

---

# Betriebssysteme I

WS 2019/2020

Roland Wismüller  
Betriebssysteme / verteilte Systeme  
roland.wismueller@uni-siegen.de  
Tel.: 0271/740-4050, Büro: H-B 8404

Stand: 23. Januar 2020



---


# Betriebssysteme I

WS 2019/2020

## 7 Ein-/Ausgabe und Dateisysteme



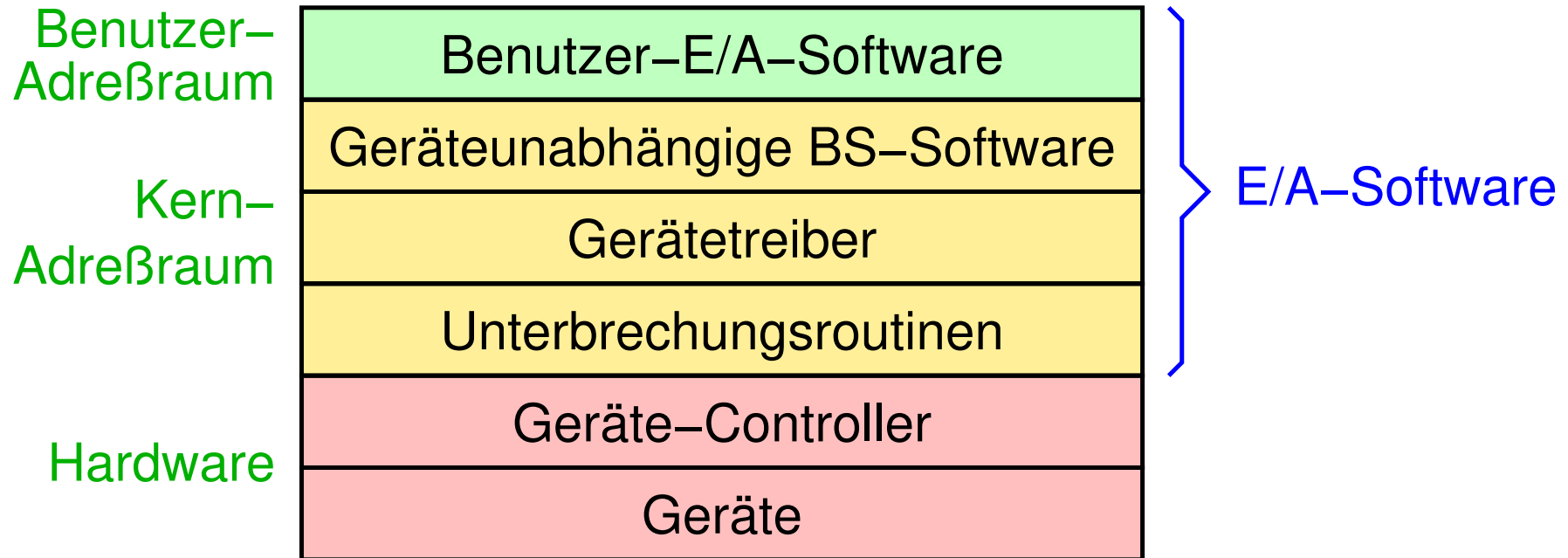
## Inhalt:

- ➔ Schichten der E/A-Software
- ➔ Ansätze zur Durchführung der E/A
- ➔ Festplatten
- ➔ (Dateiverwaltung,  **1.5.3**)
- ➔ Realisierung von Dateisystemen
  
- ➔ Tanenbaum 5.2 - 5.4, 6.1-6.3, 6.4.5
- ➔ Stallings 11.2 - 11.6, 12.1, 12.3, 12.5-12.7
- ➔ Nehmer/Sturm 10.1, 9

# 7.1 Schichten der E/A-Software



➔ Schichten bei der Ein-/Ausgabe:



➔ In der Regel für jeden Gerätetyp eigene Controller und Gerätetreiber



## Gerätetreiber

- ➔ Geräteabhängiger Teil der E/A-Software
  - ➔ kommuniziert direkt mit dem jeweiligen Geräte-Controller
- ➔ Heute i.d.R. in Kern-Adreßraum eingebunden
  - ➔ beim Hochfahren des BSs oder zur Laufzeit
  - ➔ Problem: fehlerhafte Treiber können zu BS-Abstürzen führen
  - ➔ Lösungsmöglichkeiten:
    - ➔ zertifizierte Treiber
    - ➔ Treiber als Systemprozesse im Benutzeradreßraum
    - ➔ Universalgeräte mit universellen Treibern
- ➔ Einheitliche Schnittstelle zwischen Treibern und BS
  - ➔ nur Unterscheidung block-/zeichenorientierte Geräte



## Geräteunabhängige E/A-Software

- ➔ Aufgaben:
  - ➔ Einheitliche Schnittstelle für Gerätetreiber
    - ➔ Namensgebung für Geräte, Zuordnung zu Treibern
    - ➔ Zugriffsschutz
  - ➔ Pufferung von Daten
  - ➔ Fehlerbehandlung
  - ➔ Anforderung und Freigabe von Geräten
    - ➔ für exklusive Nutzung, z.B. CD-Brenner
  - ➔ Verdeckung von Unterschieden der Geräte
    - ➔ z.B. unterschiedliche Blockgrößen



### Geräteunabhängige E/A-Software ...

- ➔ Bereitgestellte Grundfunktionen:
  - ➔ Öffnen eines Geräts
    - ➔ Argumente: Gerätename, Betriebsparameter
    - ➔ Ergebnis: *Handle* zum Zugriff auf das Gerät
  - ➔ Schließen des Geräts
  - ➔ Lesen / Schreiben von Daten(blöcken)
- ➔ In UNIX:
  - ➔ Geräte sind in das Dateisystem abgebildet
    - ➔ z.B. /dev/mouse, /dev/hda
  - ➔ Zugriff auf Geräte über normale Dateioperationen



## Benutzer-E/A-Software

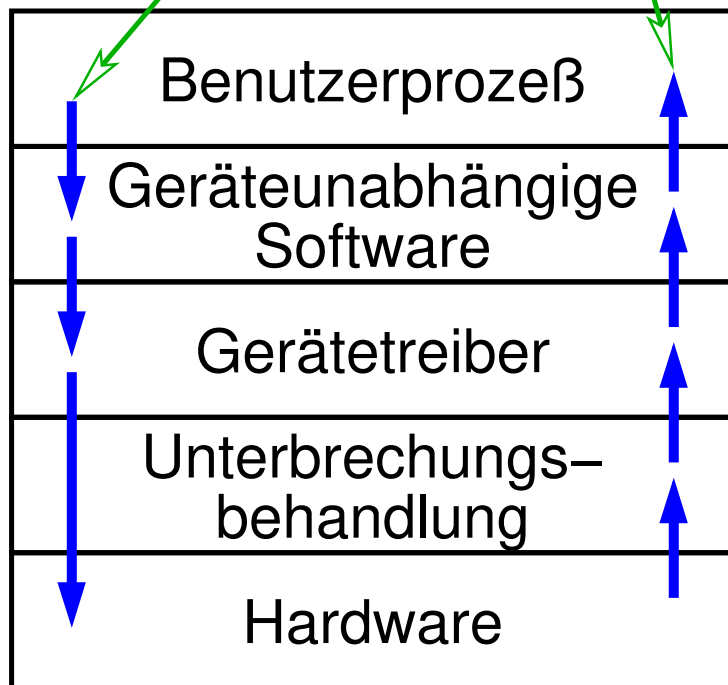
- ➔ Bibliotheksfunktionen
  - ➔ z.B. zur formatierten Ein-/Ausgabe von Zeichenketten
- ➔ Spooling-System
  - ➔ Hintergrundprozesse nehmen Auftrag entgegen und führen eigentliche E/A durch
  - ➔ E/A-Geräte müssen nicht mehr exklusiv zugeteilt werden
  - ➔ Beispiele: Drucker, Mail-System





## Zusammenfassung: Kontrollfluß im E/A-System

E/A-Anforderung    E/A-Antwort



### E/A-Funktionen:

E/A-Aufruf, Formatierung, Spooling

Benennung, Schutz, Puffern, Belegen

Gerätregister schreiben/lesen, Status prüfen

Treiber aktivieren, wenn E/A beendet

E/A-Operation durchführen



### Programmierte E/A

➔ Gerätetreiber wartet nach einem Auftrag an den Controller aktiv (in einer Warteschleife) auf das Ergebnis

➔ **aktives Warten** (*busy waiting*)

➔ Beispiel: Treiber-Code zur Ausgabe auf einen Drucker

Puffer aus Benutzer-Adreßraum in Kern-Adreßraum kopieren;

// buf = Puffer im Kern, count = Länge

```
for (i=0; i<count; i++) {  
    while (printer.status != READY); // aktives Warten!  
    printer.data = buf[i];           // ein Zeichen ausgeben  
}
```

scheduler(); // Rückkehr in Benutzermodus

➔ Nachteil: ineffizient

➔ CPU könnte der Zwischenzeit andere Aufgaben erfüllen



### Interrupt-gesteuerte E/A

- ➔ Treiber beauftragt den Controller und kehrt sofort zurück
  - ➔ auftraggebender Thread wird ggf. blockiert
- ➔ Controller sendet Interrupt an CPU, wenn der Auftrag erledigt ist
  - ➔ d.h. Gerät ist wieder bereit
  - ➔ i.d.R.: Interrupt-Nummer identifiziert das Gerät
- ➔ Treiber behandelt die Unterbrechung
  - ➔ auftraggebender Thread ggf. wieder rechnend gesetzt
- ➔ Sinnvoll bei langsamen E/A-Geräten



### Interrupt-gesteuerte E/A ...

➔ Beispiel: Ausgabe auf Drucker

#### Treiber-Code

```
Benutzer-Puffer kopieren;  
// ==> buf, count  
while (printer.status  
        != READY);  
printer.data = buf[0];  
i = 1;  
aktuellen Thread T blockieren;  
scheduler();
```

#### Interrupt-Handler (im Treiber)

```
if (i == count) {  
    Thread T bereit setzen;  
} else {  
    printer.data = buf[i];  
    i = i+1;  
}  
Interrupt bestätigen;  
Rückkehr aus Interrupt-Routine;
```



### Direkter Speicherzugriff (*Direct Memory Access, DMA*)

- ➔ Transport der Daten zwischen Controller und Speicher erfolgt nicht durch die CPU, sondern durch separate Hardware (**DMA-Controller**)
  - ➔ oft auch in Geräte-Controller integriert
- ➔ Treiber sendet Geräte-Adresse, Startadresse und Länge der Daten, sowie Transferrichtung an DMA-Controller
- ➔ DMA-Controller erzeugt Interrupt als Fertigmeldung
- ➔ Vorteile: Entlastung der CPU, Einsparung von Interrupts
- ➔ Sinnvoll (nur) bei Übertragung größerer Datenblöcke
  - ➔ DMA-Controller arbeitet nebenläufig mit CPU
  - ➔ DMA-Controller aber oft langsamer als CPU



### Direkter Speicherzugriff (*Direct Memory Access, DMA*) ...

➔ Beispiel: Ausgabe auf Drucker

#### Treiber-Code

evtl.: Benutzer-Puffer kopieren;

// ==> buf, count

DMA-Controller aufsetzen

// Parameter: buf, count, printer

aktuellen Thread T blockieren;

**scheduler ();**

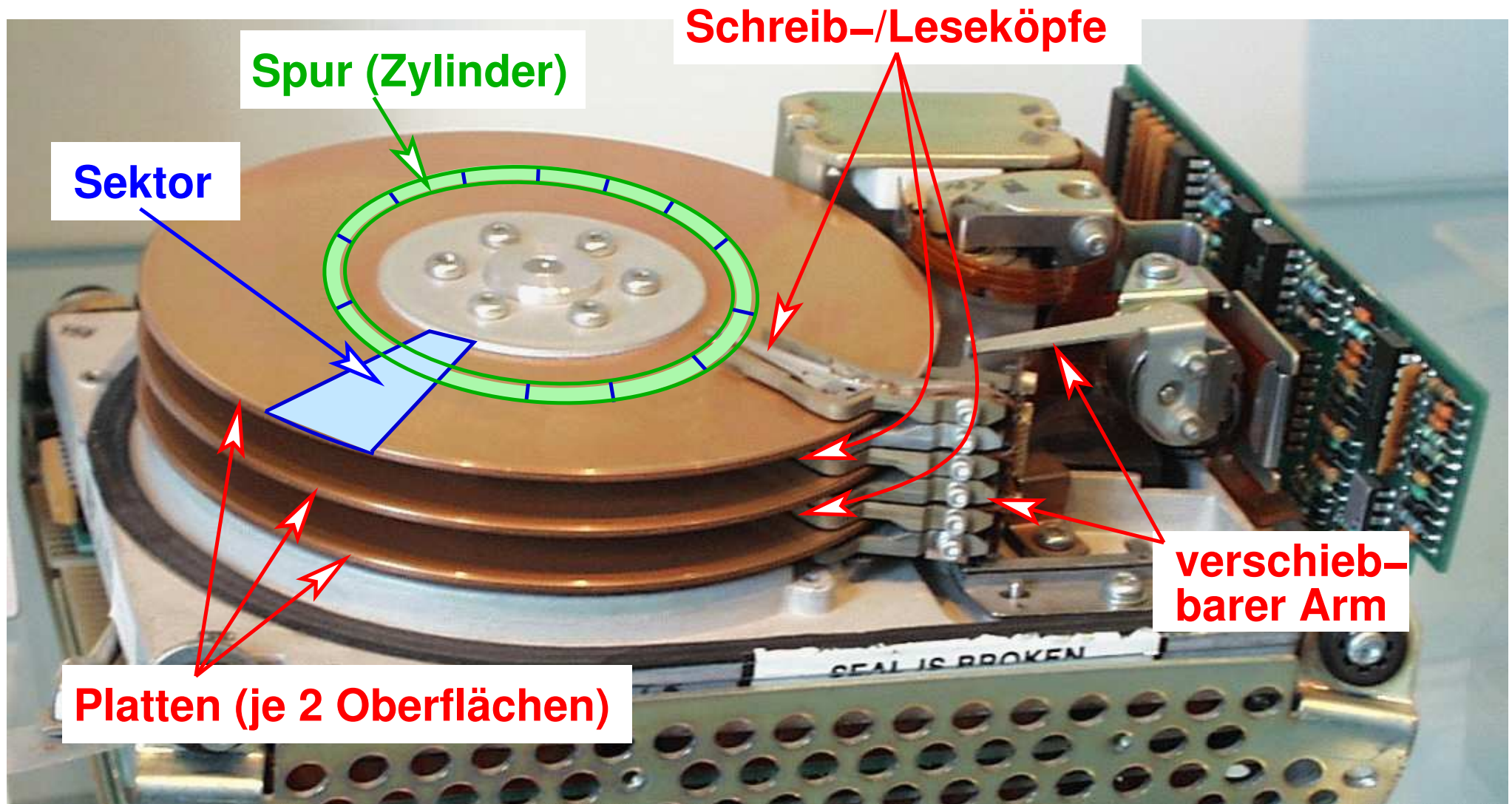
#### Interrupt-Handler (im Treiber)

Thread T bereit setzen;

Interrupt bestätigen;

Rückkehr aus Interrupt-Routine;

### Aufbau einer Festplatte





### Aufbau einer Festplatte ...

- ➔ Spurwechsel durch Verschieben des Arms (d.h. aller Köpfe)
  - ➔ Menge der jeweils übereinanderliegenden Spuren: Zylinder
- ➔ Einteilung der Festplatte (Adressierung):
  - ➔ Oberfläche (Kopf), Zylinder, Sektor
    - ➔ Sektor einer Spur nimmt einen Datenblock auf (meist 512 Byte)
  - ➔ Beispiel: physische Geometrie einer 18.3 GByte Festplatte
    - ➔ 12 Oberflächen, 10601 Zylinder, ca. 281 Sektoren
- ➔ Seit längerem: äußere Spuren besitzen mehr Sektoren als innere
  - ➔ früher: Controller zeigte dem BS eine virtuelle Geometrie an
  - ➔ heute: lineare Adressierung der Blöcke (LBA)

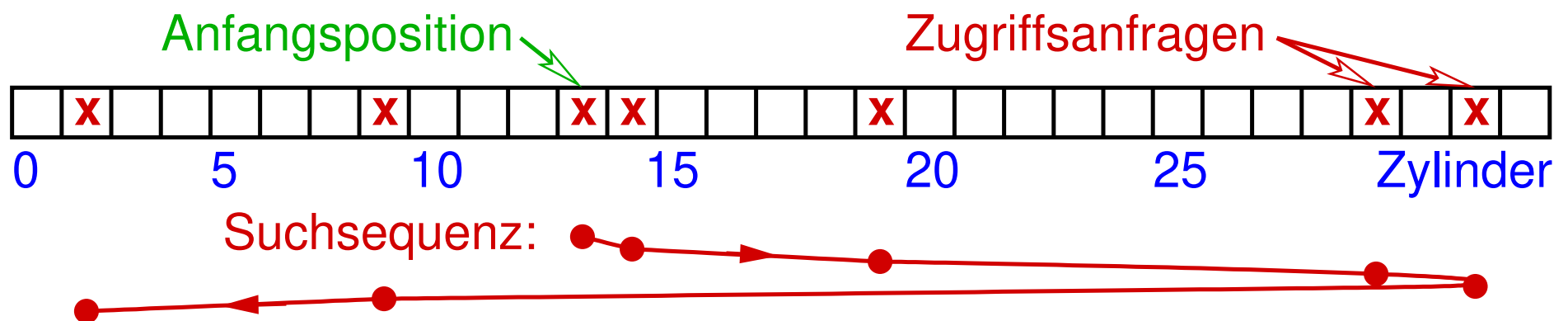


### Zugriffszeit einer Festplatte

- ➔ Drei bestimmende Faktoren:
  - ➔ Suchzeit (Anfahren der gewünschten Spur)
    - ➔ im Durchschnitt ca. 5 - 10  $ms$
  - ➔ Rotationsverzögerung (bis Sektor unter dem Kopf ist)
    - ➔ im Durchschnitt ca. 2 - 6  $ms$  (5400 - 15000 U/min)
  - ➔ Dauer der Datenübertragung
    - ➔ ca. 5 - 100  $\mu s$  pro Block
- ➔ Zugriffszeit dominiert durch Suchzeit, daher:
  - ➔ Dateien möglichst in aufeinanderfolgenden Sektoren
    - ➔ meist auch: *Prefetching* und *Caching*
  - ➔ geeignetes Scheduling der Plattenzugriffe

### Scheduling von Plattenzugriffen

- ➔ FCFS: viele unnötige Bewegungen des Plattenarms
- ➔ SSF (*Shortest Seek First*)
  - ➔ Zugriffe in der Nähe der aktuellen Position bevorzugen
  - ➔ Problem: Unfairness, Verhungerung möglich
- ➔ Aufzug-Algorithmus
  - ➔ erst in eine Richtung, bis es in dieser Richtung keine Anfragen mehr gibt; dann Richtung wechseln



---

# Betriebssysteme I

**WS 2019/2020**

23.01.2020

Roland Wismüller  
Betriebssysteme / verteilte Systeme  
roland.wismueller@uni-siegen.de  
Tel.: 0271/740-4050, Büro: H-B 8404

Stand: 23. Januar 2020



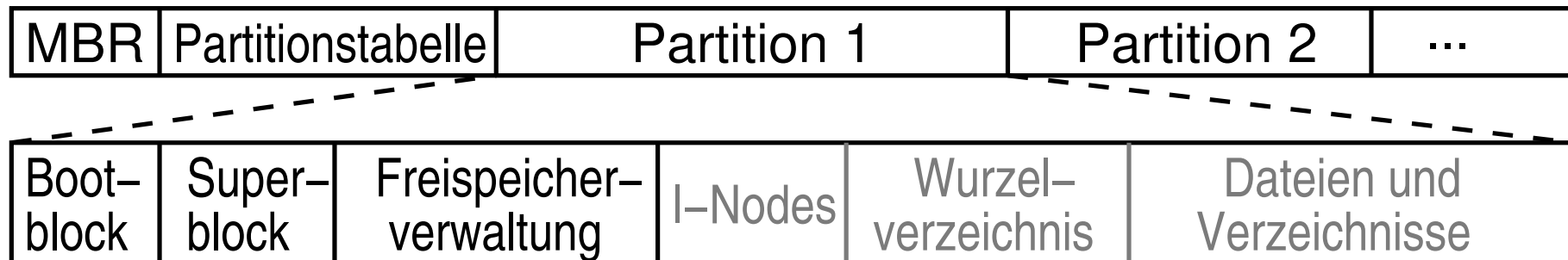
### Schichtenmodell

- ➔ Datenträgerorganisation
  - ➔ einheitliche Schnittstelle zu allen Datenträgern
  - ➔ Datenträger als Folge von Blöcken betrachtet
- ➔ Blockorientiertes Dateisystem
  - ➔ Realisierung von Dateien
- ➔ Dateiverwaltung
  - ➔ Dateinamen und Verzeichnisse



### Datenträgerorganisation

- ➔ Evtl. Einteilung des Datenträgers in Partitionen
- ➔ Partition wird als Folge von (logischen) Blöcken betrachtet
  - ➔ Blöcke fortlaufend nummeriert
- ➔ Typisches Layout einer Festplatte (UNIX):



- ➔ MBR: *Master Boot Record*
- ➔ Superblock: Verwaltungsinformation der Partition
  - ➔ Größe, Blockgröße, ...



### Datenträgerorganisation: Aufgaben

- ➔ Formatieren des Datenträgers
- ➔ Lesen / Schreiben von Blöcken
- ➔ Verwaltung freier Blöcke
  - ➔ vgl. dynamische Speicherverwaltung!
  - ➔ meist: Bitvektoren statt Freispeicherliste
    - ➔ Bit  $i$  gesetzt  $\Leftrightarrow$  Block  $i$  belegt
    - ➔ Bitvektor wird auf Datenträger gespeichert
- ➔ Verwaltung defekter Blöcke
  - ➔ ebenfalls über Bitvektoren

### Blockorientiertes Dateisystem

➔ Datei als Folge von Blöcken realisiert

➔ Zuteilung von Blöcken an Dateien:

➔ zusammenhängende Belegung



➔ Datei durch Anfangsblock und Blockanzahl beschrieben

➔ sehr gute Performance beim Lesen

➔ Problem: Speicherverwaltung (vgl. 6.2)

➔ Anfügen an Dateien, Fragmentierung, ...

➔ verteilte Belegung

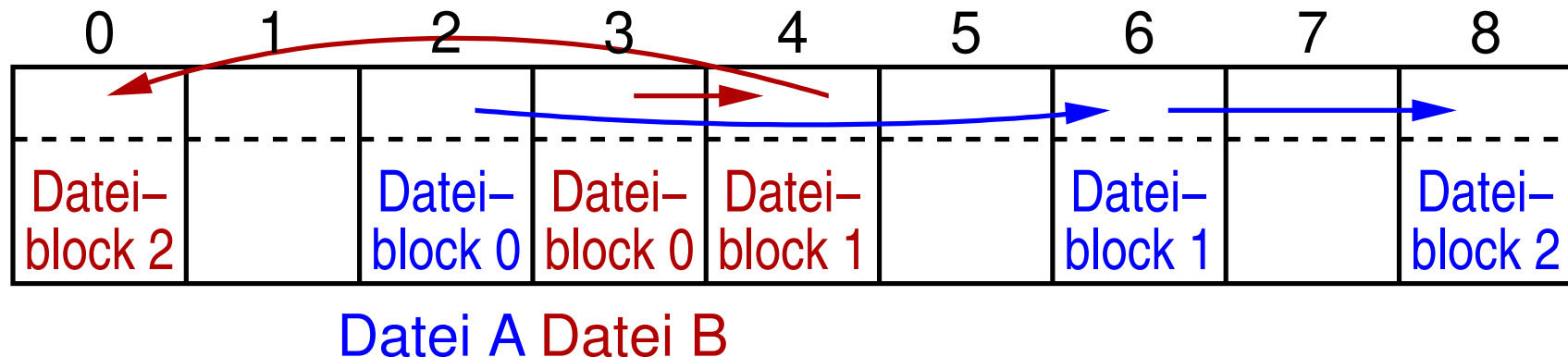


➔ einfache Speicherverwaltung, schlechtere Performance

➔ Praxis: Belegung möglichst zusammenhängend

### Blockorientiertes Dateisystem: verteilte Belegung

- ➔ Realisierung durch verkettete Listen
- ➔ Verkettung innerhalb der Blöcke



- ➔ Nachteile: wahlfreier Zugriff ineffizient, Dateiblock-Länge keine Zweierpotenz mehr
- ➔ Realisierung durch externe Tabelle (*File Allocation Table*, FAT)
- ➔ Verkettung ausserhalb der Blöcke
- ➔ Tabelle kann (teilweise) im Hauptspeicher gehalten werden



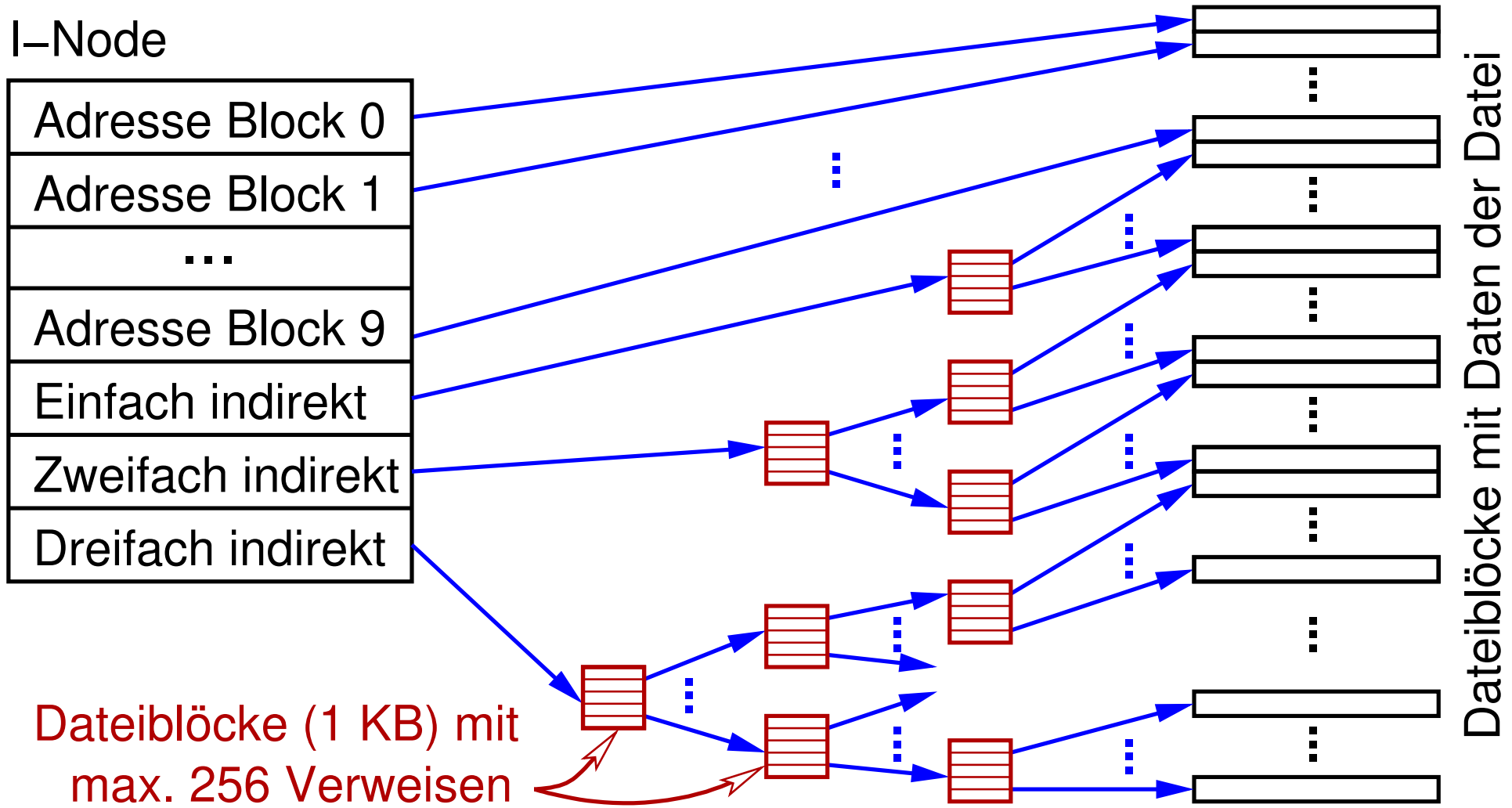


### Blockorientiertes Dateisystem: verteilte Belegung ...

- ➔ Realisierung durch Index-Knoten (*I-Nodes*)
  - ➔ jeder Datei ist eine Datenstruktur (*I-Node*) zugeordnet
  - ➔ *I-Node* enthält Tabelle mit Verweisen auf Dateiblöcke
    - ➔ wegen Speicherplatzbedarf: Tabelle ggf. mehrstufig
  - ➔ Tabelle kann im Hauptspeicher gehalten werden
  - ➔ schneller wahlfreier Zugriff auf Datei möglich
  - ➔ Beispiel: *I-Nodes* in UNIX
  - ➔ ähnliches Konzept auch in NTFS



## Blockorientiertes Dateisystem: *I-Nodes* in UNIX





### Dateiverwaltung

- ➔ Aufgabe: Realisierung von Verzeichnissen
- ➔ Verzeichnis enthält für jede Datei:
  - ➔ Dateiname, Plattenadresse (erster Dateiblock, *I-Node*), Dateiattribute
- ➔ Anmerkung: Bei Verwendung von *I-Nodes* werden die Dateiattribute im *I-Node* gespeichert
- ➔ Dateiattribute (u.a.):
  - ➔ Eigentümer, Schutzinformation (Zugriffsrechte), ...
  - ➔ Dateityp, Sperre, archiviert, ...
  - ➔ Erstellungszeit, Zeit der letzten Änderung, ...



- ➔ Schichten der E/A-Software
  - ➔ Benutzer-E/A-Software, geräteunabhängige BS-Software, Gerätetreiber, Unterbrechungsrouinen
  - ➔ Treiber: geräteabhängig, zur Laufzeit in BS-Kern geladen
- ➔ Durchführung der E/A: programmierte E/A, Interrupts, DMA
- ➔ Festplatten: eingeteilt in Oberfläche, Zylinder, Sektor
- ➔ Aufbau von Dateisystemen: Schichtenmodell
  - ➔ Datenträgerorganisation
    - ➔ Freispeicherverwaltung
  - ➔ Blockorientiertes Dateisystem (Datei = Menge von Blöcken)
    - ➔ zusammenhängende / verteilte Belegung von Blöcken
  - ➔ Dateiverwaltung
    - ➔ Verzeichnis: Name, Adresse, Attribute für jede Datei