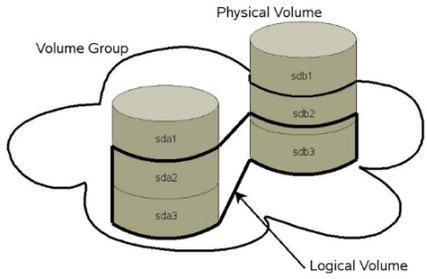
- **Verteile Daten auf mehrere Platten und speichere Paritätsbits**.
- Kapazitätsverluste gering.
- Der Ausfall einer Platte kann kompensiert werden.
- Wiederherstellung dauert.
- Lesegeschwindigkeit > Schreibgeschwindigkeit

## Logical Volume Management (LVM)

**Problem**: Festplattengrößen und -Partitionierungen ändern sich zur Laufzeit nicht. Der Bedarf nach Größenänderungen von Dateisystemen im laufenden Betrieb besteht aber.
**Lösung**: **Füge eine weitere Abstraktionsschicht zwischen Festplattenpartition und Dateisystem ein.**

- Festplatten, bzw. Partitionen werden vom LVM verwaltet.
- LVM bietet dem Betriebssystem (logische) Partitionen an, in denen dann ein Dateisystem installiert wird.
- Die logischen Partitionen kann der LVM dann zur Laufzeit vergrößern oder verkleinern.

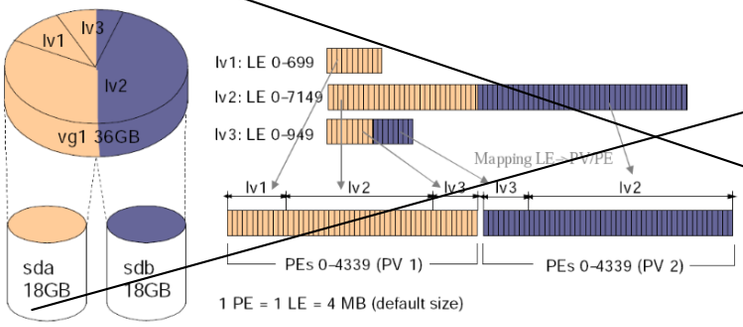
### *Begriffe zum Linux Logical Volume Manager*




- **Physical Volume (PV):** Partition oder Platte.
- **Volume Group (VG):** Menge der physical volumes (PV), die vom LVM verwaltet werden (siehe Wolke).
- **Logical Volume (LV):** Teilmenge einer volume group (VG), die dem Betriebssystem als Partition angeboten wird.

### *Realisierung der Logical Volumes*

**Physical Extend (PE):** Kleinste adressierbare Einheit in einer Volume Group (VG), typische Größe 4 MB.
**Logical Extend (LE):** Kleinste adressierbare Einheit in einem Logical Volume (LV); dieselbe Größe wie Physical Extend.




Í

### *Logical Volumes benutzen*

- Direkt bei der Installation des Betriebssystems sollten Logical Volumes angelegt werden.
- Die Boot-Partition (/boot) sollte nicht in einem LV liegen, denn dann muss das Kernel damit umgehen können.
- Logical Volumes werden typischerweise angelegt für die home Partition /home, die Partition für optionale Software /opt, Server-Daten (z. B. HTML-Seiten) in /srv, Protokolldateien in /var/log/, . . .
- Ein Dateisystem in den Logical Volumes anlegen, das in der Größe dynamisch angepasst werden kann, wie ext3, ext4, XFS oder JFS.

### Festplattenverschlüsselung

- Mit **Festplattenverschlüsselungsverfahren** (engl. full disc encryption (FDE)) lassen sich Datenträger zu teilen oder vollständig verschlüsseln.
- Adressiert das Schutzziel der Vertraulichkeit, nicht das Schutzziel der Integrität
- **Erfolgt in den meisten Fällen transparent**
- Verschlüsselung kann in Hardware (bspw. **Trusted Platform Module (TPM**)) oder in Software (bspw. VeraCrypt) erfolgen
- Meist in modernen Betriebssystemen bereits implementiert
    - BitLocker in Windows
    - dm-crypt in populären Linux-Distributionen (und Android)
    - FileVault in macOS (und ähnlich in iOS)

## Zusammenfassung Dateien und Dateisysteme

- Dateisysteme bieten Benutzern eine abstrakte Schnittstelle zum Zugriff auf persistente Speicher wie z. B. Festplatten.
- Festplatten bestehen aus Zylindern, Sektoren (Blöcke) und Köpfen.
- Dateien sind Bytefolgen, die auch als Folge von logischen Blöcken interpretiert werden können. Sie haben einen Namen und Attribute, z. B. Zugriffsrechte. Verzeichnisse gruppieren Dateien oder Unterverzeichnisse und werden als spezielle Dateien behandelt.
- Verschiedenen Dateisysteme implementieren die Abbildung von Dateiblöcken auf reale Blöcke einer Festplatte, bzw. einer Partition.
- Mit RAID können mehrere Festplatten kombiniert werden, um größere Plattenkapazitäten oder größere Ausfallsicherheit zu bekommen.
- Mit einem Logical Volume Manager lässt sich dem Betriebssystem eine Abstraktion von realen Partitionen, bzw. Festplatten anbieten.