

Systemnahe Programmierung in C

25 Dateisysteme – Einleitung

J. Kleinöder, D. Lohmann, V. Sieh

Lehrstuhl für Informatik 4
Systemsoftware

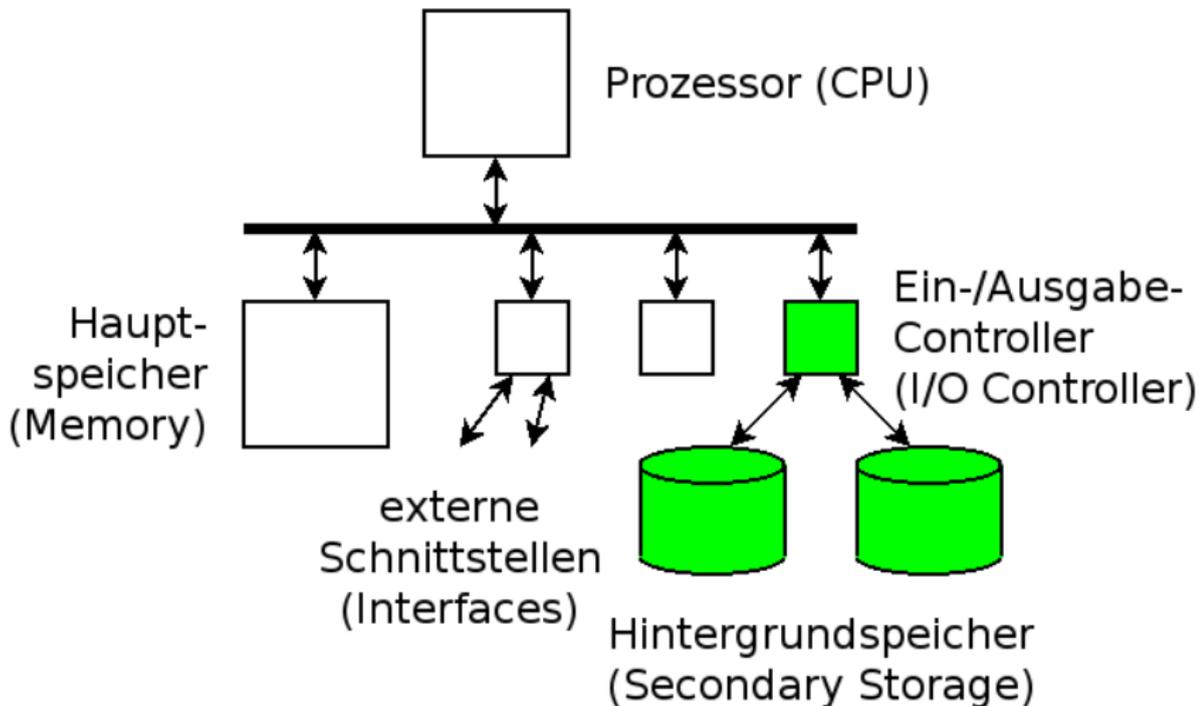
Friedrich-Alexander-Universität
Erlangen-Nürnberg

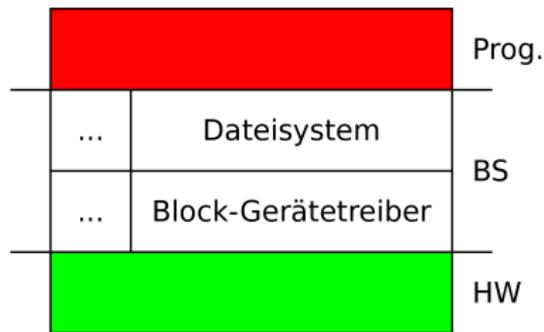
Sommersemester 2025

<http://sys.cs.fau.de/lehre/ss25>



■ Einordnung





Anwendungsprogramm:

- liest/schreibt Dateiinhalte
- liest/schreibt Verzeichnisinhalte

Dateisystem:

- liest/schreibt Blöcke

Block-Gerätetreiber:

- liest/schreibt I/O-Register

Hardware:

- liest/schreibt Bytes von/auf Datenträger



- Speichermedien (z.B. Platten, SSD / Flash-Speicher, DVD, CD-ROM) mit Unterschieden; Beispiele
 - Blockgrößen:
 - Festplatten: 512 Bytes/Block
 - CDs: 2048 Bytes/Block
 - Flash: 4096 Bytes/Block
 - Nutzung der Blöcke
 - Flash-Speicher hat nur begrenzte Anzahl von Schreibzyklen pro Block => gleichmäßig beschreiben
 - Festplatten können auf benachbarte Blöcke jeweils schneller zugreifen
 - Größe der Medien (typ.)
 - CD-ROM: ca. 750 MByte
 - DVD: ca. 8,5 GByte
 - Festplatte: ca. 4 TByte
 - SSD: ca. 500 GByte



Beispiel: PC-IDE-Festplatten-Treiber (vereinfacht):

```
void block_read(uint32_t nr, uint8_t buf[]) {
    /* Read 1 data block. */
    IDE_COUNT = 1;

    /* Set block number. */
    IDE_BLK0 = (nr >> 0) & 0xff;
    IDE_BLK1 = (nr >> 8) & 0xff;
    IDE_BLK2 = (nr >> 16) & 0xff;
    IDE_BLK3 = (nr >> 24) & 0xff;

    /* Send command. */
    IDE_CMD = IDE_READ;

    /* Wait for READY bit set. */
    while (! (IDE_STATUS & IDE_READY)) { /* Wait... */ }

    /* Read data. */
    for (i = 0; i < 512; i++) {
        buf[i] = IDE_DATA;
    }
}
```



- Dateisysteme speichern Daten und Programme persistent in Dateien
 - Benutzer muss sich nicht um die Ansteuerung und Verwaltung verschiedener Speichermedien kümmern
 - einheitliche Sicht auf den Hintergrundspeicher
- Wesentliche Elemente eines Dateisystems:
 - Dateien (Files)
 - Verzeichnisse / Kataloge (Directories)
 - Partitionen (Partitions)



Dateisystem (2)

■ Datei (File)

- speichert Daten oder Programme
- enthält Zusatzinformationen



Datei

■ Verzeichnis / Katalog (Directory)

- fasst Dateien (u. Verzeichnisse) zusammen
- erlaubt Benennung der Dateien
- ermöglicht Aufbau eines hierarchischen Namensraums



Verzeichnis

■ Partition (Partition)

- eine Menge von Verzeichnissen und deren Dateien
- sie dienen zum physikalischen oder logischen Trennen von Dateimengen
 - physisch: Festplatte, Diskette
 - logisch: Teilbereich auf Platte oder CD



Partition



- Kleinste Einheit, in der etwas auf den Hintergrundspeicher geschrieben werden kann.
- Unterscheidung:
 - eigentliche Daten (Bild, Text, Programm, ...)
 - Metadaten (Erstellungsdatum, Eigentümer, Zugriffsrechte, ...)

Metadaten / Dateiattribute:

Name: Symbolischer Name, vom Benutzer les- und interpretierbar

- z.B. AUTOEXEC.BAT

Typ: Für Dateisysteme, die verschiedene Dateitypen unterscheiden

- z.B. sequenzielle Datei, zeichenorientierte Datei, satzorientierte Datei

Ort: Wo werden die Daten physisch gespeichert?

- Nummern der Plattenblöcke



Dateiattribute (2)

Größe: Länge der Datei in Größeneinheiten (z.B. Bytes, Blöcke, Sätze)

- steht in engem Zusammenhang mit der Ortsinformation
- wird zum Prüfen der Dateigrenzen z.B. beim Lesen benötigt

Zeitstempel: z.B. Zeit und Datum der Erstellung, letzten Änderung

- für Backup, Entwicklungswerkzeuge, Benutzerüberwachung, ...

Rechte: Zugriffsrechte, z.B. Lese- und Schreibberechtigung

- z.B. nur für den Eigentümer schreibbar, für alle anderen nur lesbar

Eigentümer: Identifikation des Eigentümers

- eventuell eng mit den Rechten verknüpft
- Zuordnung beim Accouting (Abrechnung von Plattenplatz)



■ Erzeugen (Create)

- Nötiger Speicherplatz wird angefordert
- Verzeichniseintrag wird erstellt
- Initiale Attribute werden gespeichert

■ Schreiben (Write)

- Identifikation der Datei
- eventuell Nachfordern von Speicherplatz
- Daten werden auf Platte geschrieben
- eventuell Anpassung der Attribute (z.B. Länge der Datei, Zeitpunkt der letzten Änderung)

■ Lesen (Read)

- Identifikation der Datei
- Daten werden von Platte gelesen
- eventuell Anpassung der Attribute (z.B. Zugriffszeit)



- **Positionieren** des Schreib-/Lesezeigers für die nächste Schreib- bzw. Leseoperation (**Seek**)
 - Identifikation der Datei
 - In vielen Systemen wird dieser Zeiger implizit bei Schreib- und Leseoperationen positioniert
 - Ermöglicht explizites Positionieren
- **Verkürzen (Truncate)**
 - Identifikation der Datei
 - Ab einer bestimmten Position (oder ab Anfang) wird der Inhalt der Datei gelöscht
 - eventuell Freigeben von Speicherplatz
 - Anpassung der Attribute (z.B. Länge der Datei, Zeitpunkt der letzten Änderung)
- **Löschen (Delete)**
 - Identifikation der Datei
 - Entfernen der Datei aus dem Verzeichnis und Freigabe der Plattenblöcke



- Ein Verzeichnis gruppiert Dateien und evtl. weitere Verzeichnisse
- Gruppierungsalternativen
 - Verknüpfung mit Benennung
 - Verzeichnis enthält Namen und Verweise auf Dateien und andere Verzeichnisse (z.B. UNIX, Windows)
 - Gruppierung über Bedingung
 - Verzeichnis enthält Namen und Verweise auf Dateien, die einer bestimmten Bedingung gehorchen:
 - z.B. gleiche Gruppennummer in CP/M
 - z.B. eigenschaftsorientierte und dynamische Gruppierung in BeOS-BFS
- Verzeichnis ermöglicht das Auffinden von Dateien
 - Vermittlung zwischen externer und interner Bezeichnung (Dateiname – Plattenblöcke)



- **Lesen** der Einträge (**Read, Read Directory**)
 - Daten des Verzeichnisses werden gelesen und meist eintragsweise zurückgegeben
- **Erzeugen** und **Löschen** der Einträge erfolgt implizit beim Anlegen bzw. Löschen der Dateien
- **Erzeugen** von Verzeichnissen (**Create, Create Directory**)
- **Löschen** von Verzeichnissen (**Delete, Delete Directory**)

Attribute von Verzeichnissen

- Die meisten Attribute von Dateien treffen auch auf Verzeichnissen zu
 - Name, Ortsinformation, Größe, Zeitstempel, Rechte, Eigentümer, ...

