DATEISYSTEM

Einführung

Eine Datei ist ein Bestand meist inhaltlich zusammengehöriger Daten, der auf einem Datenträger oder Speichermedium gespeichert ist. Diese Daten können somit über die Laufzeit eines Programms hinaus existieren und werden als ***persistent*** *bezeichnet – sie sind bei Programmende* ***nicht verloren****.*

- **Dateien** dienen dazu, Informationen persistent abzuspeichern.
- **Dateien** werden typischerweise auf Speichermedien wie Festplatten, USB-Sticks, Disketten, CD-ROMS, etc. gespeichert.
- Ein Speichermedium kann verallgemeinert als eine lineare Folge von Blöcken aufgefasst werden.
- **Verzeichnisse** dienen zur Organisation von Dateien.
- Das Betriebssystem hat die Aufgabe die **Zuordnung** von Dateien zu Blöcken und Verzeichnissen zu organisieren und zu steuern.

Einführung

- Ein **Betriebssystem** stellt mit dem Konzept einer Datei eine Abstraktion einer Menge von Blöcken zur Verfügung.
- Das **Dateisystem** hat die Aufgabe die Zuordnung von Dateien zu Blöcken und Verzeichnissen zu organisieren und zu steuern.
- Dateisysteme unterscheiden sich darin, wie Dateien
  - gegenüber dem Benutzer **repräsentiert** werden (Dateinamen, Dateiattribute, Dateitypen)
  - und wie sie abhängig vom Speichermedium **implementiert** sind
- Der **Inhalt** von Dateien ist für das Dateisystem nicht relevant, die Interpretation der Inhalte erfolgt durch Anwendungsprogramme.

Forderungen an das Betriebssystem für persistente Speicherung

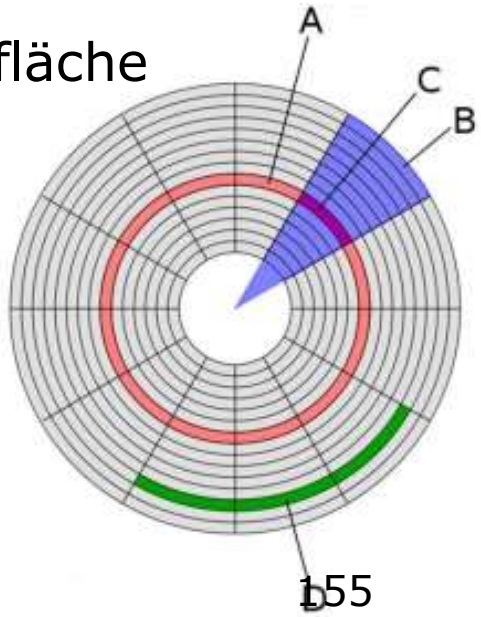
1. Es muss möglich sein, eine große Menge von Informationen zu speichern
2. Eine Information muss auch nach der Beendigung des auf sie zugreifenden Prozesses noch erhalten bleiben
3. Mehrere Prozesse müssen gleichzeitig auf die Information zugreifen können.

Realisierungen persistenter Speicherung

- magnetisch: Festplatte, Disketten
- optisch: CD, DVD, Blue-Ray
- elektrisch: Flash

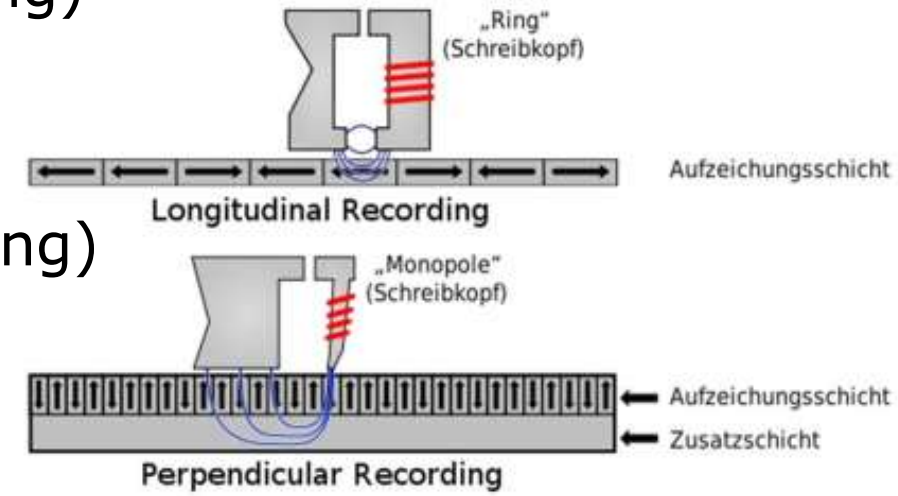
Magnetische Speicherung (Festplatte, Diskette)

- Speichern der Daten auf einer magnetischen Oberfläche
  - Magnetisierung mittels Spulen
  - Restmagnetismus erhält Informationen über längere Zeit (Remanenz)
- strukturlose, magnetische Oberfläche wird durch Formatierung mit einer Zugriffstruktur versehen
- magnetische Oberflächen werden zu Plattenstapeln zusammengefasst
  - Schreib-/Leseköpfe magnetisieren Teile der Oberfläche
  - jede Festplatte besteht aus Spuren (A) (Zylinder) und Sektoren (B)
  - Pro Spur gibt es Blöcke (C) und Sektoren (D)
  - Zusätzlich Zonen, wobei die Blockzahl pro Spur von Zone zu Zone variiert



Magnetische Speicherung

- In der Fläche (longitudinal recording)
  - 15 – 30 GBit pro cm2
  - veraltete Technik (bis ca. 2010)
- In der Tiefe (perpendicular recording)
  - 155 GBit pro cm2
  - spitzer Kopf
  - geringerer Abstand zur Platte



- Mit Wärmeeinwirkung
  - Erhitzen der Oberfläche, um magn. Feldstärken möglichst gering zu halten
  - Einsatz von Lasern
  - Geplant: 20 Terabyte im 2.5" Format (Seagate)

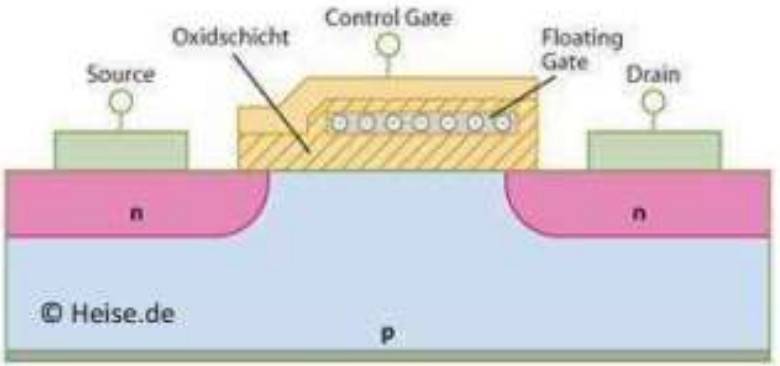
Optische Speicherung (CD, DVD, Blu-Ray)

- Speichern der Daten auf einer reflektierenden Oberfläche
- Varianten
  - CD: max. 900 MB, 1 Schicht, 1- oder 2-seitig
  - DVD: 4.7 GB / 8.5 GB, 1-2 Schichten, 1-seitig
  - Blue-Ray: 25 GB / 50 GB, 1-2 Schichten, 1-seitig
    unter Laborbedingungen 500 GB auf 20 Schichten
- Eine spiralförmige Spur enthält alle Informationen
  - CD: 5.5 km
  - DVD: 12 km (4.7 GB)
  - Blu-Ray: 27.5 km (25 GB)

Speicherkapazität	900MB	4,7/8,5GB	25/50GB
Übertragungsgeschwindigkeit	1,4MBit/s	11Mbit/s	36 Mbit/s
Spurwechselzeit	~ 75ms	~125ms	~180ms

Elektrische Speicherung (Flash)

- Speichern der Daten als elektrische Ladung
  - Einbringen von Elektronen in das Floating Gate
  - Entfernen von Elektronen aus dem Floating Gate
- Organisation
  - Seiten mit 8 KB
  - Blöcke mit 256 Seiten
- Schreiben / Löschen
  - Schreiben häufig seitenweise möglich
  - Löschen nur blockweise möglich, dabei wird Oxidschicht abgenutzt
- Varianten
  - SLC (single level cell): 1 Bit pro Zelle
  - MLC (multi level cell): 3 Bit pro Zelle, über 8 Spannungsniveaus



Elektrische Speicherung

Bauformen

- USB-Sticks
- Compact-Flash
- SD Memory Card
- Solid-State-Disks

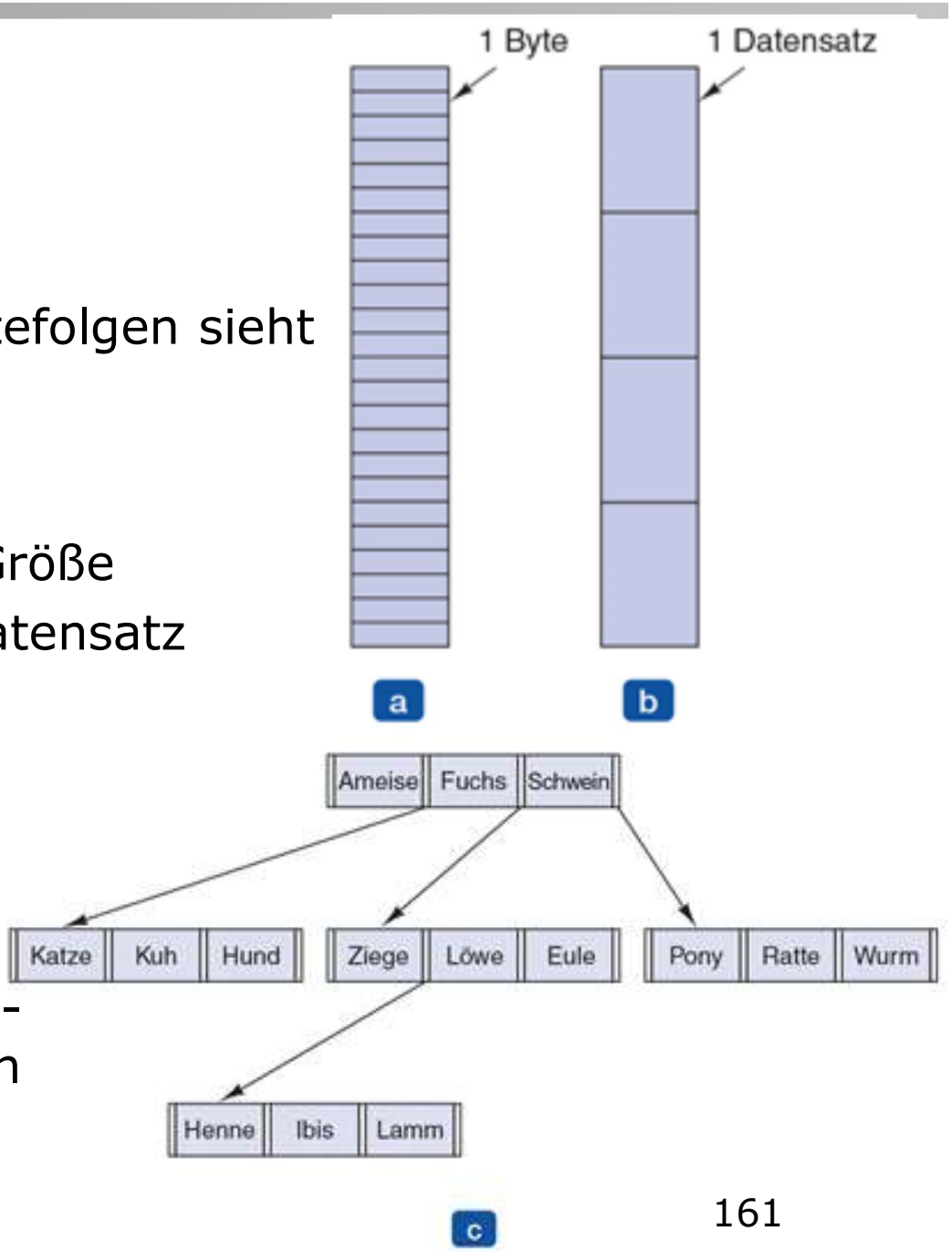
	USB Stick	SD-Card	SSD
Speicherkapazität	1TB	256GB	2TB
Übertragungsgeschwindigkeit	240MB/s	45MB/s	400MB/s

Dateien
Merkmale

- Name
  - Zeichenvorrat, Länge, Case-sensitiv
  - Durch Punkt (.) geteilte Namen (Name + Namenserweiterung)
  - Attribute
- Dateistruktur
  - Byte-Folge
  - Block-Folge
  - hierarchisches Baumsystem
- Dateityp
  - normale Dateien / Verzeichnisse
  - Zeichen- / Blockdateien
  - sequentieller / wahlfreier Zugriff

Dateistruktur

- strukturlos (a)
  - Folge von Bytes
  - max. Flexibilität, da BS nur Bytefolgen sieht
  - Windows, LINUX, etc.
- Satzorientiert (b)
  - Folge von Datensätzen fester Größe
  - kleinste Dateneinheit ist der Datensatz
  - heutzutage nicht verwendet
- indizierte Datensätze (c)
  - Datensätze var. Größe
  - Zugriff mittels Schlüssel
  - z. T. Großrechner, die DB-Funktionen in das Dateisystem legen

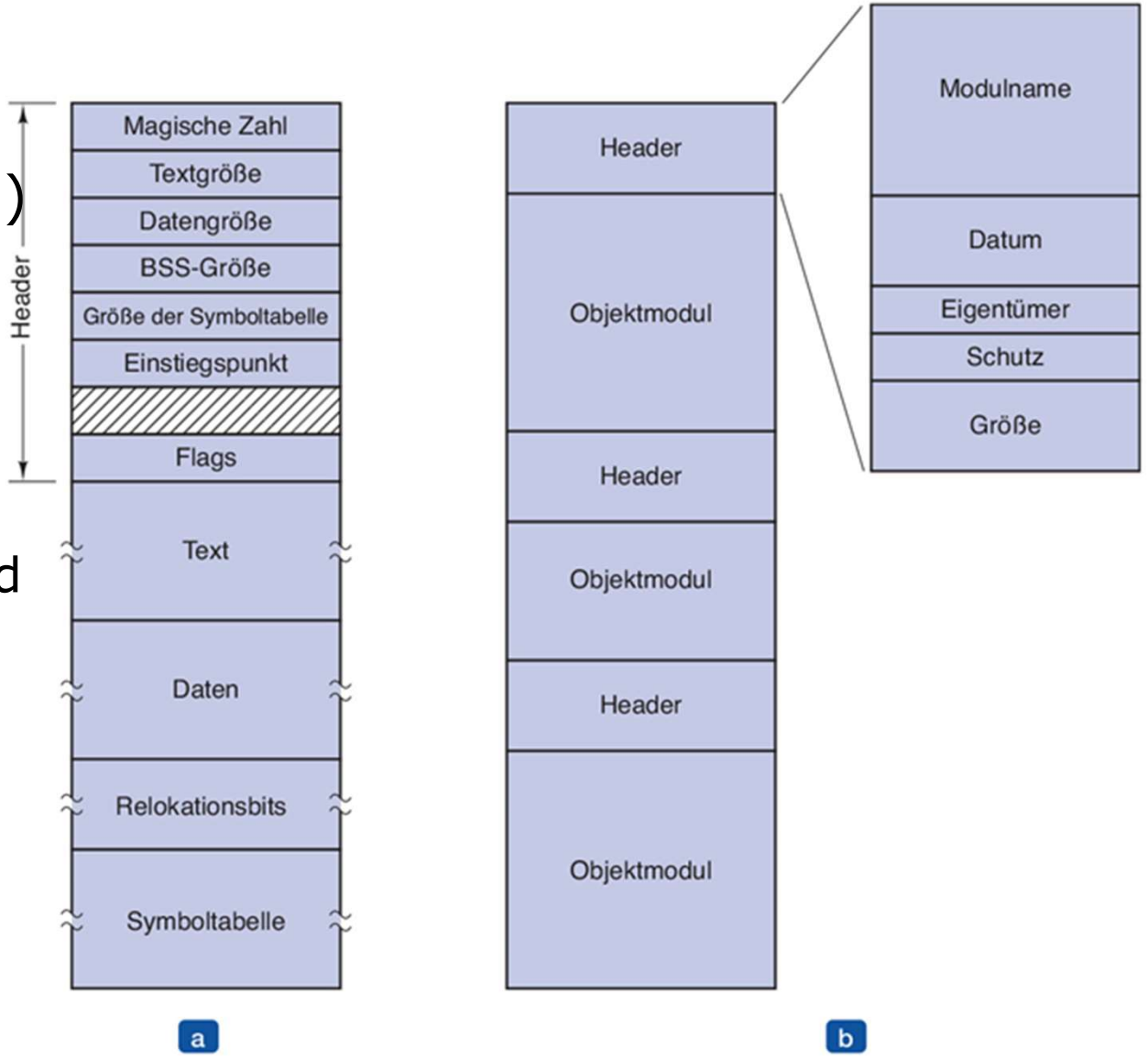


Dateitypen

- reguläre Dateien
  - Datendateien
  - ausführbare Dateien
- Verzeichnisse
  - Systemdateien zur Verwaltung von einzelnen Dateien.
- Zeichendateien
  - beziehen sich auf Ein-/Ausgabe, modellieren serielle Ein-/Ausgabegeräte, wie z.B. Terminals, Drucker und Netzwerke.
  - Zeichenweises Lesen der Daten.
  - häufig kein direkter Zugriff auf bestimmte Positionen
- Blockdateien
  - Werden verwendet, um Plattenspeicher zu modellieren
  - Blockweises Lesen der Daten
  - Positionierung auf beliebigen Block möglich

Dateitypen – Innere Struktur

- Abhängig von Dateityp
- Ausführbare Dateien (a)
  - festes Format
  - Headerinformationen
  - Nutzinformationen
- Bibliotheksarchiv (b)
  - Folge von Headern und
  - Modulen
- Erkennung über
  - Dateiendung
  - magic number
  - Prüfung



Magic Number – Magische Zahl

- kennzeichnet die Datei als ausführbar und beschreibt zudem den genauen Dateityp.
  - 0x8BADF00D: "ate bad food„
    Exception Code in iOS Crash Reports, falls Apps zu lange beim Start oder Beenden brauchen
  - 0xBEEFCACE: "beef cake"
    magische Zahl in Ressource-Dateien von Microsoft .NET
  - 0xDEADC0DE: "dead code"
    Marker in der OpenWRT-Routerfirmware
  - 0xDEADFA11: "deadfall"
    Exception Code in iOS Crash Reports bei erzwungenem Beenden von Apps
  - 0xDEFEC8ED: "defecated"
    Kennzeichnung für Core Dumps bei Open Solaris
  - 0xFEE1DEAD: "feel dead"
    Signal für einen Linux-Reboot
  - 0x0DEFACED: "defaced"
    "Gastsignatur" von Linux in Microsofts Hyper-V

Magic Number – Magische Zahl

- MS-DOS-EXE-Dateien starten mit den ASCII-Zeichen ‚MZ' (0x4D5A) bzw. selten auch ‚ZM' (0x5A4D), den Initialen des Erfinders dieses Formats, Mark Zbikowski.
- Fat Binaries auf Mac OS 9 beginnen mit der ASCII-Zeichenfolge von ‚Joy!' (engl. Freude!) (0x4A6F7921).
- Der Berkeley-Fast-File-System-Superblock wird identifiziert durch 0x19540119 oder 0x011954 je nach Version; beides ist das Geburtsdatum des Designers Marshall Kirk McKusick.
- kompilierte Java-Klassendateien (Bytecode) beginnen mit 0xCAFEBABE.

Weitere Inform.: https://www.garykessler.net/library/file_sigs.html

Dateiattribute

- Dateien verfügen über Zusatzinformationen, sog. Attribute
- Attribute werden **nicht in der Datei** sondern im **Dateisystem** gespeichert
- Beispiele
  - Eigentümer, aktueller Besitzer der Datei
  - Urheber, Erzeuger der Datei
  - Schreibschutz
  - Sichtbarkeit
  - Zugriffsrechte, für Besitzer, evtl. Gruppen, alle
  - Zeitpunkt der Erzeugung
  - Zeitpunkt der letzten Veränderung
  - Archivierungsinformationen
  - Systemdatei

Attribute

Attribute	Bedeutung
Schutz	Wer kann wie auf die Datei zugreifen?
Passwort	Passwort für den Zugriff auf die Datei
Urheber	ID der Person, die die Datei erzeugt hat
Eigentümer	Aktueller Eigentümer
Read-only-Flag	0: Lesen/Schreiben; 1: nur Lesen
Hidden-Flag	0: normal; 1: in Listen nicht sichtbar
System-Flag	0: normale Datei; 1: Systemdatei
Archiv-Flag	0: wurde gesichert; 1: muss noch gesichert werden
ASCII/Binär-Flag	0: ASCII-Datei; 1: Binärdatei
Random-Access-Flag	0: nur sequenzieller Zugriff; 1: wahlfreier Zugriff
Temporary-Flag	0: normal; 1: Datei bei Prozessende löschen
Sperr-Flags	0: nicht gesperrt; nicht null: gesperrt
Datensatzlänge	Anzahl der Bytes in einem Datensatz
Schlüsselposition	Offset des Schlüssels innerhalb des Datensatzes
Schlüssellänge	Anzahl der Bytes im Schlüsselfeld
Erstellungszeit	Datum und Zeitpunkt der Dateierstellung
Zeitpunkt des letzten Zugriffs	Datum und Zeitpunkt des letzten Zugriffs
Zeitpunkt der letzten Änderung	Datum und Zeitpunkt der letzten Dateiänderung
Aktuelle Größe	Anzahl der Bytes in der Datei
Maximale Größe	Anzahl der Bytes für maximale Größe der Datei

Metainformationen

- Dateien verfügen über weitere Zusatzinformationen, die Metainformationen
- Metainformationen werden in der Datei gespeichert und können speziell für bestimmte Dateitypen existieren
- Beispiele
  - Codierung der Datei (ASCII, binär, ...)
  - Datensatzlänge
  - Schlüssel
  - Schlüsselpositionen
  - technische Informationen, z. B. bei Bildern oder Dokumente

Dateioperationen

- Create
  - Erzeugt eine Datei ohne Inhalt. Dateierzeugung findet im Dateisystem statt, Attribute werden gesetzt: z. B. Erzeugungszeit, Dateirechte
- Delete
  - Löscht eine Datei aus dem Dateisystem. Verwendeter Plattenplatz wird wieder freigegeben.
- Open
  - Öffnet eine Datei. Es wird an den Anfang der Datei positioniert (Dateizeiger initialisiert). Daten selber werden noch nicht übertragen.
- Close
  - Schließt eine Datei. Interne Strukturen des BS im Hauptspeicher werden freigegeben. Sollte das Dateisystem blockweise arbeiten, wird der letzte, evtl. unvollständige Block mit Nullen gefüllt und auf die Platte geschrieben.

Dateioperationen

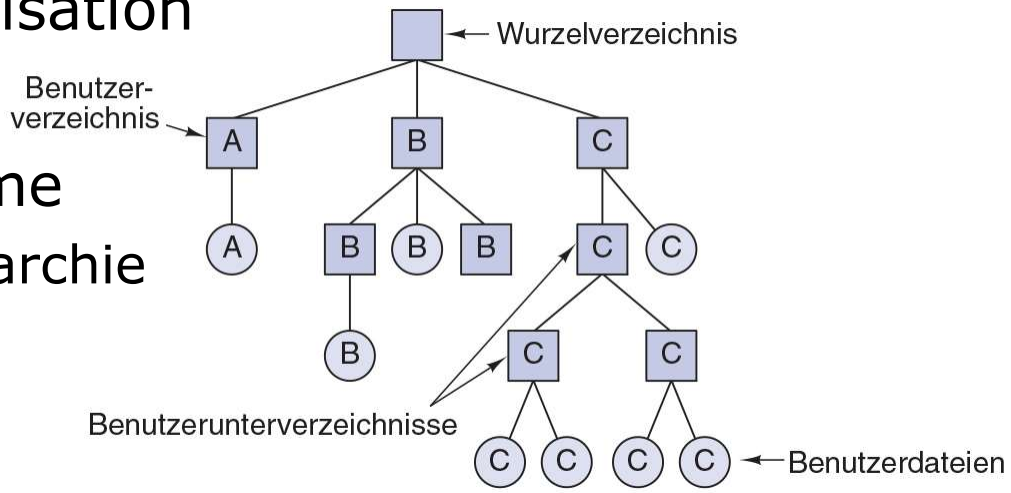
- Read
  - Daten werden aus der Datei in einen Puffer gelesen. Die Anzahl der zu lesenden Bytes muss angegeben werden. Dateizeiger wird angepasst.
- Write
  - Daten werden in die Datei an die aktuelle Position des Dateizeigers geschrieben. Zeigt der Dateizeiger auf das Ende wird die Datei vergrößert, ansonsten werden Inhalte überschrieben. Die zu schreibenden Daten befinden sich in einem Puffer.
- Append
  - Fügt Daten an das Ende der Datei an. Hierzu wird der Dateizeiger auf das Ende der Datei positioniert und dann ein write() durchgeführt.
- Seek
  - Positioniert den Dateizeiger an eine bestimmte Stelle, die die Position für weitere read()- oder write()-Befehle festlegt.

Dateioperationen

- GetAttributes
  - Liest Dateiattribute.
- SetAttributes
  - Setzt Dateiattribute.
- Rename
  - Benennt eine existierende Datei um. Alternativ kann eine Datei auch unter Verwendung eines neuen Dateinamens umkopiert werden.

Verzeichnisse

- Verzeichnisse dienen der Organisation von Dateien
- Hierarchische Verzeichnissysteme
  - Verzeichnisbaum realisiert Hierarchie
  - Hierarchie nach Aufgaben der Verzeichnisse
  - Hierarchie nach Benutzer
  - Zugriffsrechte lassen sich über Verzeichnisse allgemeiner steuern
  - Wurzelverzeichnis ist Startpunkt
  - Dateinamen müssen nur im Verzeichnis eindeutig sein
  - (fast) beliebige Struktur möglich
  - Standard bei aktuellen Betriebssystemen



Pfad

- Zugriff auf Dateien in hierarchischen Verzeichnissystemen
  - Auflistung aller Zwischenverzeichnisse
  - Sonderverzeichnisse
  - . ⇨ aktuelles Verzeichnis
  - .. ⇨ übergeordnetes Verzeichnis
  - \ ⇨ Wurzelverzeichnis (auch / oder >)
- absolute Pfade
  - beginnen immer bei \
  - eindeutig
- relative Pfade
  - beginnen immer im aktuellen Verzeichnis

Operationen auf Verzeichnisse

- Create
  - Erzeugt ein (fast) leeres Verzeichnis. Bereits enthalten sind . und ..
- Delete
  - Löscht ein leeres Verzeichnis (das nur . und .. enthält)
- Opendir
  - Öffnet ein Verzeichnis, um z. B. alle Dateien aufzulisten.
- CloseDir
  - Schließt ein geöffnetes Verzeichnis. Verwaltungsstrukturen werden dabei wieder freigegeben.

Operationen auf Verzeichnisse

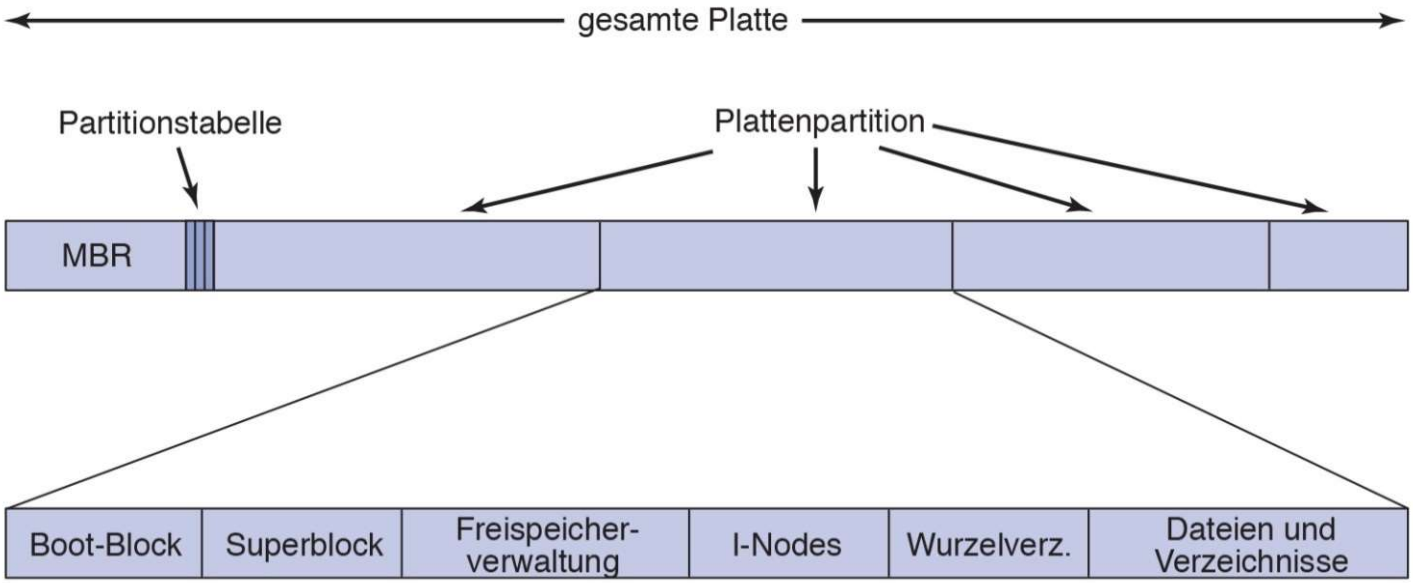
- Readdir
  - Gibt den nächsten Eintrag im Verzeichnis zurück.
- Rename
  - Benennt ein Verzeichnis um.
- Link
  - Erzeugt einen weiteren Eintrag in der Verzeichnisstruktur für eine existierende Datei.
- Unlink
  - Entfernt einen Link-Eintrag. Entfernt wird nur der Link, nicht die Datei

Verweise auf Dateien: Link

- hard link
  - Eigenständiger Verzeichniseintrag wird erzeugt.
  - Eintrag verweist auf die Datei.
  - Datei kennt die Anzahl der Verweise. Wichtig für das Löschen.
  - Original und Verweis können nicht unterschieden werden.
- soft link
  - Statt Verzeichniseintrag wird Datei erzeugt, die auf das Original verweist.
  - Keine weiteren Prüfungen durch das Betriebssystem danach.
  - Original kann gelöscht werden, link zeigt danach "ins Leere".
- Unter Windows ‚mklink‘

Partitionen

- Grundsätzlich ist eine Festplatte in Partitionen organisiert, jede Partition hat ein eigenes Dateisystem.
- MBR Partitionsschema (x86-Rechner)
  - Zu Beginn der Festplatte (Sektor 0) befindet sich der Master Boot Record (MBR), der Teile des Bootloaders und die Partitionstabelle enthält.
  - Eine Partition wird als aktiv deklariert, von ihr bootet ein Rechner.
- Beispiel für ein mögliches Layout eines Dateisystems:

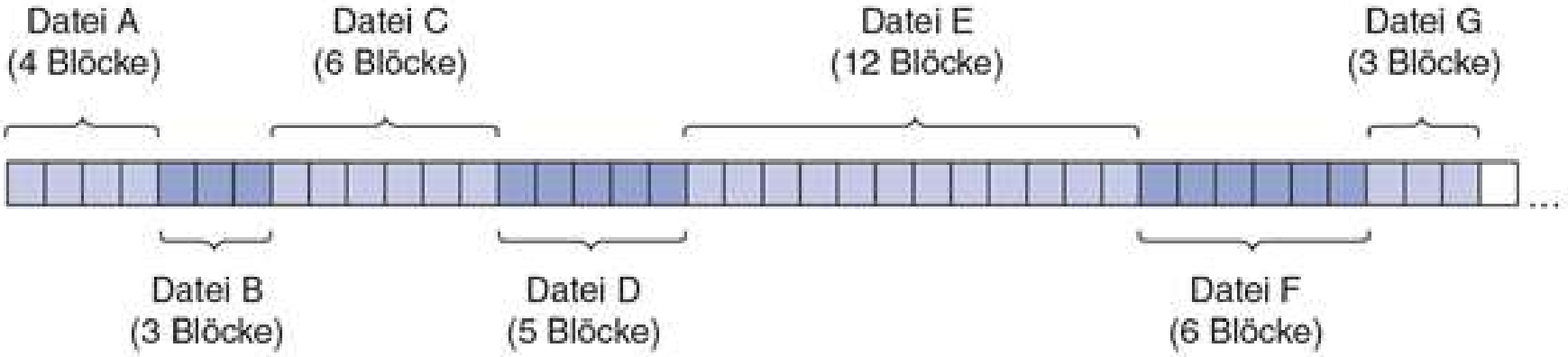


Implementierung von Dateisystemen

- Wie wird eine Datei auf einer Festplatte gespeichert, die blockorientiert ist (z.B. 4k), d.h. wie werden Blöcke einzelnen Dateien zugeordnet?
- Ansatz:
  - Zusammenhängende Belegung
  - Verkettete Listen
  - Index-Nodes („I-Nodes“)
- Kriterien
  - Möglichst wenig Fragmentierung durch Löschen und Größen-änderung von Dateien
  - Effizienz im Zugriff (wahlfreier Zugriff möglich)
  - Speichereffizienz (kompakte Verwaltungsstrukturen)

Zusammenhängende Belegung

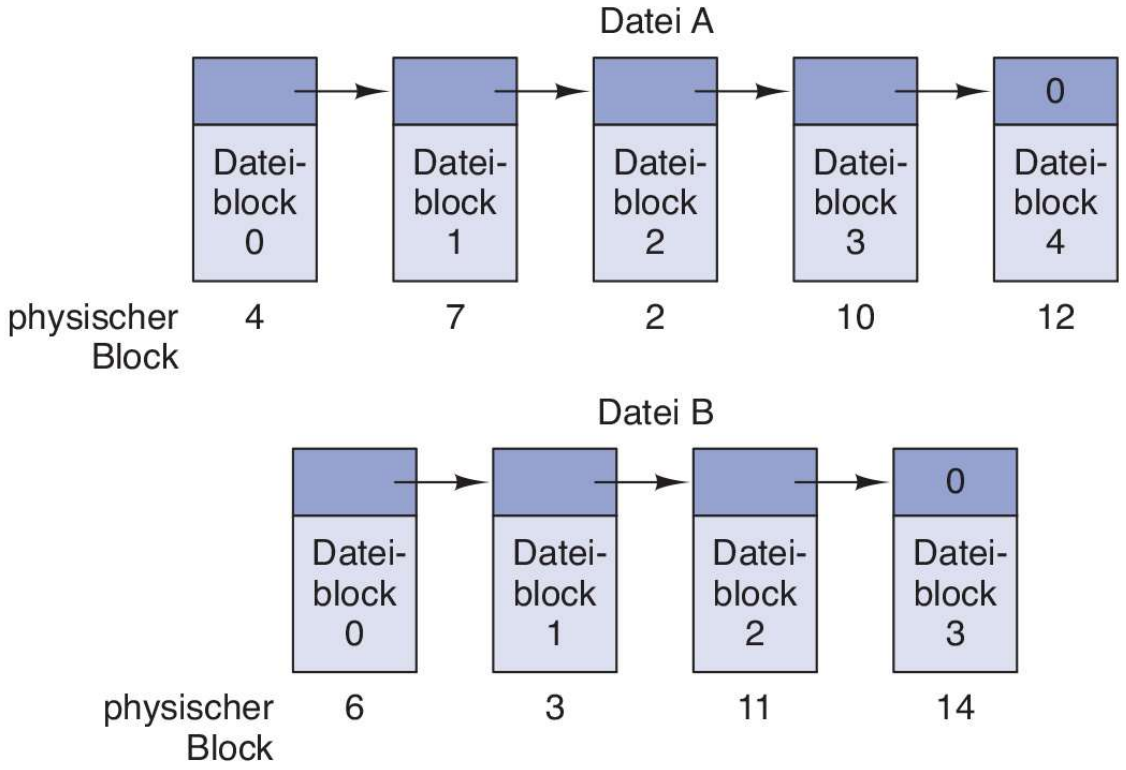
- Blockstruktur
- Dateien werden am Stück hintereinander abgelegt



- Vorteile
  - sehr einfache Implementierung
  - Datei wird vollständig durch Startblock und Länge beschrieben
  - Datei kann mit einer Leseoperation gelesen werden, das sie zusammenhängend ist
- Nachteil
  - fehlendes Verdichten bei Löschen von Dateien führt zu Fragmentierung

Verkettete Listen

- Jede Datei wird durch eine verkettete Liste von Plattenblöcken dargestellt
  - Nutzinformationen
  - Erste Wort ist Zeiger auf nächsten Datenblock

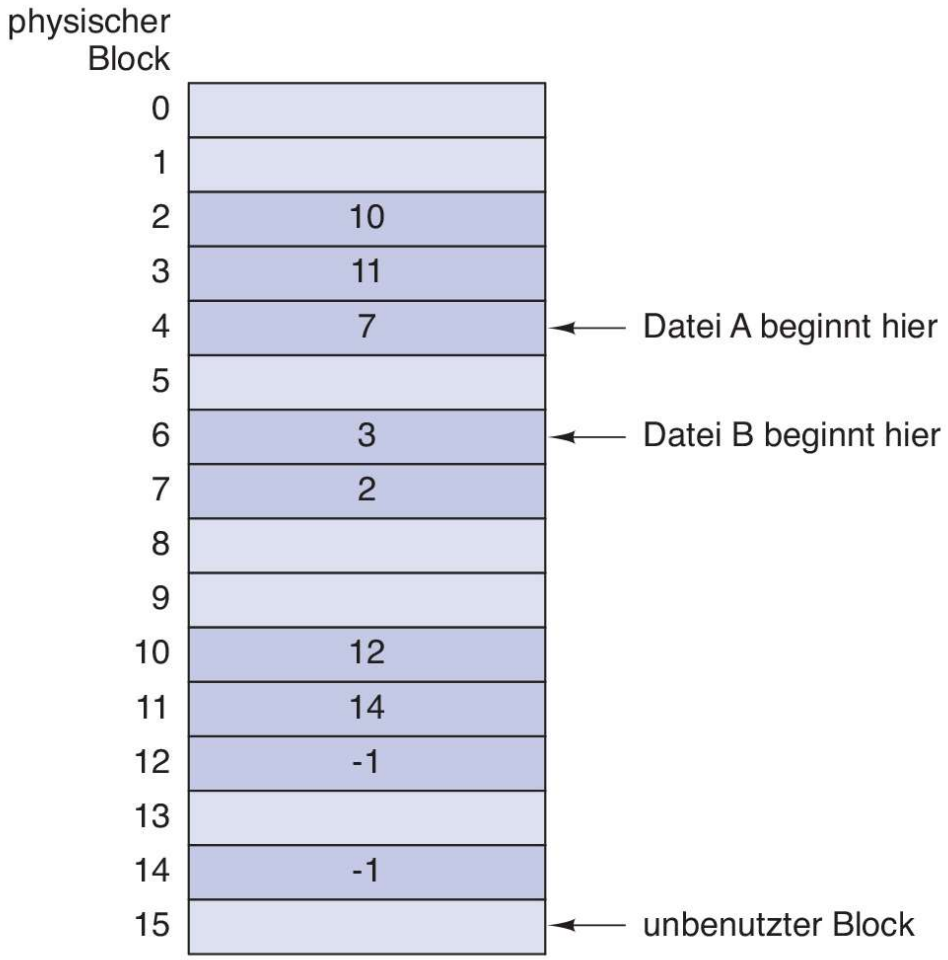


Verkettete Listen

- Vorteile
  - keine Verlust von Speicherplatz durch Fragmentierung
  - Datei wird vollständig durch ersten Eintrag beschrieben
  - sequentielles Lesen ist einfach
- Nachteile
  - wahlfreier Zugriff ist langsam
  - ganze Liste muss gelesen werden
  - auf Block muss positioniert werden
  - Platzverbrauch durch Zeiger auf nächsten Block
  - Problematisch, wenn Nutzgröße pro Block eine 2er-Potenz seien muss / soll. (nicht mehr gegeben, da Zeiger nicht zu Nutzdaten)

Verkettete Listen mit Tabelle

- Zeiger jedes Plattenblocks in Tabelle im Arbeitsspeicher
  - o Verkettung nicht in den Blöcken sondern in Tabelle
- Tabelle ⇒ FAT (File Allocation Table)
- Gesamte Block steht Daten zur Verfügung
- Kette im Arbeitsspeicher verfolgen, ohne Plattenzugriff
- Verzeichnis speichert nur Startblock



Verkettete Listen mit Tabelle

- Nachteil
  - Größe der Tabelle

Bsp.: 1TB Festplatte

- 1 KB Blockgröße ⇒ 1 Mrd. Einträge
- Mind. 3 Byte, besser 4 Byte pro Eintrag

⇒ belegt 3 oder 4 GB RAM

- weiterer Nachteil: Speichern der Tabelle in flüchtigem RAM

Index-Node / I-Node

- Jede I-Node enthält Zeiger auf 10 Datenblöcke
- Speicherbereich der 10 Zeiger kann bereits für kleine Dateien verwendet werden
- Jede Zeiger zeigt z.B. auf 1 KB großen Datenblock ⇒ 10KB
- reicht dieser Platz nicht, wird eine indirekte Adressierung verwendet
  - Zeiger zeigt auf Block mit 256 weiteren Zeigern
  - Mechanismus kann kaskadiert werden
- maximale Dateigröße: 16GB – 2TB