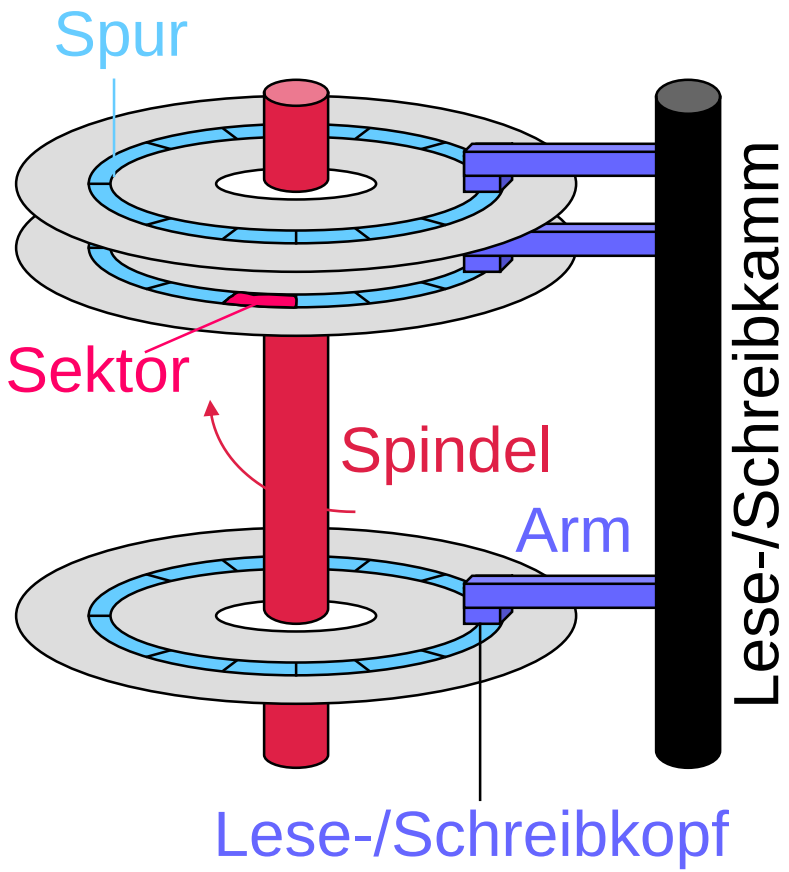
2405 Betriebssysteme
IX. Massenspeicher und Dateisysteme

Thomas Staub, Markus Anwander
Universität Bern

Inhalt

1.1 Magnetplattenspeicher

sammeln aufträge und ordnen um zugriff zu optimieren



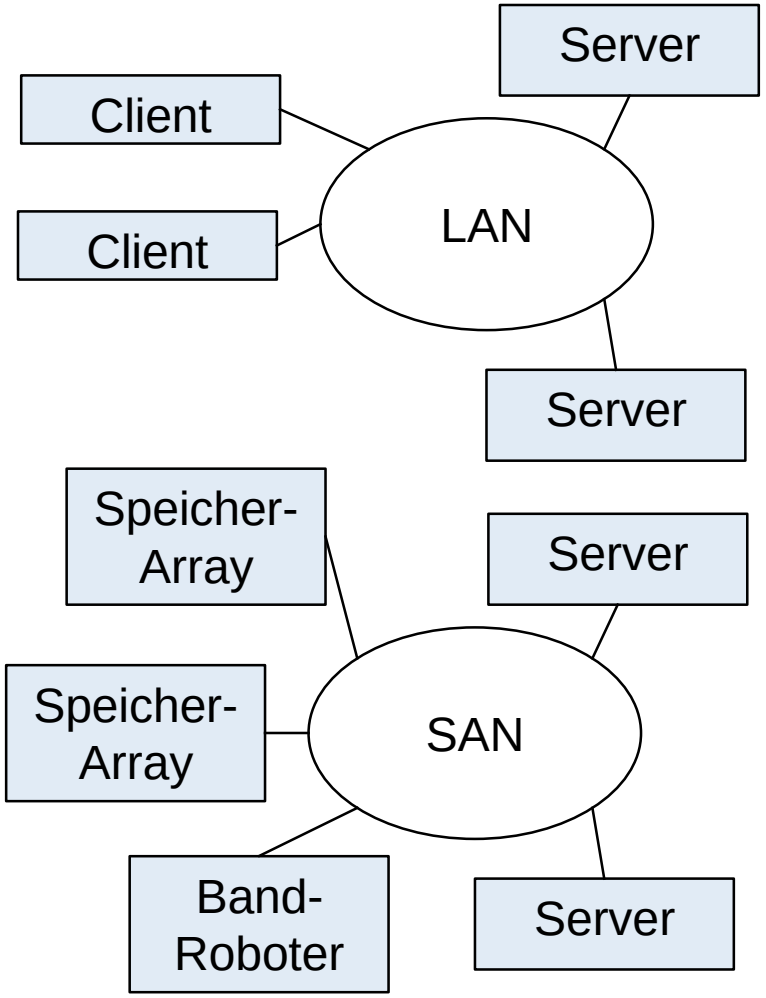
Zylinder = Menge von Spuren mit gleicher Armposition

1.2 Solid State Disks

- Nichtflüchtiger Speicher
  - Dynamisches RAM mit Batterie
  - Flash-Speicher
- Eigenschaften
  - Ggf. höhere Zuverlässigkeit wegen fehlender beweglicher Teile
  - Geringerer Energieverbrauch
  - Schnellerer Zugriff (dadurch oft Systembus als Flaschenhals)
  - Teurer Preis
  - Geringere Lebenszeit
  - Geringere Kapazität

1.3 Anbindung von Disks

- Host-Anbindung
  - Zugriff über lokale E/A-Ports bzw. E/A-Busse
  - Beispiele: IDE, (S)ATA, SCSI
- Netz-Anbindung
  - über lokales Netz (local area network, LAN) zugreifbare Disk
    - Remote Procedure Calls (RPCs)
    - iSCSI: SCSI über IP
- Storage Area Network (SAN)
  - dediziertes Netz mit speziellen Protokollen für Disk-Zugriff
  - Beispiele: Fibre Channel, Infiniband

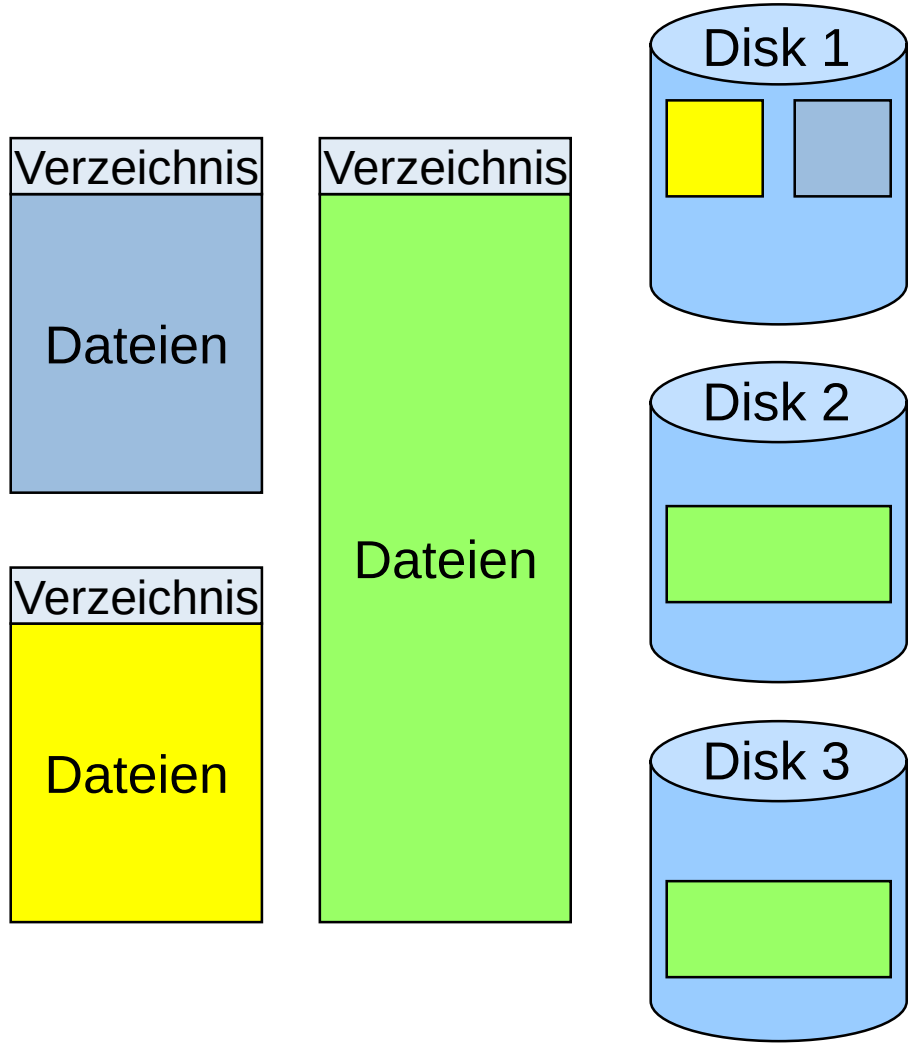


2.1 Formatierung

- Physikalische (Low-Level) Formatierung
  - Unterteilen einer Disk in Sektoren,
    um dem Controller Lese- und Schreiboperationen auf diese zu erlauben
  - Erzeugen von Sektor-Datenstruktur (Header – Daten – Trailer)
    - Header und Trailer enthalten Sektornummer, fehlerkorrigierenden Code etc.
- Logische Formatierung
  - Einrichten von Verwaltungsdatenstrukturen für das Betriebssystem
    - Partitionierung
    - Erzeugen des Dateisystems

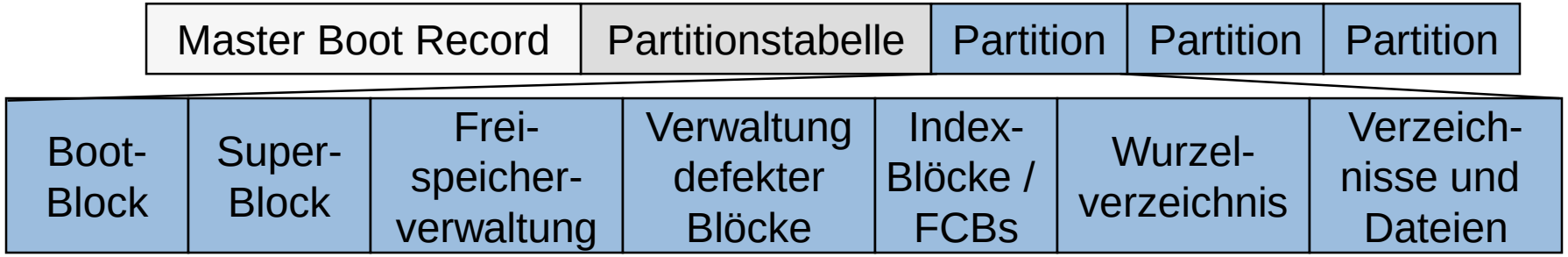
2.2.1 Partitionen

- Partitionierung
  - Gruppierung von Zylindern
- Partitionen (minidisks, volumes)
  - zur Verwaltung einer grossen Anzahl von Dateien
  - entsprechen logischen Laufwerken
  - z.B. separate Partitionen für Betriebssystem, Programme, Daten



2.2.2 Partitionierung

- Master Boot Record
  - in Sektor 0 einer Disk
  - Code zum Booten eines Rechners: Lokalisieren der aktiven Partition und Ausführen des Boot-Blocks
- Partitionstabelle
  - beschreibt Anfang und Ende der Partitionen.
  - 1 Partition wird als aktiv gekennzeichnet.
- Partition
  - Einfaches Programm im Boot-Block lädt Betriebssystem aus dieser Partition.
  - Super-Block (auch Volume Control Block) enthält grundlegende Informationen über Datenträgeraufbau (z.B. Datenträger-, Blockgrösse)
  - Freispeicherliste und Liste fehlerhafter Blöcke
  - Datenstrukturen zur Speicherallokation, Indexblöcke, File Control Blocks (FCBs)
  - Wurzelverzeichnis, Verzeichnisse, Dateien

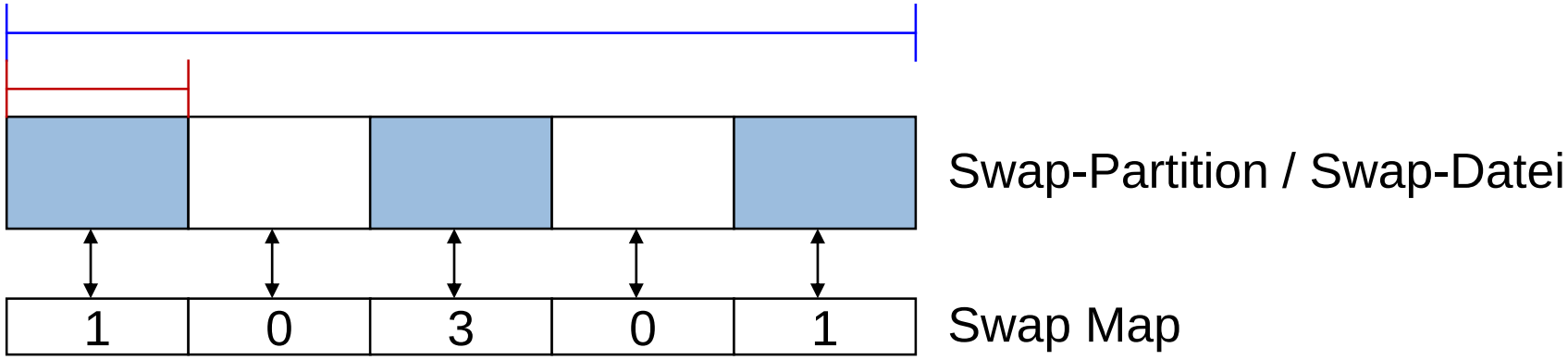


2.3 Behandlung fehlerhafter Blöcke

- Disks sind fehleranfällig
  - Köpfe schweben über der Oberfläche.
- teilweise fehlerhafte Blöcke bei Auslieferung
  - Erstellen einer Liste fehlerhafter Blöcke bei Formatierung
  - MS-DOS: format, chkdsk: spezielle FAT-Einträge für fehlerhafte Blöcke
- Sector Sparing (Forwarding)
  - Controller hat Liste „schlechter“ Blöcke (bad blocks) und leitet Zugriffsversuch auf fehlerhaften Block (unsichtbar für Betriebssystem) auf einen Reserveblock um.
  - Auswirkungen auf Disk-Scheduling !
  - Reserveblöcke in allen Zylindern und/oder in einem Reservezylinder
- Sector Slipping
  - Verschieben von Sektoren in einer Spur

3. Swap-Space-Management

- Swapping und Paging benötigen Platz auf Disk.
- Swapping-Lastverteilung auf verschiedene Disks
- Normale Dateioperationen für Swapping sind ineffizient.
  - Verzeichnishierarchie, externer Verschnitt und Fragmentierung erhöhen Zeit für Swapping.
- Ansatz
  - zusammenhängende Swap-Bereiche in speziellen Disks oder Partitionen mit fester Grösse
  - Geschwindigkeits-optimierte Speicherallokation durch Swap-Space Storage Manager, ggf. interner Verschnitt
  - Swap Space nicht für Daten aus Dateien, die nur gelesen werden, z.B. Code, sondern eher für Daten, Stack, Heap
- Beispiel: Linux
  - Ein oder mehrere Swap Areas in Dateisystem oder „roher“ Swap Partition
  - Swap Area mit 4 kB Page Slots
  - Swap Map zur Verwaltung: Wert zeigt an, wie viele Prozesse den Slot benutzen (shared memory)



4. Zuverlässigkeit: Redundant Array of Inexpensive Disks

- Hohe Fehleranfälligkeit von Disks

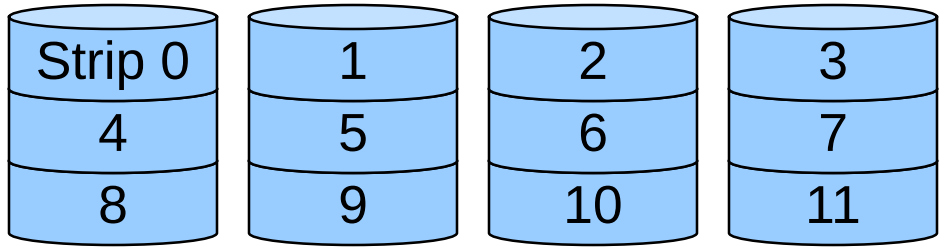
⇒ Redundanzmechanismen

⇒ Redundant Array of Inexpensive Disks (RAID)

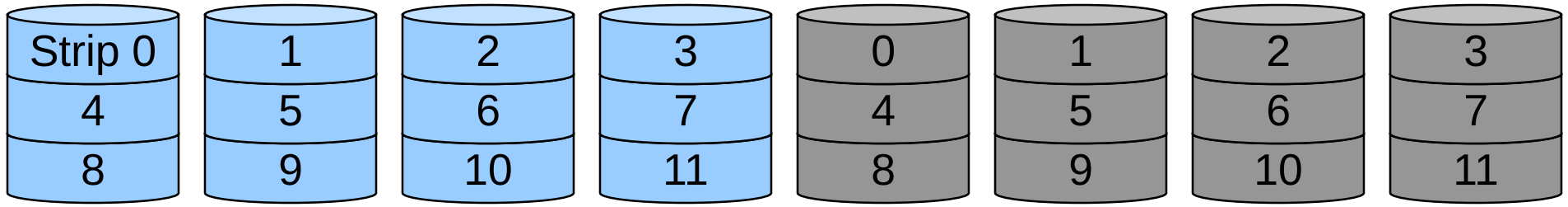
- RAID-0: Disk Striping (Interleaving)
  - Jede Disk enthält Streifen (Strip = k Sektoren) der virtuellen Disk.
  - Block-Level Striping: Verteilen 1 Datei auf N Disks → paralleler Transfer von/zu Disks
- RAID-1: Mirroring (Shadowing): Duplikate
- RAID-2: Bit-Interleaving: Nibble → 7-Bit-Hamming-Code
- RAID-3: vereinfachte RAID-2 Version (nur Paritätsbit)
- RAID-4: Paritätsblöcke (Block Interleaved Parity, Block-Level Striping)
- RAID-5: verteilte Speicherung von Daten und Redundanzinformation (Paritätsbits)
- RAID-6: wie RAID-5 aber mit zusätzlicher Redundanzinformation für den Fall mehrerer Disk-Fehler
- RAID-0+1 / RAID-1+0: Kombination von RAID-0 und RAID-1

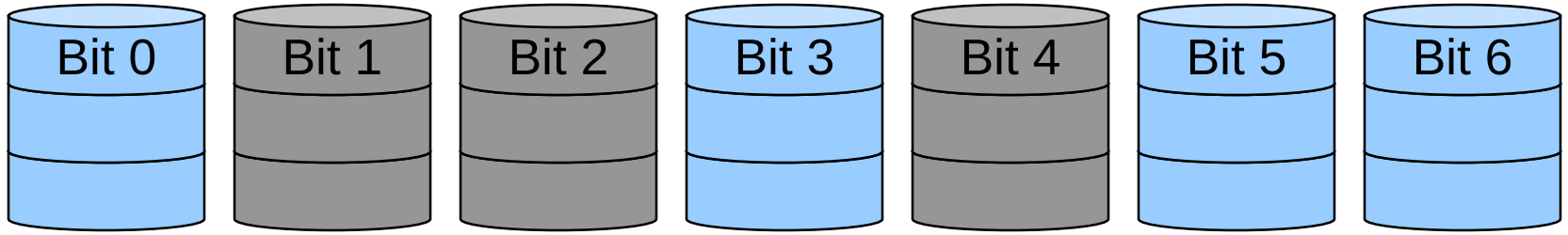
- Weiterer Vorteil von RAID: paralleler Zugriff auf Disks (Disk Striping)

4.1.1 RAID-0, RAID-1, RAID-2



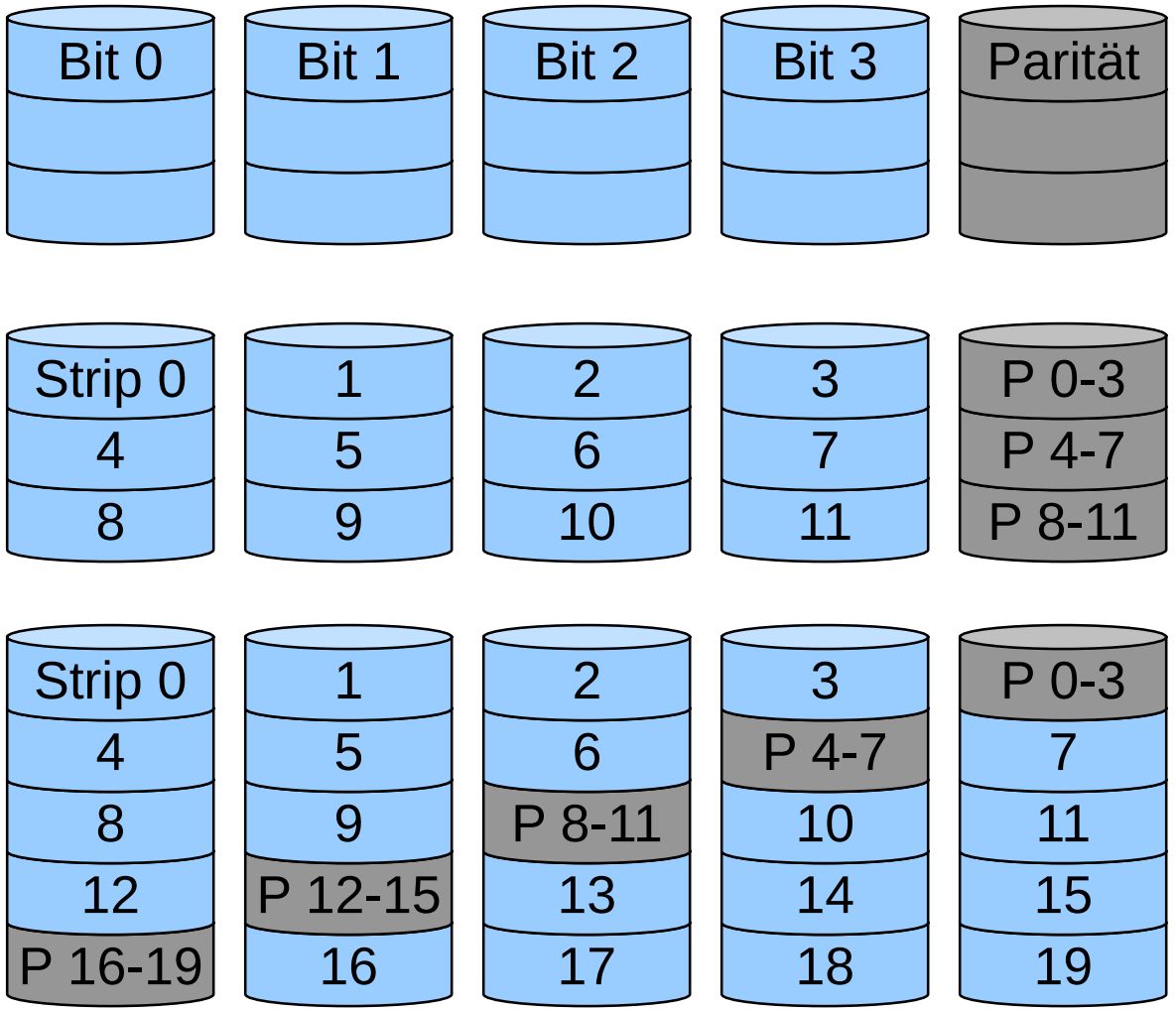
1 Strip (Streifen) = k Sektoren





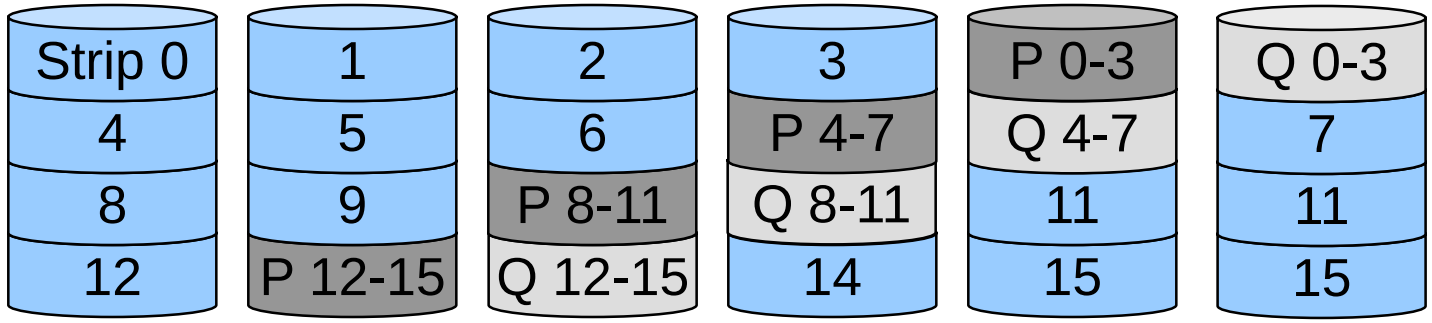
berechnen graue aus blauen

4.1.2 RAID-3, RAID-4, RAID-5

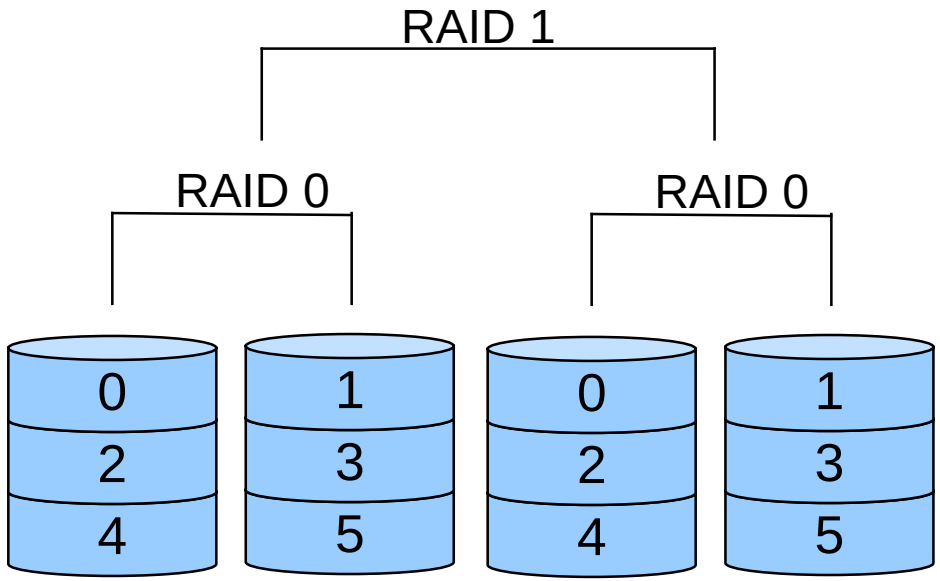


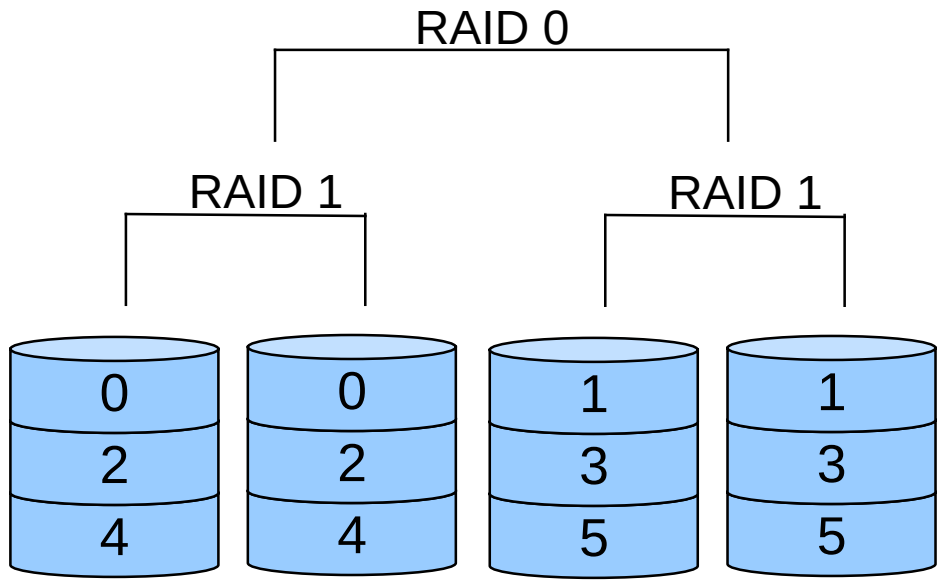
4.1.3 RAID-6

zwei fehler korrigierbar



4.1.4 RAID-0+1 und RAID-1+0





raid 1+0 zuverlässiger, da bei
0+1 fstplatte verlust zu doppeltem verlust führt

4.2 Auswahl geeigneter RAID-Verfahren

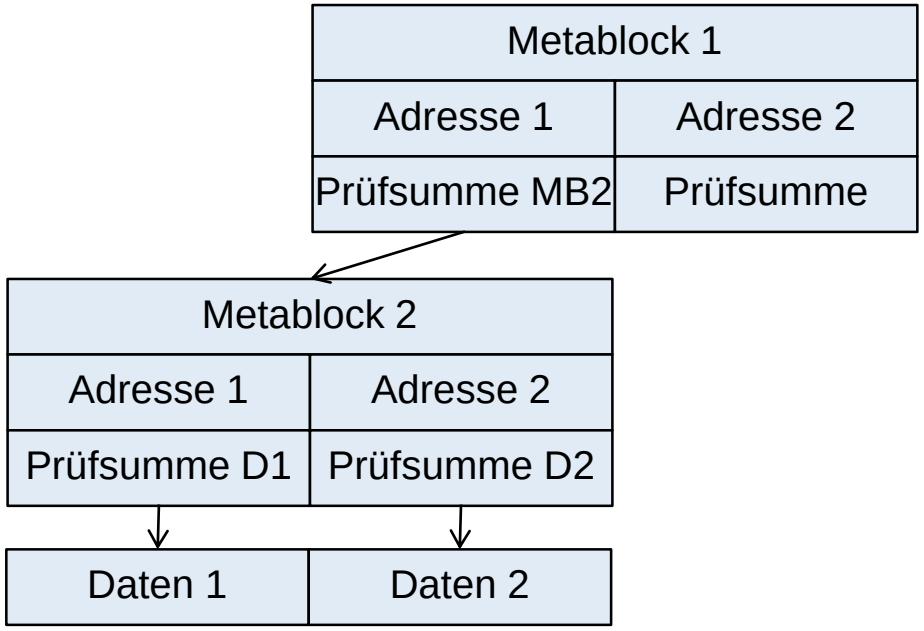
- Wiederherstellung bei RAID-1 einfach, da nur Kopieren von einer Disk auf die andere. Ansonsten müssen alle Disks einbezogen werden.
- RAID-0 für hohe Leistungsanforderungen
- RAID-1 für Anwendungen mit hohen Anforderungen an Zuverlässigkeit und schneller Wiederherstellung
- RAID 0+1 und 1+0 für hohe Leistungsfähigkeit und Zuverlässigkeit
- RAID 5 für grosse Datenvolumina wegen geringerem Overhead

4.3 RAID-Implementierung

- in Software als Teil des Betriebssystems
  (Verwendung eher einfacher Verfahren wie RAID 0, 1, 0+1)
- auf Host-Bus-Adapter (geringe Kosten, aber wenig flexibel)
- in Hardware des Speicher-Arrays
- in SAN-Interconnect

4.4 Probleme mit RAID

- RAID schützt vor physikalischen Fehlern auf dem Speichermedien, aber nicht vor anderen, durch Hardware oder Software verursachten Fehlern, z.B. korrupte Daten aufgrund falscher Zeiger o.ä.
- Mögliche Abhilfe (Beispiel ZFS):
  - Prüfsummen über Daten und Metadaten
  - Speichern der Prüfsumme eines Blocks zusammen mit dem Zeiger auf diesen Block
  - Mögliche Korrektur falls Block gespiegelt vorhanden ist



5.1 Begriff Dateisysteme

- dienen der dauerhaften und persistenten Speicherung von Programmen und Daten auf Sekundärspeichern
- müssen Anwendungsprogrammen effizienten Zugriff auf gespeicherte Daten erlauben.
- Abstraktionen
  - Datei: Behälter für die Speicherung beliebiger Information
  - Verzeichnis: vom Dateisystem verwaltete Dateien zur Strukturierung externer Speichermedien

5.2 Anforderungen an Dateisysteme

- Prozesse benötigen lesenden oder schreibenden Zugriff auf Dateien.
- Zugang über Namen
- Ein oder mehrere Benutzer sollen zugreifen dürfen.
  → Ordnungs- und Strukturierungsverfahren
- Mehrbenutzerbetrieb → Zugriffsrechte
- Typischer Dateizugriff
  - kleine Dateien
  - Lesezugriff dominiert
  - sequenzieller Zugriff
  - Benutzung durch meist 1 Programm oder Person

6. Dateien

- Datei
  - logische Speichereinheit, die vom Betriebssystem auf physikalische Geräte abgebildet wird
  - Behälter für die dauerhafte Speicherung von Informationen
  - zusammenhängender logischer Adressraum
- Dateitypen
  - Daten
  - Programme
  - Dokumente
  - Bilder
  - …
- Dateistruktur hängt vom Dateityp ab.
  - Text: Sequenz von Zeichen
  - Quelldatei:
    Sequenz von Subroutinen und Funktionen
  - ausführbare Datei:
    Sequenz von Code-Sektionen

- Dateiattribute
  - Name
    - z.B. example.c
  - Identifikator
    - eindeutige Nummer zur Identifikation der Datei in einem Dateisystem
  - Typ
    - z.B. Text-, Binär-, Verzeichnis-Dateien, ...
  - Lokation
    - Zeiger auf Gerät und Geräte-spezifische Information zum Auffinden
  - Grösse
    - Grössenangabe in Bytes, Worten, Blöcken; Maximalgrösse
  - Besitz- und Zugriffsrechte
    - d.h. wer darf lesend, schreibend oder ausführend zugreifen
  - Zeit- und Benutzerinformation
    - Information über letzte Benutzung, Schreiben oder Lesen inklusive Information über den jeweiligen Benutzer

6.1.1 Dateizugriffsoperationen

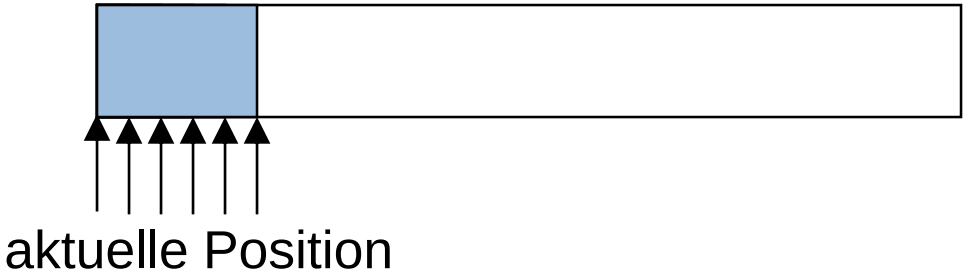
- Erzeugen und Öffnen einer Datei
  - Dateisystem
    - lokalisiert / erzeugt Datei auf externem Speicher.
    - initialisiert interne Datenpuffer für anschliessenden Zugriff.
    - überprüft Zugriffsrechte.
  - Beispiel (POSIX): `fd = open (filename, flags, mode); fd: file descriptor`
- Lese- oder Schreibzugriff
  - Beispiele: `m = read (fd, buffer, max_n); m = write (fd, buffer, n)`
- Positionierung
  - Beispiel: `m = lseek (fd, offset, whence)`
- Schliessen
  - Freigabe von Ressourcen, z.B. Datenpuffer
  - Freigabe des Zugriffs für andere Prozesse bei exklusivem Zugriff (ansonsten automatische Freigabe bei Prozessterminierung)
  - Beispiel: `m = close (fd)`
- Löschen
- Abschneiden (Truncating): Beibehalten der Attribute, aber (teilweise) Löschen von Daten
- Anhängen von Daten
- Lesen und Setzen von Attributen
- Umbenennen

6.1.2 Beispiel: Dateizugriff

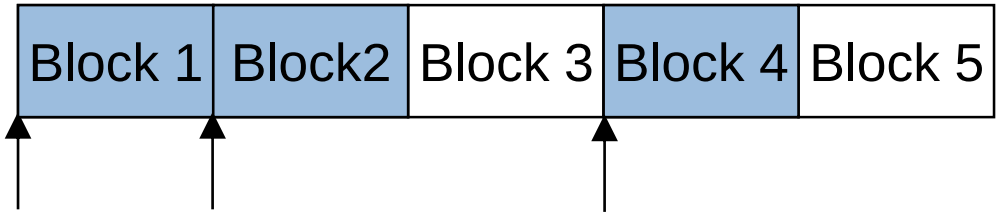

```
/* File copy program. Error checking and reporting is minimal. */

#include <sys/types.h>                  /* include necessary header files */
#include <fcntl.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char *argv[]);       /* ANSI prototype */

#define BUF_SIZE 4096                   /* use a buffer size of 4096 bytes */
#define OUTPUT_MODE 0700                /* protection bits for output file */

int main(int argc, char *argv[])
{
      int in_fd, out_fd, rd_count, wt_count;
      char buffer[BUF_SIZE];

      if (argc != 3) exit(1);           /* syntax error if argc is not 3 */

      /* Open the input file and create the output file */
      in_fd = open(argv[1], O_RDONLY);    /* open the source file */
      if (in_fd < 0) exit(2);             /* if it cannot be opened, exit */
      out_fd = creat(argv[2], OUTPUT_MODE);  /* create the destination file */
      if (out_fd < 0) exit(3);            /* if it cannot be created, exit */

      /* Copy loop */
      while (TRUE) {
            rd_count = read(in_fd, buffer, BUF_SIZE); /* read a block of data */
            if (rd_count <= 0) break;           /* if end of file or error, exit loop */
            wt_count = write(out_fd, buffer, rd_count); /* write data */
            if (wt_count <= 0) exit(4);         /* wt_count <= 0 is an error */
      }

      /* Close the files */
      close(in_fd);
      close(out_fd);
      if (rd_count == 0)                  /* no error on last read */
            exit(0);
      else
            exit(5);                      /* error on last read */
}
```

6.2 Dateizugriffsmethoden

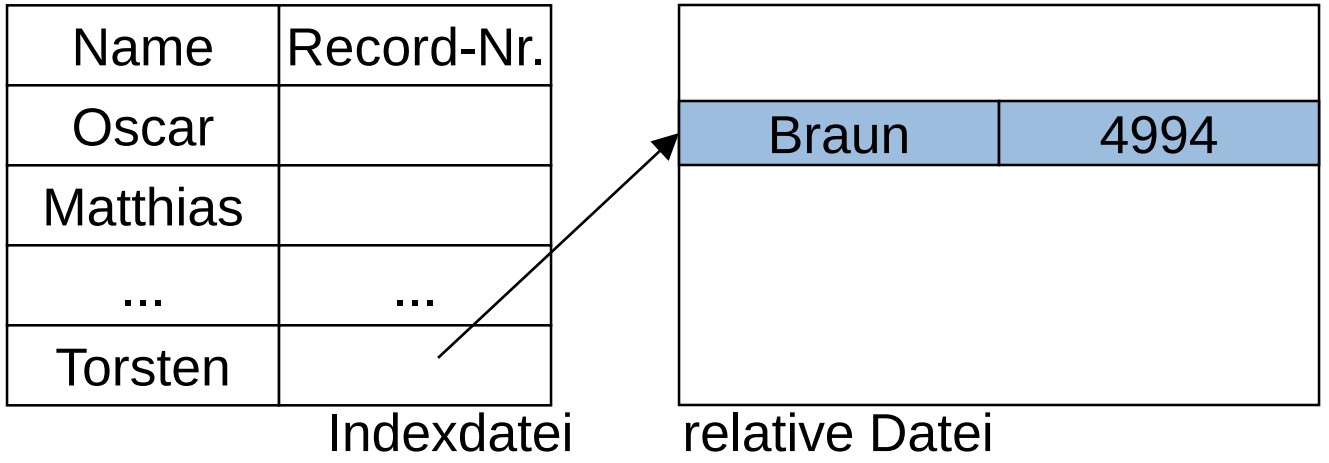
- sequenziell



- direkt

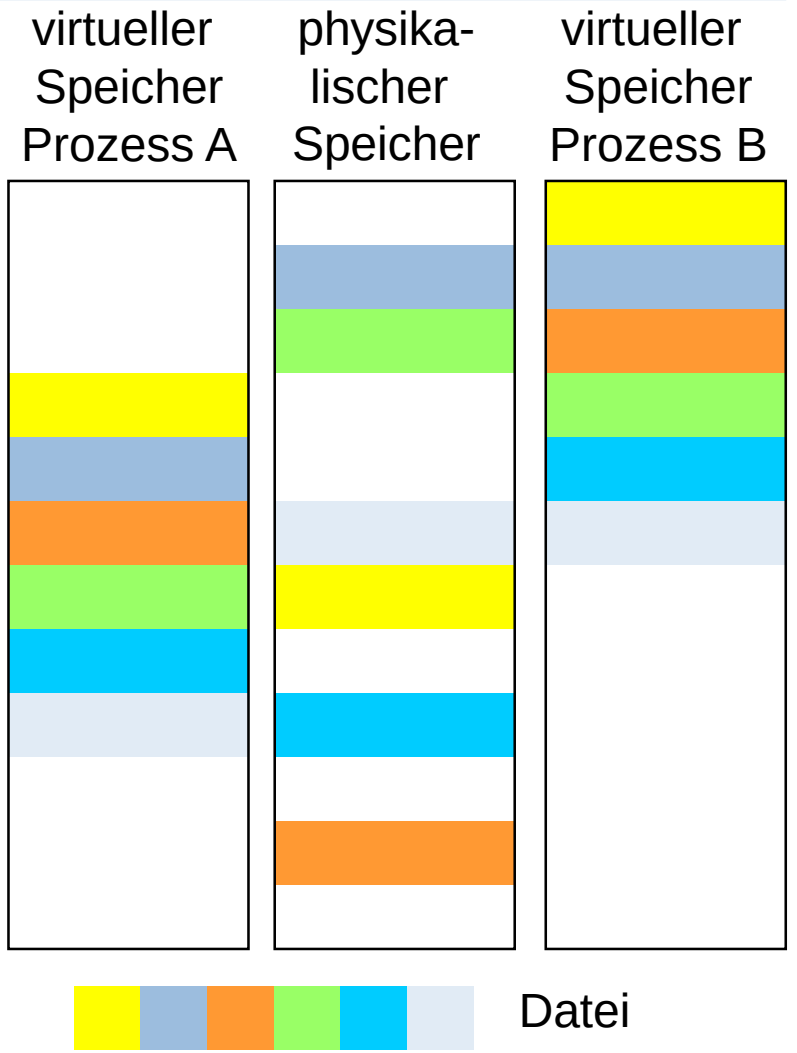


- indiziert
  - basiert auf direktem Zugriff



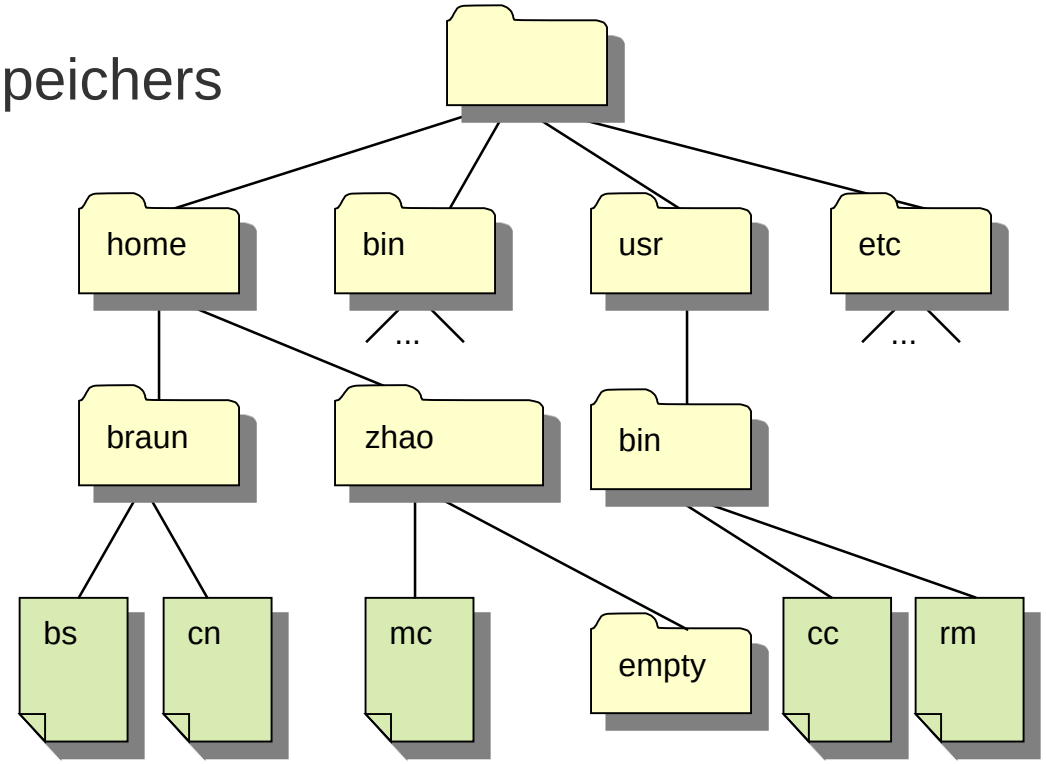
6.3 Speichereinblendung von Dateien

- Memory Mapped Files
- (Teil einer) Datei wird in virtuellen Adressraum eines Prozesses eingeblendet (POSIX: mmap).
- sequenzieller oder wahlfreier Zugriff durch Lese- oder Schreibinstruktionen
- Implementierung
  - Bestimmen eines genügend grossen Bereichs im virtuellen Adressraum (z.B. zwischen Heap und Stack)
  - Seitentabellendeskriptoren zeigen auf Blöcke der einzublendenden Datei
  - Prefetching oder Laden bei Seitenfehler
  - Zurückschreiben des Speicherinhalts auf Disk bei Schliessen der Datei



7. Verzeichnisse

- Directories
- zur hierarchischen Strukturierung des externen Speichers
- Verzeichnis kann weitere Verzeichnisse (Unterverzeichnisse) oder Dateien enthalten.
- Eltern- und Wurzelverzeichnisse
- Arbeits- und Heimatverzeichnis
- baumartige Verzeichnisstruktur
- meistens: Verzeichnis-Implementierung als Datei
- Modifikation beim Erzeugen, Löschen, Ändern von Dateien

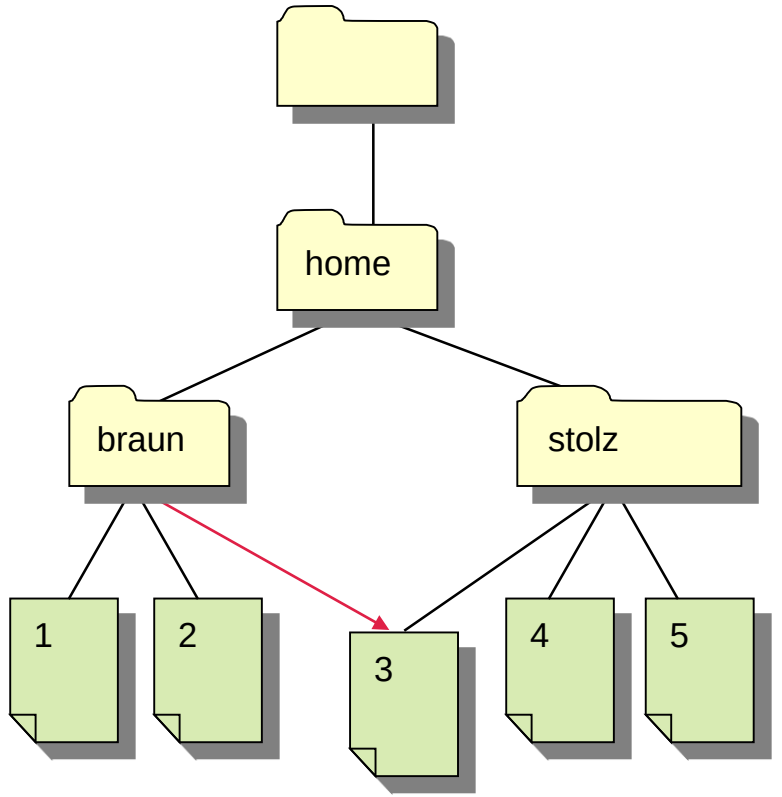


7.1 Operationen auf Verzeichnissen

- Erzeugen von Verzeichnissen (UNIX: create)
- Löschen von Verzeichnissen (delete)
- Öffnen um Lesen von Verzeichnissen (opendir)
- Schliessen von Verzeichnissen nach dem Lesen (closedir)
- Lesen des nächsten Eintrags in einem geöffneten Verzeichnis (readdir)
- Umbenennen (rename)
- Erzeugen und Löschen von Links (link, unlink)
- Erzeugen und Löschen von Dateien beeinflussen auch Verzeichnisse.

7.2.1 Links

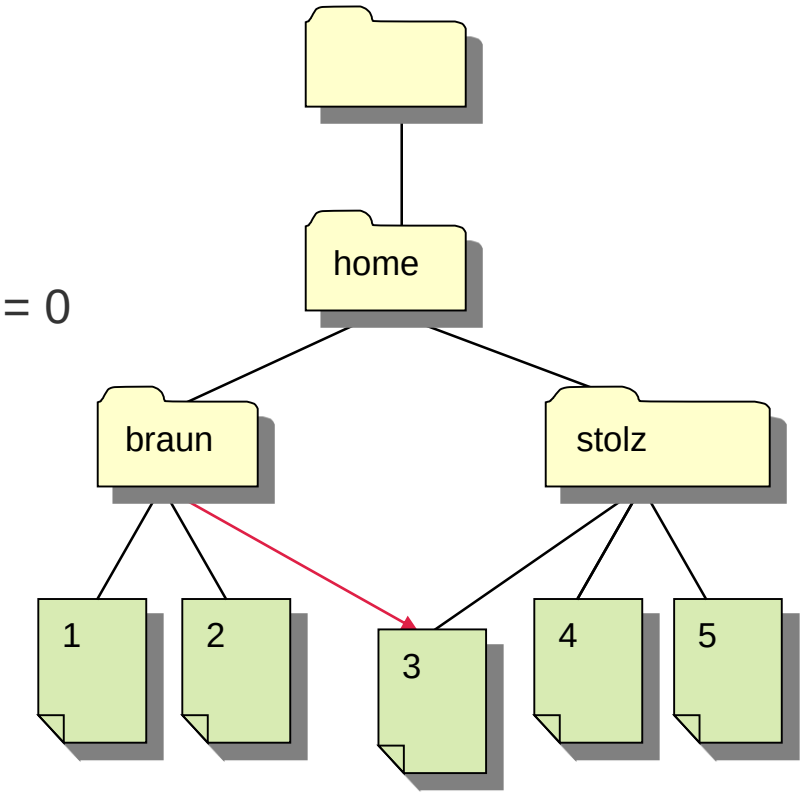
- Links verweisen auf andere Dateien oder Verzeichnisse.
- Einmal vorhandene Dateien/Verzeichnisse können dadurch an mehreren Stellen in der Verzeichnisstruktur erscheinen.
- mehrere verschiedene Namen für eine Datei (Aliasing)
- harte Links
  - Verzeichnisse enthalten Zeiger auf mit Datei verbundenen Datenstrukturen (File Control Blocks, z.B. i-nodes), welche Zeiger auf Disk-Blöcke enthalten
  - nicht vom Original-Verzeichniseintrag unterscheidbar
  - nur für Dateien eines Dateisystems, nicht für Verzeichnisse (→ Vermeiden von Zyklen)
- symbolische Links
  - Erzeugen einer neuen Datei vom Typ „Link“
  - Datei enthält Namen der Original-Datei.
  - Zugriff wird auf diese umgeleitet.
  - Zeiger über Disks oder Computer hinweg.



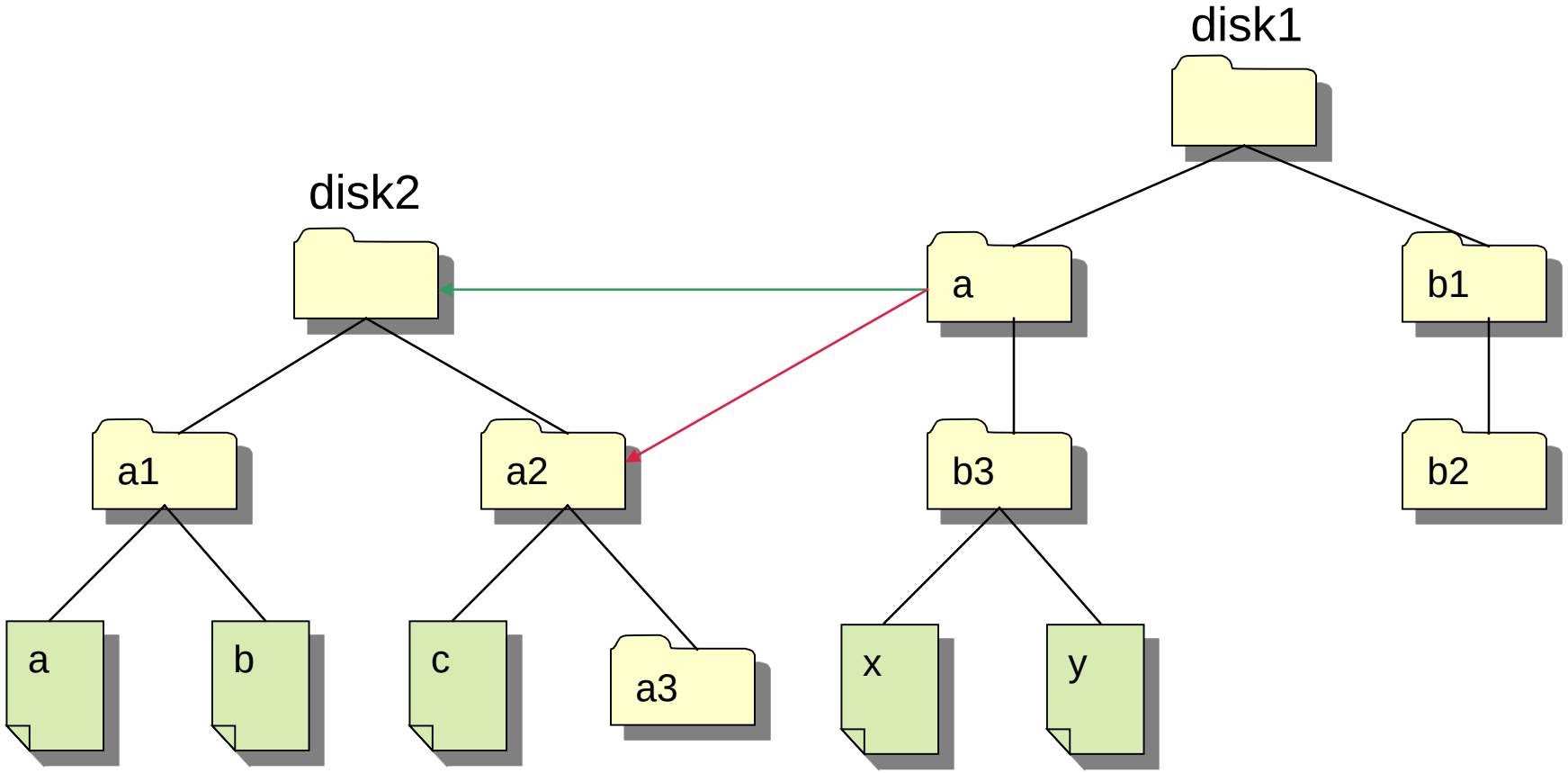
ln /home/stolz/3 /home/braun/3

7.2.2 Löschen von Links

- Problem
  - Benutzer stolz löscht 3.
- Lösungen
  - Referenzzähler in File Control Block (harte Links)
    - Dekrementieren des Referenzzählers beim Löschen
    - Datei oder Verzeichnis wird nur gelöscht, falls Referenzzähler = 0
  - Benutzer müssen Link selbst löschen (symbolische Links).



7.3 Mounting


disk1
disk2
a
b1
a1
a2
b3
b2
a
b
c
a3
x
y
mount disk2:/ a/
mount disk2:/a2 a/

8. Zugriffsschutz

- Erzeuger oder Eigentümer einer Datei bzw. eines Verzeichnisses sollte kontrollieren können, wer mit welchen Möglichkeiten darauf zugreifen kann.
- Zugriffsrechte
  - Lesen
  - Schreiben
  - Ausführen
  - Anhängen
  - Löschen
  - Auflisten

8.1 Zugriffsrechte unter UNIX

- Zugriffsrechte: Lesen, Schreiben, Ausführen (Read, Write, eXecute)
- 3 Benutzerklassen
  - Eigentümer (user)
  - Gruppe (group)
  - öffentlich (others)
- 750 = 111 101 000
  - Eigentümer darf lesen, schreiben und ausführen.
  - Gruppe darf lesen und ausführen.
  - Sonstige Benutzer haben keinen Zugriff.
- **`chmod 750 tmp`**
  - ordnet Verzeichnis tmp das Zugriffsrechtemuster 750 zu
- **`chmod go+r tmp`**
  - erlaubt zusätzlich Gruppe und allen anderen Benutzern Lesezugriff
- **`chgrp cds tmp`**
  - tmp gehört zur Gruppe cds