# System-Level Programming

## 25 File Systems – Introduction

**Peter Wägemann**

Lehrstuhl für Informatik 4
Systemsoftware

Friedrich-Alexander-Universität
Erlangen-Nürnberg (FAU)

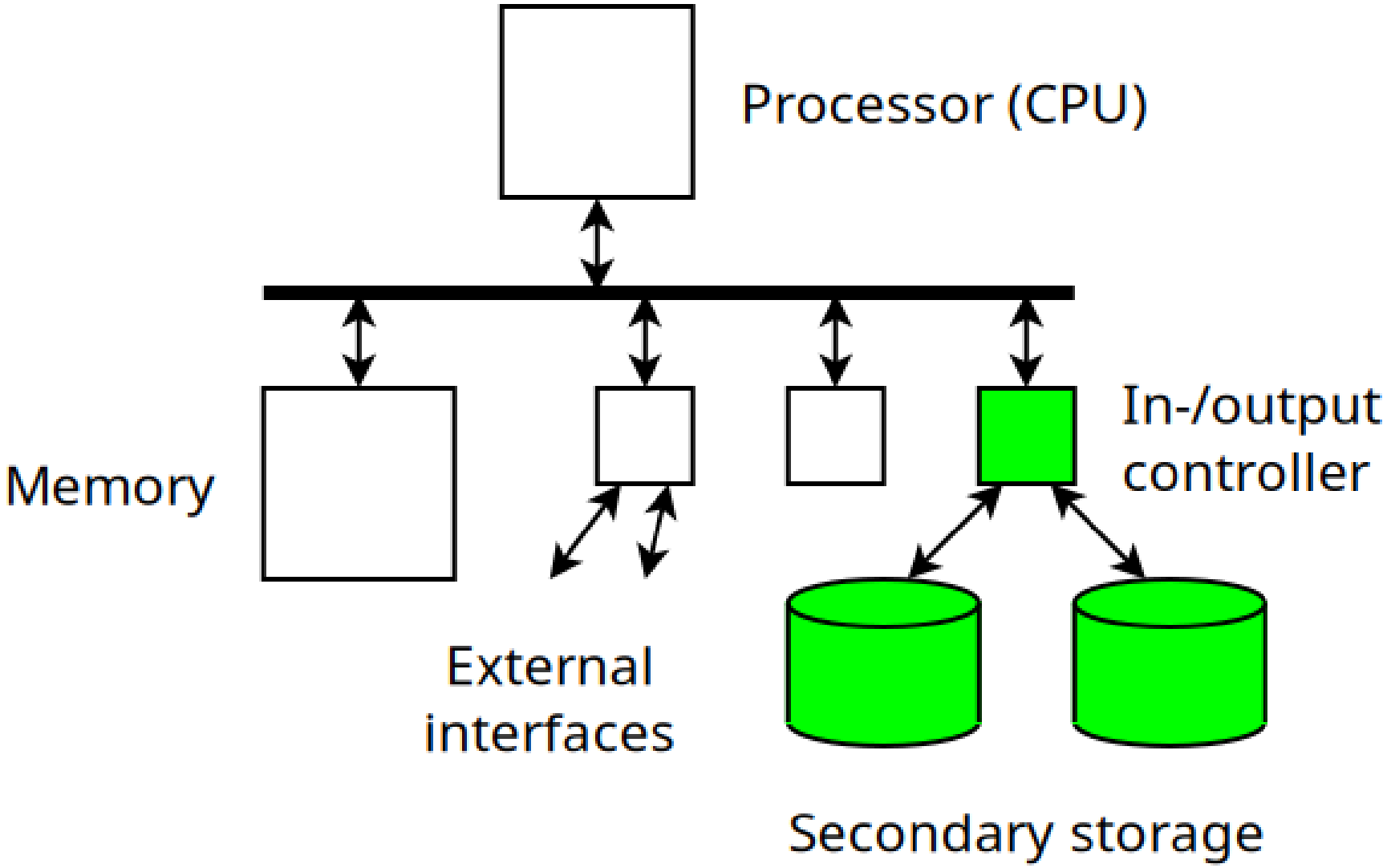
Summer Term 2025

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

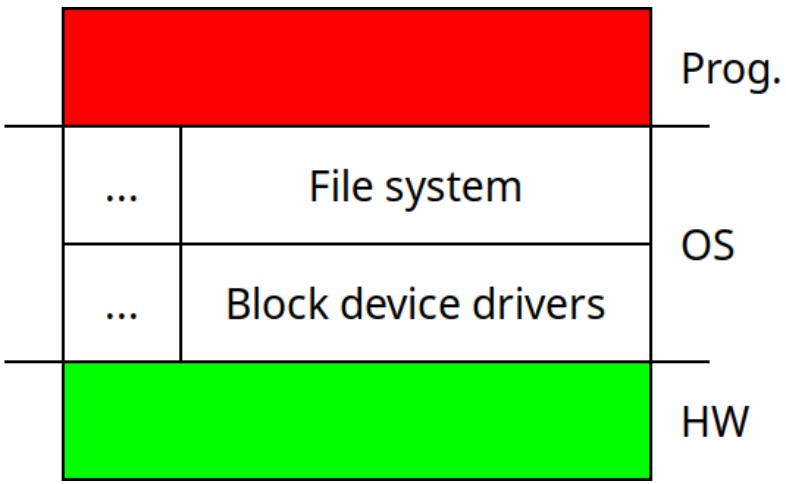


# Overview

- Classification



# Overview (2)




Application:

- reads/writes file contents
- reads/writes directory contents

File system:

- reads/writes blocks

Block device driver:

- reads/writes I/O register

Hardware:

- reads/writes bytes from/to a disk

# Hardware

- Storage medium (e. g., hard disks, SSD/flash storage, DVD, CD-ROM) with differences; examples
  - size of **blocks**:
    - hard disks: 512 bytes/block
    - CDs: 2048 bytes/block
    - flash: 4096 bytes/block
  - usage of the blocks
    - flash storage only has a limited amount of write cycles for each block ⇒ evenly distribute data
    - hard disks can access neighboring blocks faster than others
  - (typical) size of the medium
    - CD-ROM: approx. 750 MByte
    - DVD: approx. 8.5 GByte
    - hard disk: approx. 4 TByte
    - SSD: approx. 1 TByte



# Block Device Drivers

Example: PC-IDE hard-disk driver (simplified):

```
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;
    }
}
```



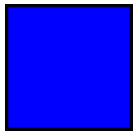
# File System

- File systems **persistently store data and programs** in files
  - user does not have to care about accessing and managing different storage types
  - unified view of the background storage
- Essential parts of a file system:
  - files
  - directories
  - partitions

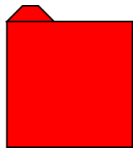


# File System (2)

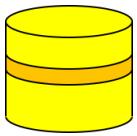
- File
  - stores data or programs
  - contains additional information
- Directory
  - combines files and other directories
  - makes it possible to give names to files
  - enables an hierarchical name space
- Partition
  - a set of directories and their files
  - are used to physically or logically split amounts of data
    - physical: hard disk, floppy
    - logical: partitioned area on a disk or CD



File



Directory



Partition



# Files

- Smallest unit that can be written to the persistent storage
- Classification:
  1. actual files (image, text, program, ...)
  2. meta data (date of creation, owner, access permissions, ...)
- Meta data/file attributes:

  Name: symbolic name, readable by the user
  - e. g., `AUTOEXEC.BAT`

  Type: For file systems that differentiate between file types
  - e. g., sequential file, character-oriented file, record-oriented file

  Place: Where are the files stored physically?
  - number of the block on the disk



# File Attributes (2)

Size: Length of the file in different units of measure (e. g., bytes, blocks, records)

- closely related to the information about storage location
- is used to verify the bounds of the file, e. g., when reading it

Time stamp: e. g., time and data of creation, last change

- for backups, development tools, monitoring users, ...

Permissions: access permissions (e. g., read & write permissions)

- e. g., only writable by the owner, other users have read-only access

Owner: identification of the owner

- closely related to the permissions
- allocation for accounting (users' quota of disk space)



# Operations on Files

- **Create**
  - required storage space is requested
  - entry in the directory is created
  - initial attributes are stored
- **Write**
  - identification of the file
  - possibly request more storage space (reallocation)
  - data is written to the disk
  - possibly modification of the attributes (e. g., length of the file, time of last change)
- **Read**
  - identification of the file
  - data is read from the disk
  - possibly modification of the attributes (e. g., time of last access)



# Operations on Files (continued)

- **Positioning** of the write/read pointer for the next write or read operation **(seek)**
  - identification of the file
  - in many systems, this positioning takes place automatically when a read or write operation is requested
  - enables explicit positioning
- **Truncate**
  - identification of the file
  - beginning at a certain position (or from the start) the contents of the file get deleted
  - storage is possibly freed
  - modification of the attributes (e. g., length of the file, time of last change)
- **Delete**
  - identification of the file
  - removing of the file from the directory and release of blocks on the disk



# Directories/Catalogs

- A directory groups files and other directories
- Alternatives for grouping
  - Linking by naming
    - directory contains names and references to files and other directories (e. g., UNIX, Windows)
  - Grouping by conditions
    - directory contains names and references to files that comply with certain conditions:
      – e. g., same group number in CP/M
      – e. g., attributes-dependent grouping or dynamic grouping in BeOS-BFS
- Directories enable locating files
  - link between internal and external identification (name of the file – blocks on the disk)



# Operations on Directories

- **Reading** of directory content
  - data of the directory is read and typically returned entry-by-entry
- **Creating** and **Deleting** takes place implicitly during creation and deletion of files
- **Creation** of directories
- **Deletion** of directories

Attributes of directories

- Most attributes of file also apply for directories
  - name, storage location, size, time stamps, permission, owner, ...