Betriebssysteme
10. Dateisysteme

Tobias Lauer

Aufgabe der Dateisysteme

- Adressraum eines Prozesses hält grundsätzlich alle Daten bereit, aber:
  - Größe ist durch virtuellen Adressraum begrenzt
  - Daten gehen verloren, wenn der Prozess terminiert (transiente Daten)
  - Jeder Prozess kann nur auf eigenen Prozessraum zugreifen (Ausnahme: Shared Memory)

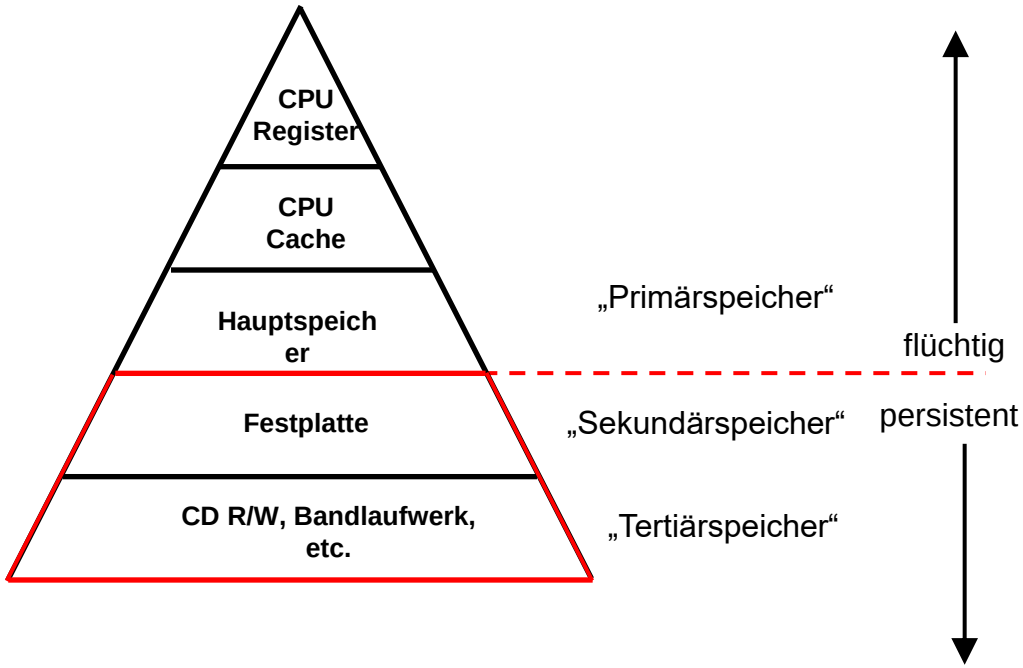
- Magnetbänder / -platten lösen diese Probleme, jedoch haben diese nur einen begrenzten Befehlssatz
  - Lesen und Schreiben von Blöcken
  - Kein direktes Auffinden der Daten
  - Kein Berechtigungskonzept

- Betriebssysteme bieten eine **Abstraktionsebene** bei der Verwaltung von Hardware
  - Prozessor ⇨ Prozesse, Threads
  - Speicher ⇨ virtueller Adressraum
  - Langzeitspeicher ⇨ **Dateisysteme**

Dateien

Eigenschaften

- **Persistenz** (sind nicht an den Lebenszyklus von Prozessen gebunden)
- Werden von Prozessen **aktiv** erzeugt / gelöscht
- Werden im Sekundär- / Tertiärspeicher abgelegt
- Bieten abstrakten Zugriff auf Daten (über Dateinamen)



Themen

- Dateien
- Verzeichnisse
- Technische Implementierung

} Anwender

} Systemprogrammierer

Grundlagen Dateien

Benennung von Dateien

- Benennungsregeln variieren zwischen Betriebssystemen
  - Unterscheidung von Groß- / Kleinschreibung
  - Zulässige Länge der Namen (Bsp. 8.3 Konvention bei MS-DOS)
  - Verwendung von Sonderzeichen
  - Bedeutung von Dateiendungen (Beispiele: .txt, .pdf, .zip)
    - Feste Bedeutung → registrierte Prozesse (Windows)
    - Konvention → rein informativ (UNIX)

Dateitypen

- Reguläre Dateien (regular files)
  - Textdateien (Binärdatei, die mit Hilfe einer Codierung (ASCII, Unicode, …) für den Menschen lesbar ist)
  - Binärdateien (kann beliebige Bytewerte enthalten)
- Verzeichnisse (directories)
  - Systemdatei, die auf weitere Dateien verweist
- Spezialdateien (UNIX)
  - Zeichendateien (character special file) → serielle Ein-/Ausgabegeräte (z.B. /dev/tty1)
  - Blockdateien (block special file) → blockorientierte Ein-/Ausgabegeräte zum Schreiben von rohen Platten (z.B. /dev/hd1)

Grundlagen Dateien (Fortsetzung)

Dateizugriffe

- Sequentieller Zugriff

  ⇨ byte-weises Lesen, am Anfang der Datei beginnend

- Wahlfreier Zugriff (random access)

  ⇨ Lesen an beliebiger Stelle (z.B. Zugriff auf Datenbanksysteme)

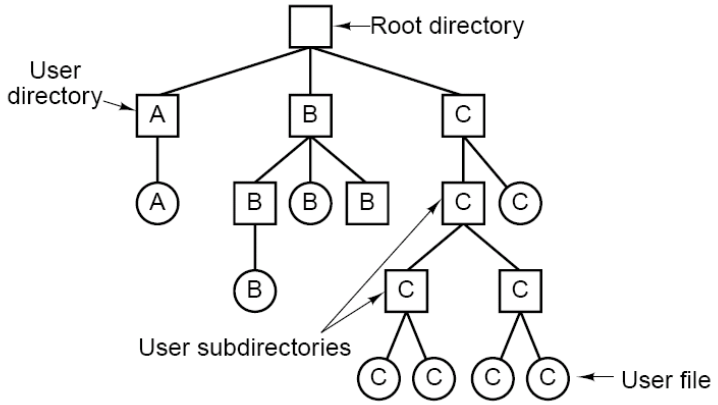
Dateiattribute

- Metadaten für
  - Dateischutz (Zugriffsrechte, …)
  - Steuerung von Dateieigenschaften (System-Flag, Random-Access Flag, ...)
  - Zeitfelder (Erstellungszeit, letzter Zugriff, …)
  - Größeninformation (Anzahl an Bytes)

Verzeichnisse

Verzeichnissysteme

- Früher nur eine Ebene: alle Dateien liegen im Wurzelverzeichnis
- Hierarchische Verzeichnissysteme
  - Hierarchischer Namensraum
  - Eindeutige Namen auf einer Hierarchiestufe
  - Lage im Verzeichnisbaum bestimmt → Pfadname



[A.S. Tanenbaum, „Moderne Betriebssysteme“]

Pfadnamen

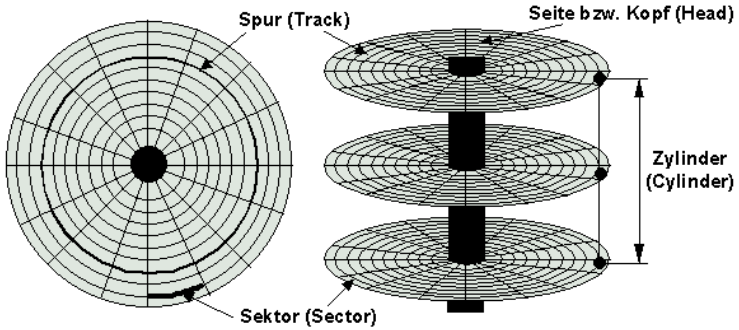
- Absoluter Pfadname (ausgehend vom Root-Verzeichnis)
- Relativer Pfadname (ausgehend vom aktuellen Verzeichnis)
- Spezielle Einträge
  - „/“, „C:\“ Root-Verzeichnisse
  - „..“ Elternverzeichnis
  - „.“ aktuelles Verzeichnis
- Links („Abkürzungen“)
  - Harte Links (Verweis auf Datei an mehreren Stellen im Verzeichnissystem)
  - Symbolische Links (Datei, die den Namen einer Datei enthält)

Technische Implementierung

Organisation Festplatte

Technisch

- Blöcke
  - Unterteilung der Festplatte in logische Einheiten
  - **kleinste adressierbare Einheit** einer Festplatte
- Plattenpartitionen
  - Verwaltungseinheit aus Sicht des Betriebssystems
  - Virtuelle Festplatten



Logisch

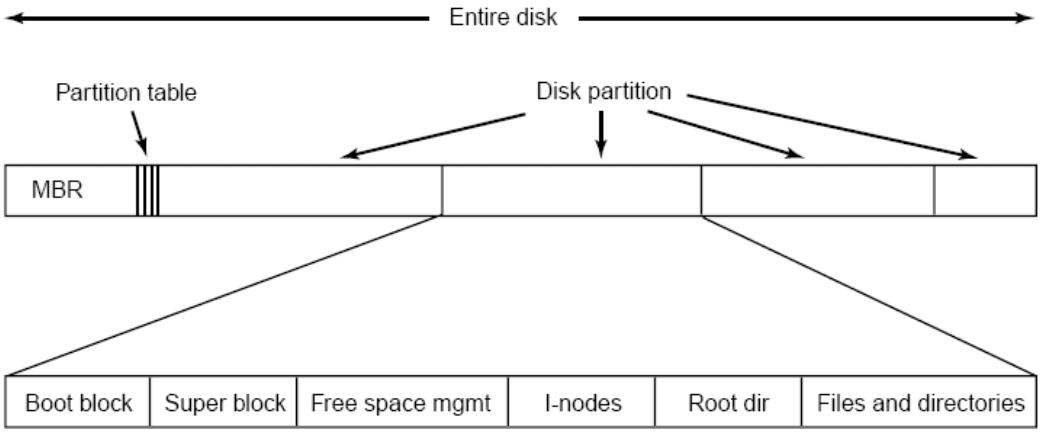
- Datei
  - Gruppierung von Daten in logischen Einheiten
  - Zugriff über Dateinamen
- Verzeichnis
  - Strukturierung von Dateien

⇨ Betriebssystem vermittelt zwischen technischer und logischer Organisation

Technische Implementierung

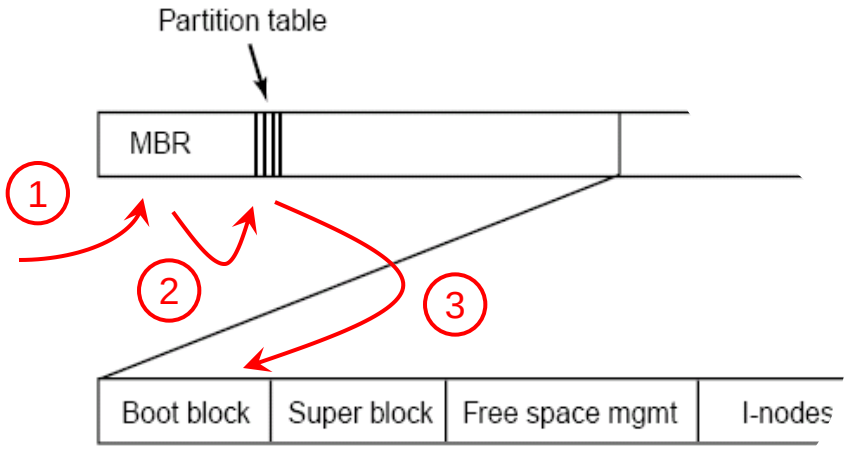
Aufteilung Festplatte

- MBR (Master Boot Record)
- Partitionstabelle (Anfangs- und Endadressen der einzelnen Partitionen)
- Typische Partitionen
  - Alle Partitionen enthalten Boot-Block
  - Super-Block (Schlüsselparameter des Dateisystems: Typ, Anzahl Blöcke, etc.)
  - Verwaltung der freien Kapazitäten
  - I-Nodes (oder File Allocation Table)
  - Wurzelverzeichnis



[A.S. Tanenbaum, „Moderne Betriebssysteme“]

Boot-Vorgang



1. BIOS (abgelegt auf Festwertspeicher) liest MBR
2. Lokalisieren der aktiven Partition
3. Lesen des ersten Blocks der aktiven Partition (Boot-Block) und laden des Betriebssystems

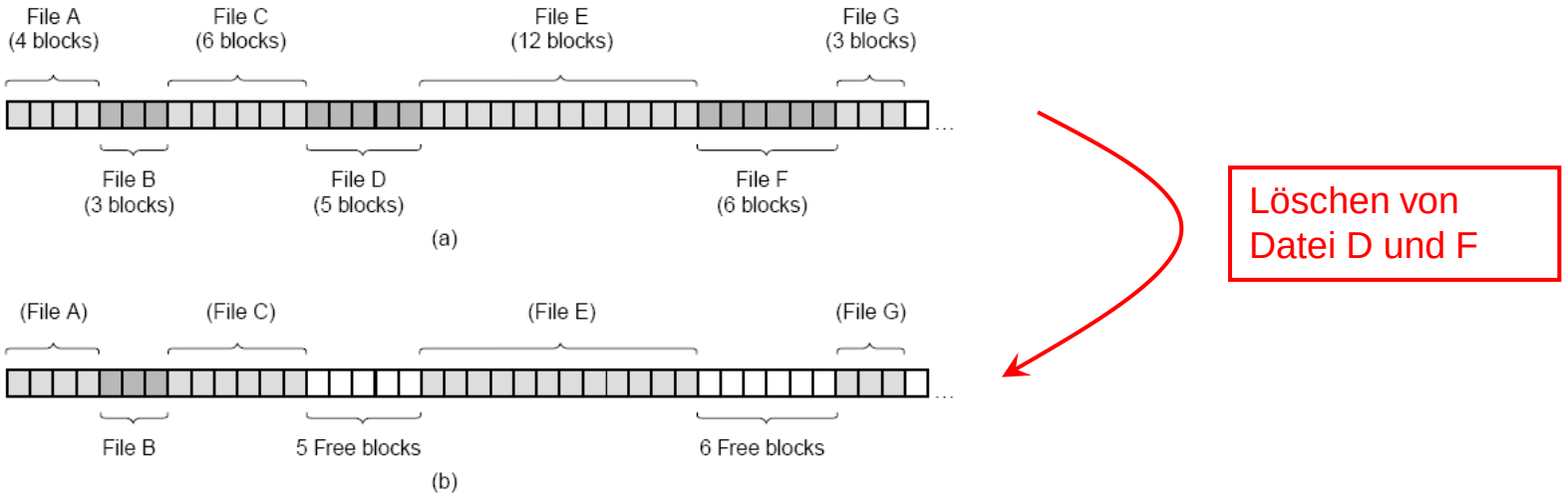
Plattenbelegung

- Aufgabe des Betriebsystems: Verteilung der Dateien auf Plattenblöcke
  - Typische Blockgröße: 512 B (bei SSDs größer: 4 KB)
  - Ggfs. Verwaltung von Clustern aus je 2^i zusammenhängenden Blöcken

- Mögliche technische Umsetzungen
  - Zusammenhängende Belegung
  - Verkettete Listen
  - Verkettete Listen mit Tabelle
  - Indexknoten (I-Nodes)

Zusammenhängende Belegung

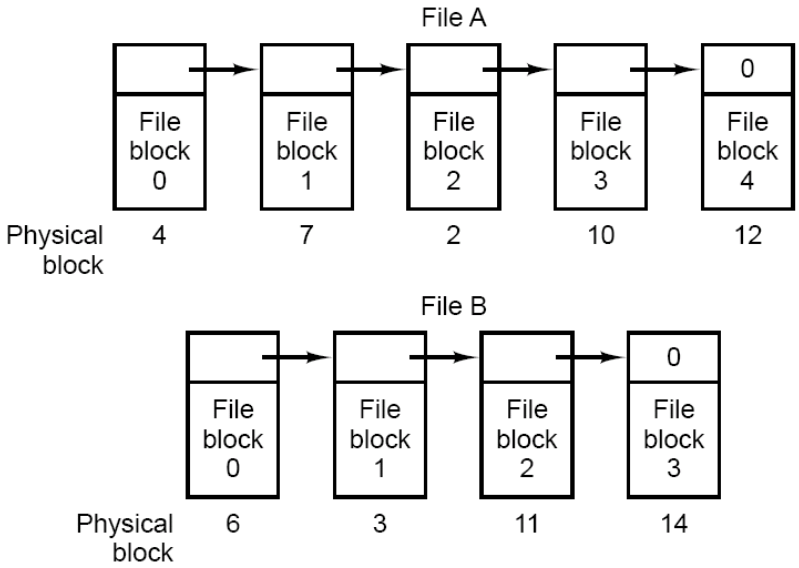
- Prinzip: Dateien werden in aufeinanderfolgenden Blöcken abgelegt



- Vorteile
  - Einfache Implementierung (Parameter: Blockadresse, Anzahl Blöcke)
  - Höchste Performanz (zusammenhängende Daten)
- Nachteile
  - Zunehmende Fragmentierung des Datenträgers durch Löschen, Hinzufügen etc.
- Verwendung bei „read-only“ Datenträgern (CD-ROM, DVD-ROM)

Verkettete Listen

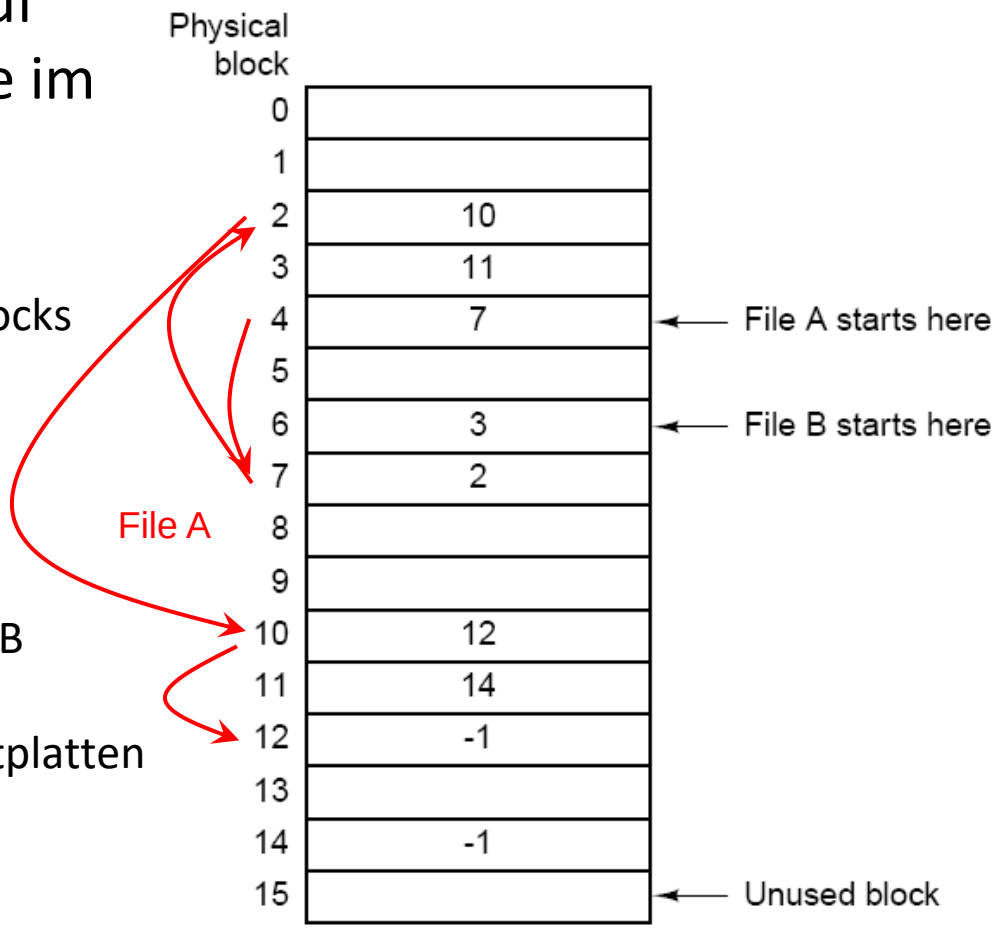
- Prinzip: Block enthält Daten und Zeiger auf den nächsten Block dieser Datei



- Vorteile
  - Kein Speicherverlust durch externe Fragmentierung
- Nachteile
  - Schlechte Performanz (v.a. für Random Access)
  - Verfügbarer Speicherplatz im Block: < 2^n Bytes (da Platz für Zeiger benötigt)

Verkettete Listen mit Tabelle

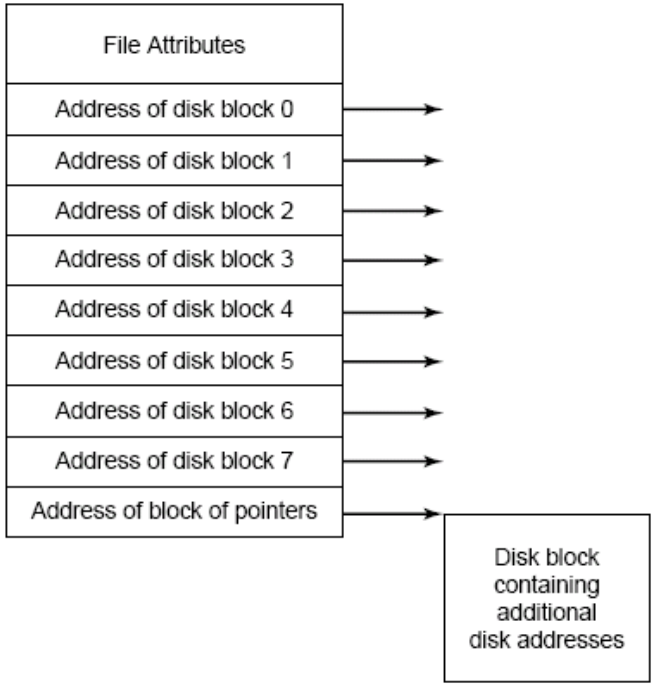
- Prinzip: Ablage des Zeigers auf den nächsten Block in Tabelle im Arbeitsspeicher
- Vorteile
  - Verfügbarer Speicherplatz des Blocks = 2^n
  - Verbesserte Performanz
- Nachteile
  - Hoher Bedarf an Arbeitsspeicher (bspw. 100 GB Festplatte ~ 800MB Arbeitsspeicher)
  - Verwendung nur für „kleine“ Festplatten (Erhöhung durch Clustering)
- Verwendung:
  FAT (File Allocation Table)



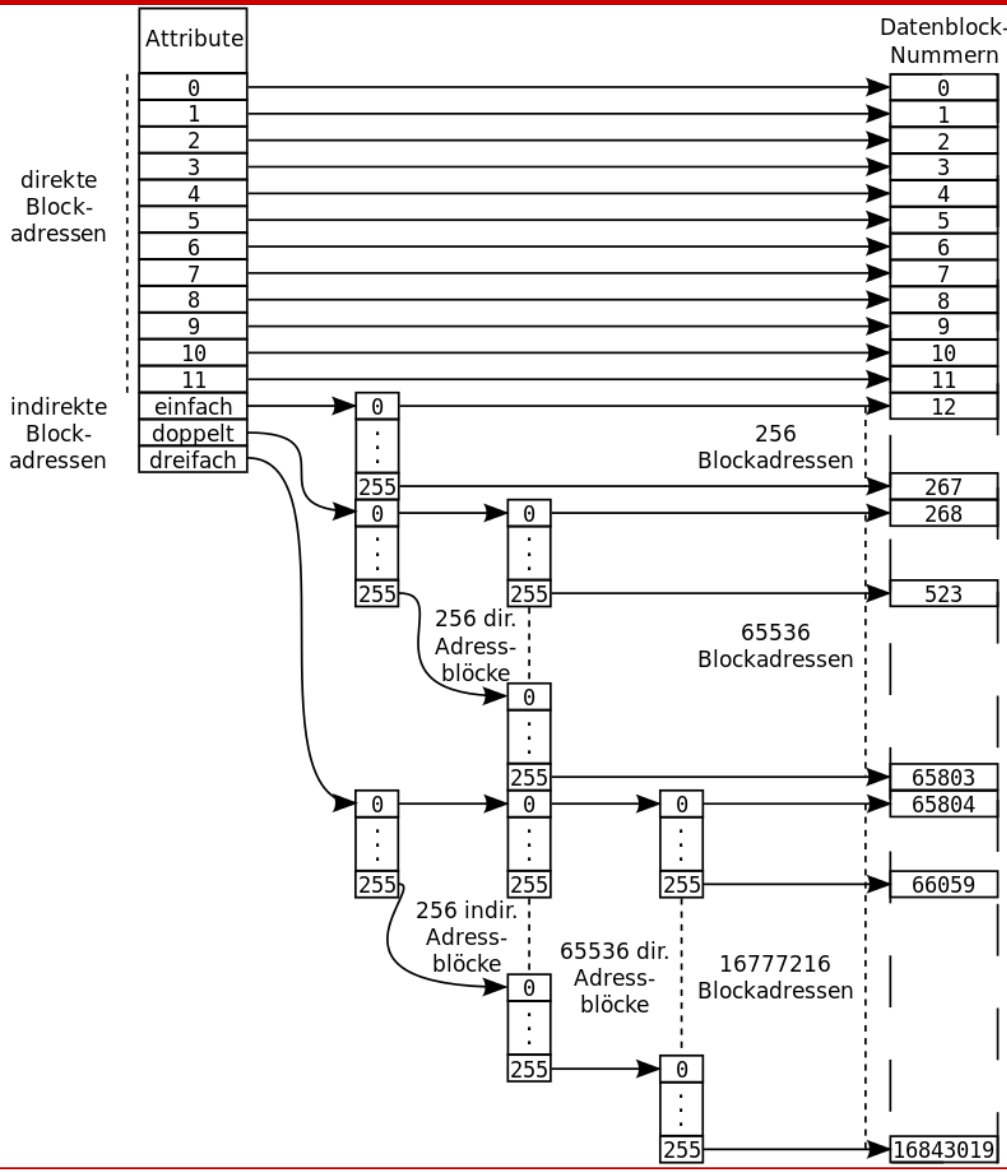
Indexknoten (I-Nodes)

- Prinzip: I-Node speichert Dateiattribute einer Datei (inkl. Verweise auf Blöcke)
- Ein I-Node pro Datei
- Vorteile
  - I-Node nur dann im Speicher, wenn Datei geöffnet ist
- Nachteile
  - Begrenzte Größe des I-Node

  ⇨ mehrstufige I-Nodes (Indirektion)
- Verwendung:
  - UNIX-Dateisystem



I-Node mit 3-fach indirekter Adressierung



Quelle: Wikipedia

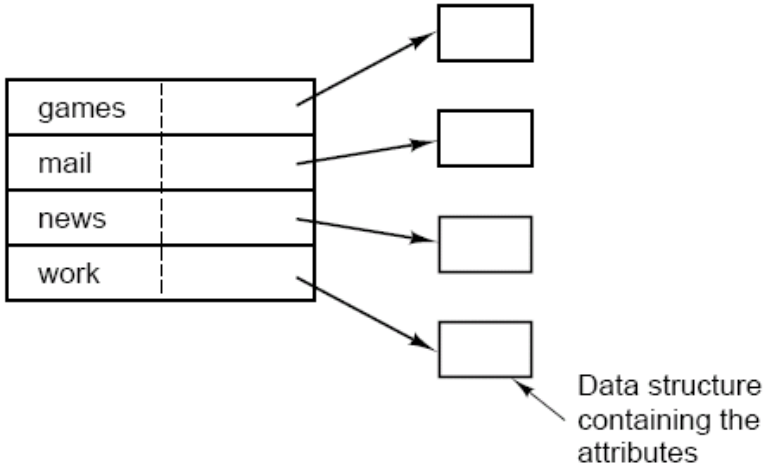
Implementierung von Verzeichnissen

- Aufgabe des Betriebssystems: Auffinden der Daten auf Plattenblöcken
- Lösung: Mapping von Pfad/Name auf Dateiattribute

  – Gemeinsame Ablage von Verzeichniseintrag und Dateiattributen
  ⇨ Windows NTFS

games	attributes
mail	attributes
news	attributes
work	attributes

  – Verzeichniseintrag enthält Verweis auf I-Nodes (I-Node-Nummer)
  ⇨ UNIX



Log-basierte Dateisysteme

- Ausgangslage: CPU immer schneller, Speicher immer größer, jedoch Festplattenzugriffszeiten verbessern sich nicht stark
- Idee: Caching im Memory
- Umsetzung:
  - Strukturierung der Platte als Log
  - Schreiben der gepufferten Schreibaufträge als Segment an das Log-Ende
  - I-Node-Map findet I-Node im Log
  - Cleaner räumt Log auf

Journaling-Dateisysteme

- Problem: viele Operationen benötigen mehrere Schritte
  Bsp: Löschen einer Datei:
  - Löschen aus Verzeichnis
  - Freigabe des I-Node
  - Freigabe der Plattenblöcke

  Was passiert bei Fehlern?
- Idee: Geplante Aktionen werden zunächst im Protokoll abgelegt und erst anschließend ausgeführt
- Im Fehlerfall kann das Protokoll erneut ausgeführt werden
- Voraussetzung: einzelne Aktionen sind idempotent
- Verbesserung durch Atomizität (alle Aktionen werden umgesetzt, oder keine)

  ⇨ Verwendung: Windows NTFS und Linux ext3

Virtuelle Dateisysteme

- Problem: Verwendung verschiedener Dateisysteme auf einem Computer
- Unterschiedliche Implementierung in Windows bzw. Linux/UNIX
- Windows: Identifikation über Laufwerksbuchstaben C:, D:, Z:
- UNIX/Linux: Integration in einen hierarchischen Verzeichnisbaum
  - Erfordert zusätzliche Abstraktionsebene: *Virtual File System* (VFS)
  - VFS bietet generische Schnittstelle für Benutzerprozesse und spezifische Schnittstelle zu konkreten Dateisystemen
  - Auch für Anbindung von entfernten Dateisystemen (*Network File System* – NFS)