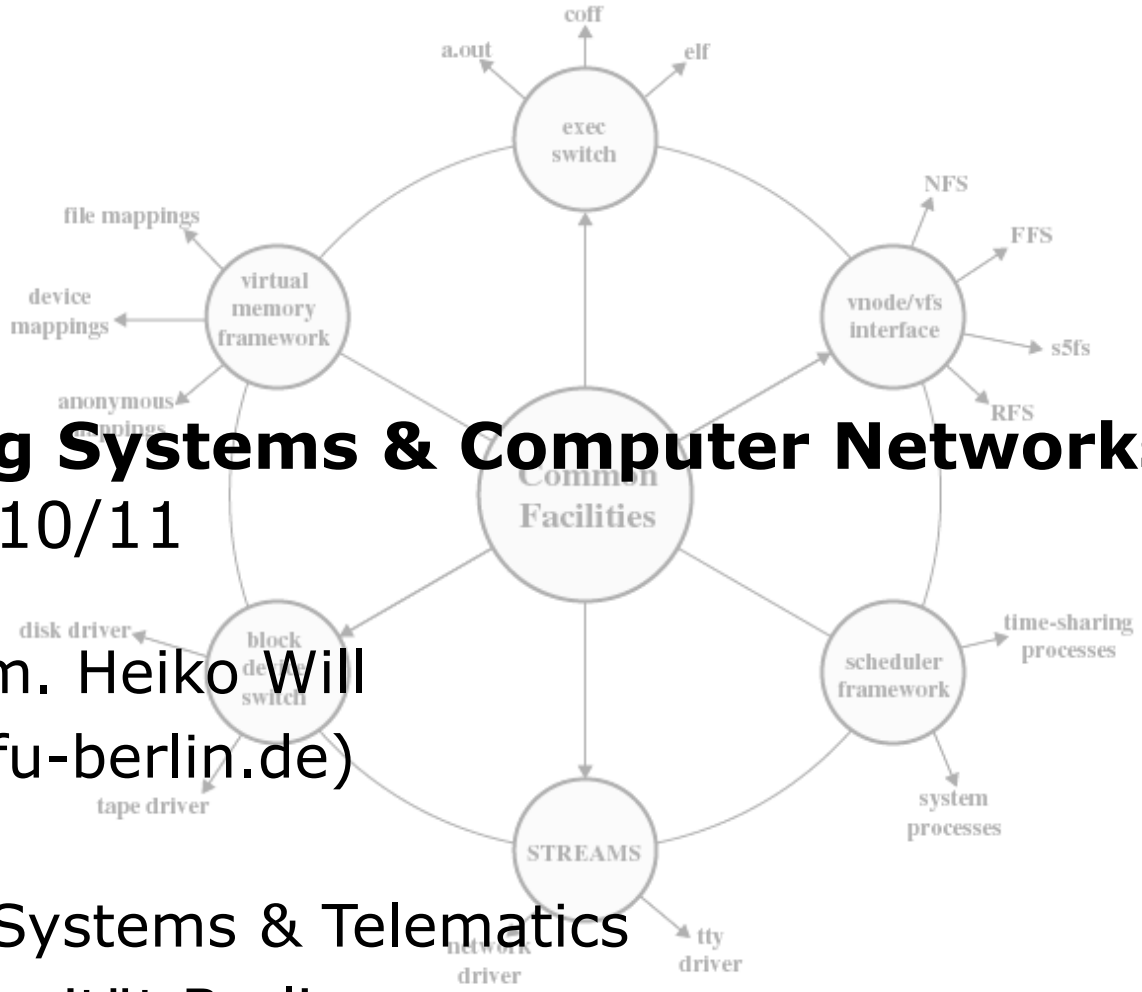



# Operating Systems & Computer Networks

Winter 2010/11

Dipl.-Inform. Heiko Will
(hwill@inf.fu-berlin.de)

Computer Systems & Telematics
Freie Universität Berlin

# Structure and Content of CST Lectures

Freie Universität Berlin

**Praktikum Mobilkommunikation** (8/6)
Medienzugriff, Mobile IP, Mobiles Web

**Embedded Sensor Web** (7+)
Projekte rund um Sensornetze

**Seminar Technische Informatik** (6+)
Forschung in Mobilkommunikation, eingebettete Systeme, Internet

**Mobilkommunikation** (8/6)
Drahtlose Übertragung, Medienzugriff, GSM, 3G, WLAN, Mobile IP, Ad-hoc-Netze, WAP

**Embedded Internet** (7)
Protokolle, Dienste, Internet, TCP/IP, Betriebssysteme für eingebettete Systeme

**Mikroprozessorpraktikum** (7/5)
Programmierung eingebetteter Systeme, mobile Endgeräte, Mikrocontroller, Steuerungssysteme

**Telematik** (5)
Protokolle, Dienste, Standards, LAN, Internet, TCP/IP, WWW, Sicherheit, ISDN/IN/ATM, Dienstgüte, Multimedia, IPv6, MPLS

**Praktikum Technische Informatik (TI IV)** (4)
Eingebettete Systeme, Schnittstellen, Treiber, Betriebssystem – programmieren, vernetzen, interagieren

**Betriebs- und Kommunikationssysteme (TI III)** (3)
Ein-/Ausgabe, DMA/PIO, Unterbrechungen, Puffer, Prozesse/Threads, UNIX/Windows, Netze, Medienzugriff, Protokolle, TCP/IP, Internet

**Rechnerarchitektur (TI II)** (2)
Harvard/v. Neumann, Mikroarchitektur, RISC/CISC, VLIW, Pipelining, Cache, Speicherhierarchie, Assembler, Multiprozessorsysteme

**Grundlagen der Technischen Informatik (TI I)** (1)
Schaltnetze, Schaltwerke, Logikminimierung, Gatter, Speicher, Halbleiter, Transistoren, CMOS, AD/DA-Umsetzer

④ Semester
Bachelor
Master

# Content (1)

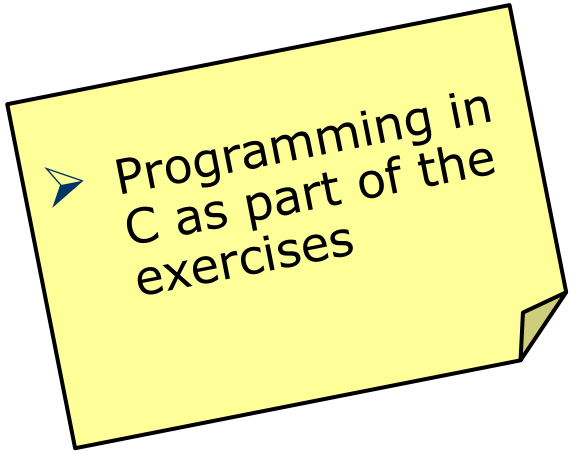
1. Introduction and Motivation
   - Tasks
   - Services
   - Virtual Resources
   - Historical Perspective
   - Examples
   - Tools
2. Subsystems, Interrupts and System Calls
   - System Structure
   - Flow of Control
   - System Library
   - POSIX
3. Processes
   - Definition
   - Implementation
   - State Model
4. Memory
   - Paging & Segmentation
   - Virtual Memory
   - Swap Policies
5. Scheduling
   - Types of Scheduling
   - Decision Modes
   - Process Priorities
   - Scheduling Policies
6. I/O and File System
   - Devices
   - Buffering and Caching
   - Files and Directories
7. Booting, Services, and Security
   - System Startup
   - System Services
   - Security Issues

# Content (2)

8. Networked Computer & Internet
   - Sockets
   - Internet
   - Layers
   - Protocols
9. Host-to-Network I
   - Physical Layer
   - Media
   - Signals
   - Modems
10. Host-to-Network II
    - Data Link Layer
    - Framing, Flow Control
    - Error Detection / Correction
    - Point-to-Point Protocol
11. Host-to-Network III
    - Topologies
    - Medium Access
    - Local Area Networks
      - Ethernet, WLAN
12. Internetworking
    - Switches, Routers
    - Routing
    - Internet Protocol
    - Addressing
13. Transport Layer
    - Protocol Mechanisms
    - TCP, UDP
    - Addressing, Ports

# Content (3)

14. Applications
    - Domain Name System
    - Email
    - World Wide Web
15. Network Security
    - Basic Concepts & Terms
    - Cryptology
    - Examples
        - Firewalls
        - Virtual Private Networks (VPNs)
        - IP Security
        - Email Security with PGP
16. Example
    - Under the Hood of Surfing the Web



# Course Organization

Freie Universität Berlin

General:

- Lecture
  - Friday, 10:00-12:00h, HS, Takustr. 9
- Office Hours
  - Heiko Will: after the lecture
  - Tutors: during tutorials
- News and Updates
  - http://lms.fu-berlin.de/
- Tutorials
  - Groups of approx. 30 students
  - Time/location depends on group
  - Registration via WWW

Assignments:

- New assignments each week
  - Available on blackboard
- Discussion
  - During the tutorials
- Practical assignments
  - Pool computers available
  - More during lecture/tutorials
- Handing in
  - *Right on time!*
  - Each tutor has his/her own box, 1st floor, Takustr. 9
  - Solutions handed in too late will be ignored!

# Criteria for Successful Participation

- Active participation in the tutorials is essential!
  - Minimum **n-2 times present**
- Hand in your assignments on time
  - Teamwork is required with two students per team
- Successful submission of at least **n-2 assignments**
  - Successful = at least 50% of the max. number of points
- Each student with a correct answer must be able to present the assignment during the tutorials
  - At least **one presentation** during the tutorials
- At least 50% of the max. number of points required in the exam
  - Only the exam counts for grading!

# Literature

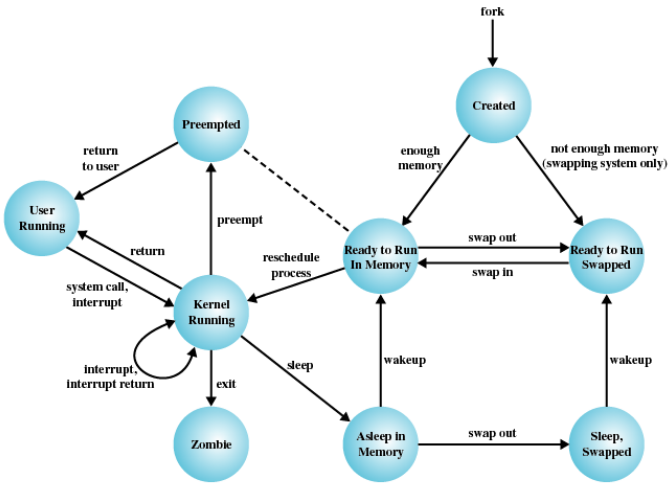
- Printouts of the slides
  - Print it yourself

- The course is based on the books:
  - William Stallings. Operating Systems: Internals and Design Principles, 5th Edition. Prentice Hall International, July 2004. ISBN-10: 0131479547, ISBN-13: 978-0131479548.
  - Larry L. Peterson, Bruce S. Davie: Computernetze - Eine systemorientierte Einführung, 3. Auflage. dpunkt Verlag, Heidelberg, April 2004, ISBN-13: 978-3-89864-242-2.

- Additional literature:
  - Erich Ehses, Lutz Köhler, Petra Riemer, Horst Stenzel und Frank Victor. Betriebssysteme, 1. Auflage. Pearson Studium. August 2005. ISBN: 3-8273-7156-2.
  - Gary Nutt. Operating Systems – A Modern Perspective, 2nd Edition. Addison Wesley. 2002. ISBN: 0-201-74196-2
  - Andrew S. Tannenbaum. Modern Operating Systems, 2nd Edition. Prentice Hall. February 2001. ISBN: 0-130-31358-0

# Operating Systems & Computer Networks

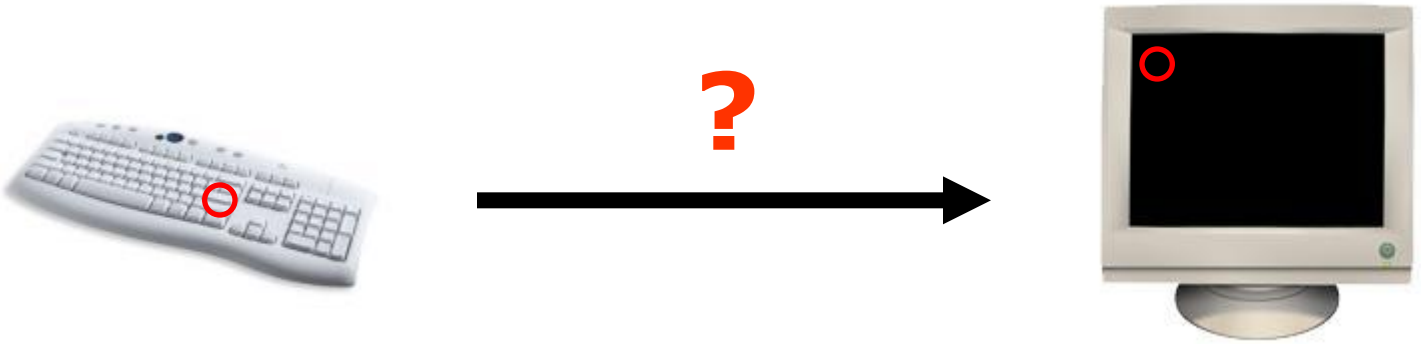
## Introduction and Motivation



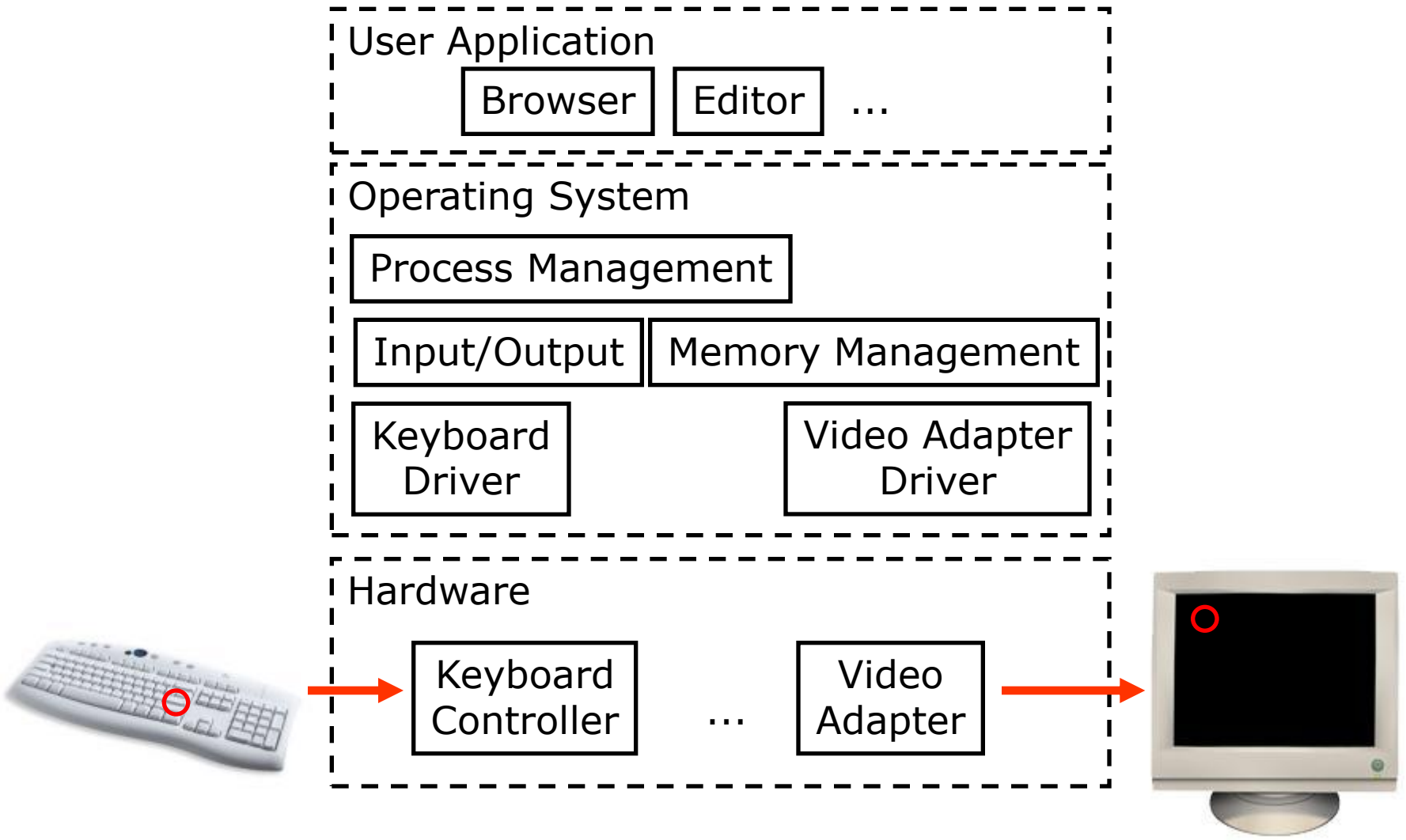
# Content (1)

# Operating System (OS) Example

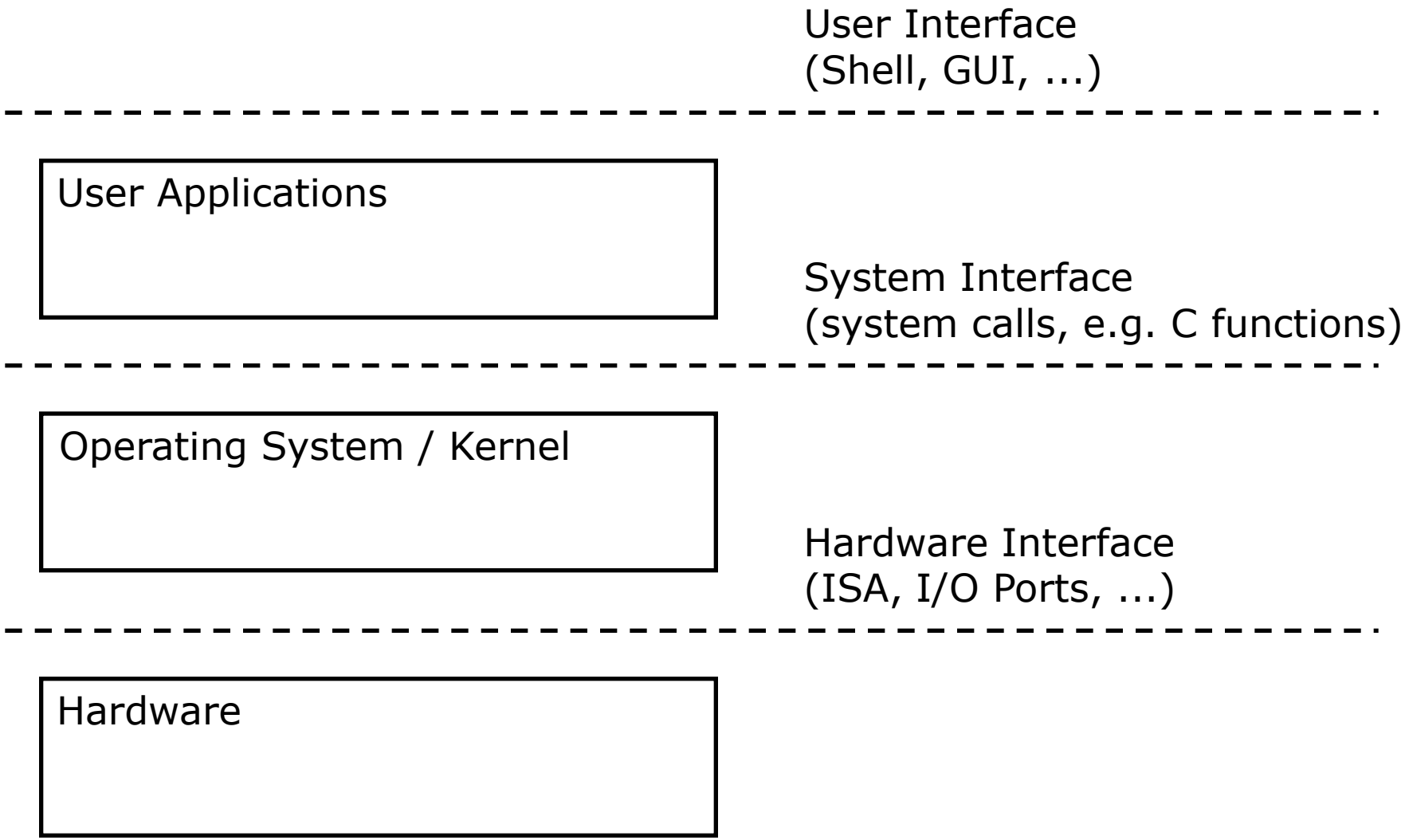
- What happens if one presses a key on the computer?



# Operating System Example

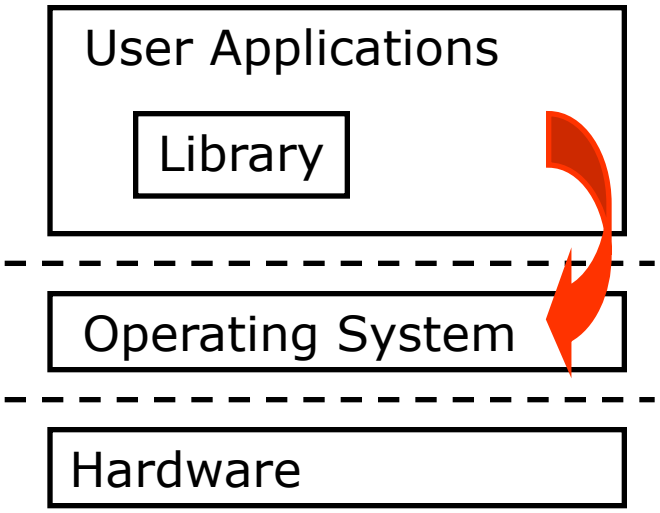


# Layers of Abstraction



# System Interface und System Calls

- System interface is the only way for user applications to interact with the operating system.
- System interface consists of system calls (supervisor calls), e.g. POSIX.




- High-level programming languages hide systems calls in their library routines.

# Tasks of an Operating System



- Typical services of a *general* purpose OS includes:
  - Program execution
  - Access to I/O-devices
    - Hardware abstraction
  - Controlled access to files
    - Non-volatile memory
  - Access control
    - Security / user management
  - Error detection and error handling
    - Both hardware and software
  - Logging
- *Special* purpose operating systems focus on different services, e.g. real-time or communication requirements.

# Goals of an Operating System

- Ease of use for users and programmers
- Efficiency when managing limited resources
- Possibility to evolve
  - New hardware standards
  - Changing user requirements

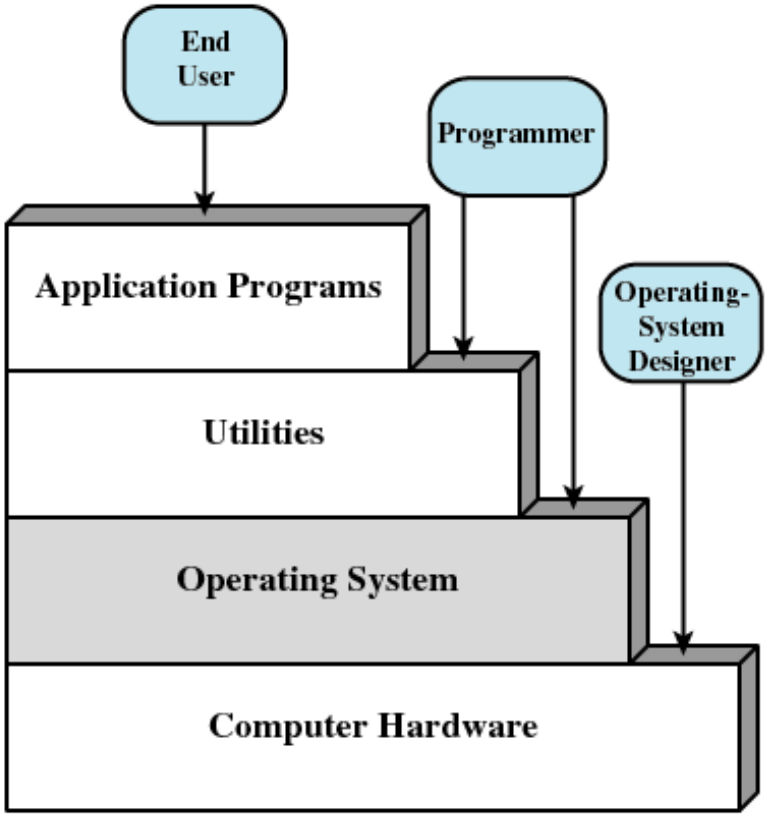



**Figure 2.1 Layers and Views of a Computer System**

# Managing Resources

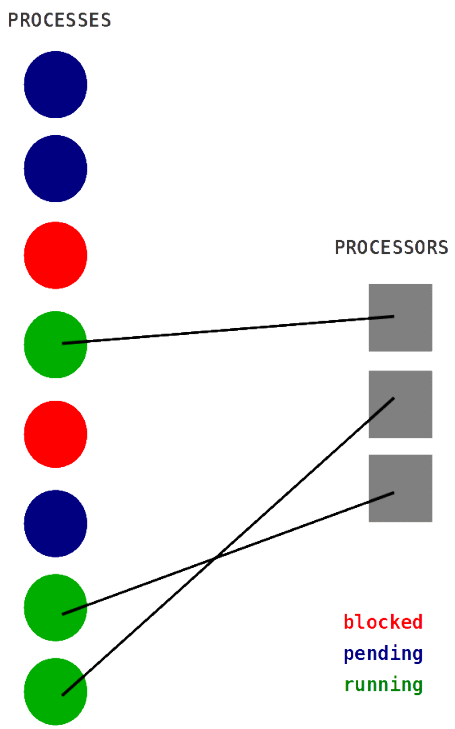
- Hardware provides the basic computing resources such as
  - Processor(s)
  - Memory
  - Persistant storage
  - Network connection
- OS **virtualizes resources** to permit controlled sharing and isolation
  - virtual instances of a resource are created
- OS provides virtual resources for user applications

# Virtual Resources

- Virtual resources and corresponding real resources:
  - Processes — processor(s)
  - Virtual Memory — main memory
  - Files — persistent memory
  - Ports — network adapter
- Advantages:
  - Easy to use — through procedural interface (system calls)
  - Secure — against hardware and software errors or manipulation

# Processes

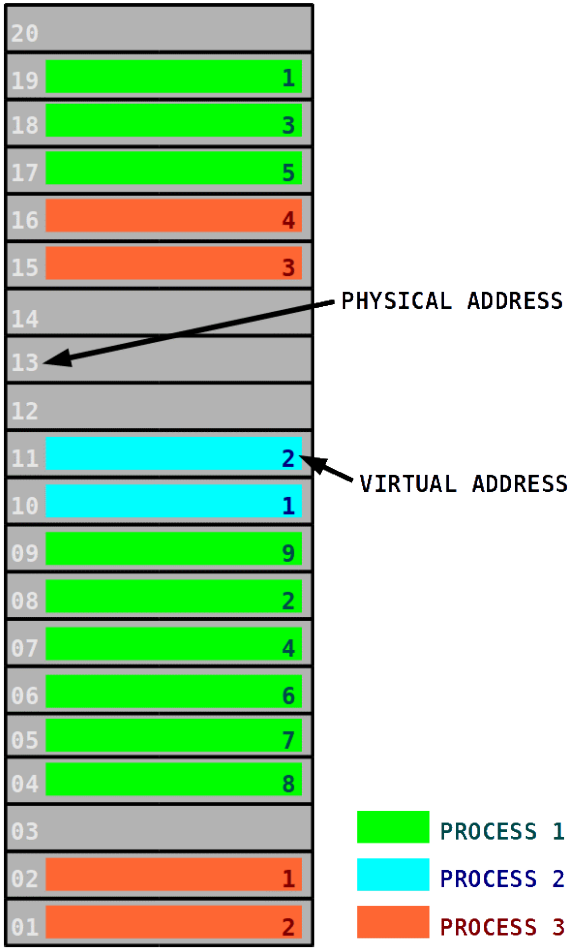
- Number of processes is not limited by the number of processors: **Multitasking**
- Processor is used efficiently: time is not wasted by processes that are waiting on I/O devices
- reduced latency (=response time)
- Different **process states**, e.g.
  - running – executing
  - pending – ready to execute
  - blocked – not ready to execute



# Virtual Memory

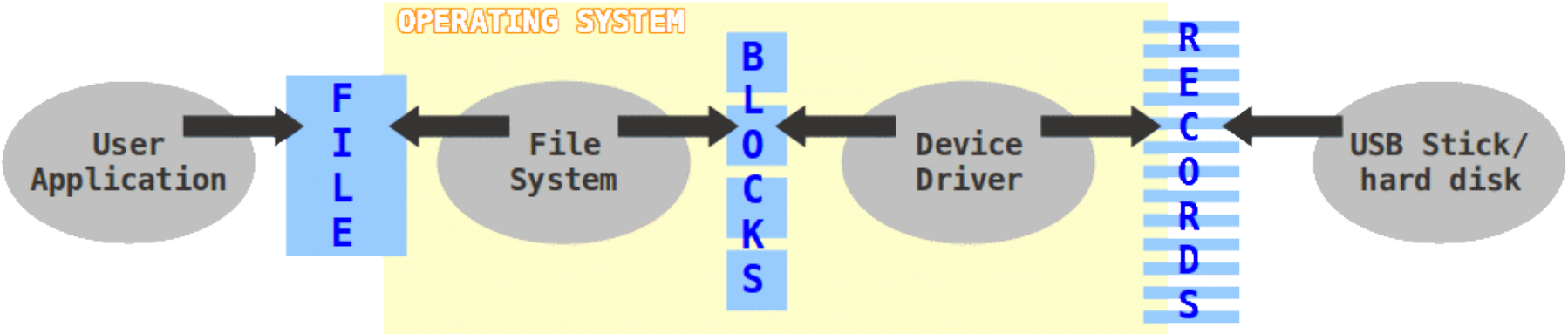
- Managed by the Memory Management Unit (MMU)

- Transportability:
  - machine independent code – program does not depend on memory architecture
- Security:
  - memory access is restricted to memory units "owned" by a process
- Efficiency:
  - external fragmentation is avoided



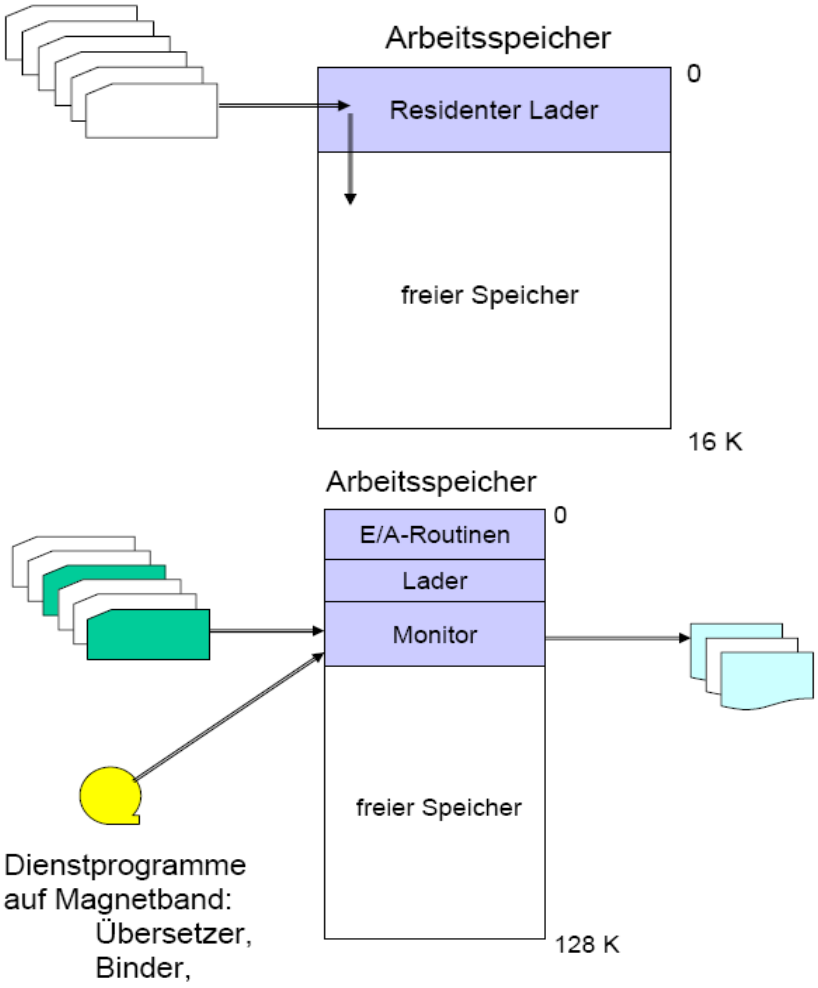
# Files

- Managed by a filesystem
- Persistent objects for long-term data storage
- Stored in secondary memory (e.g. tape, hard disk, USB flash drive)
- Similar to virtual memory - file name instead of virtual adress



# History of Operating Systems

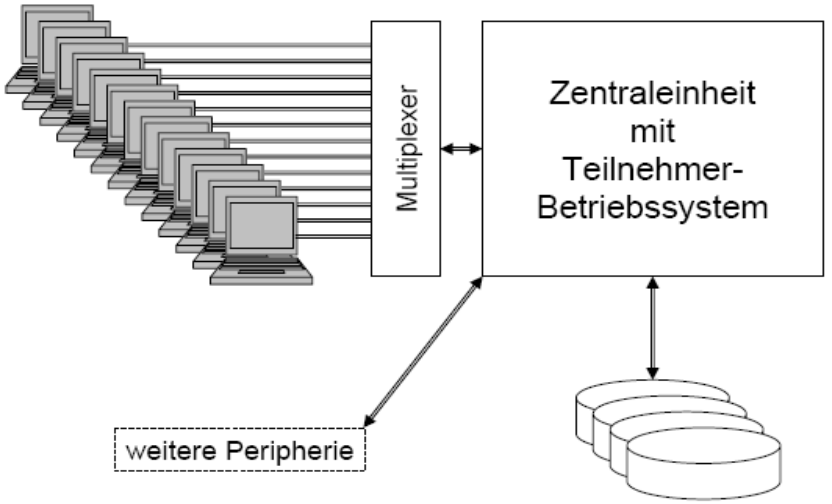
- Development of operating systems follows changes in computer architecture
- Loader (1950, e.g. IBM 704)
  - Loads programs into memory
- Batch System (1960, e.g. IBM 7090, Zuse Z 23, Telefunken TR4)
  - Processing of jobs stored on punch cards
  - Manual job control by human operator

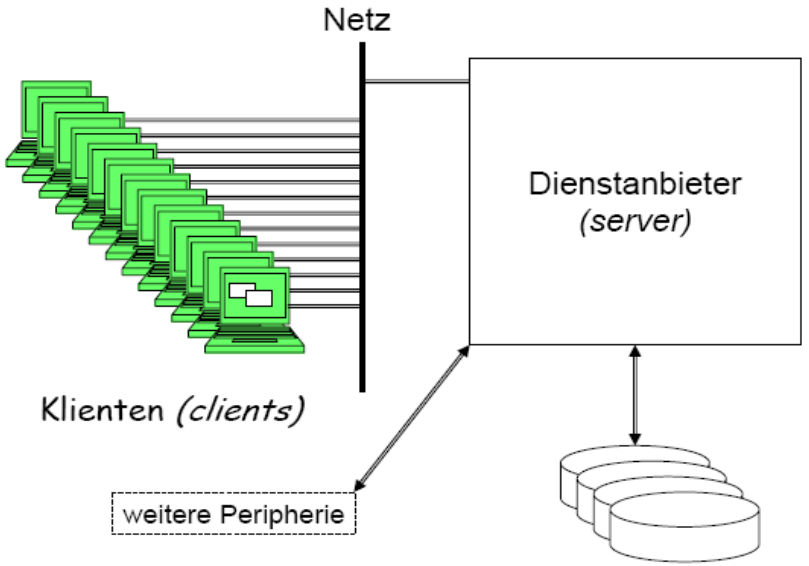


# History of Operating Systems (2)

- Multi-User / Time Sharing Systems (1970, e.g. IBM OS/360, TSS, T.H.E., Multics, UNIX)
  - Many terminals connected to one computer
  - Interactive control for users
  - Multitasking

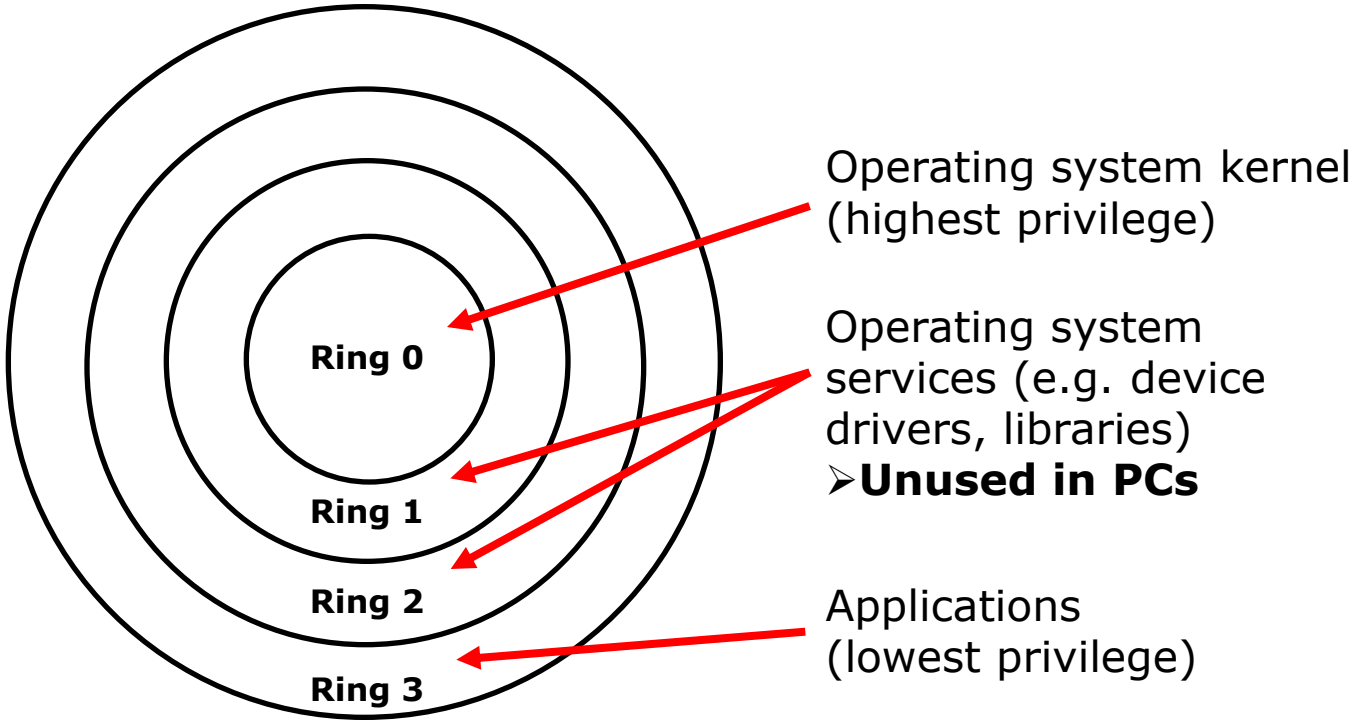
- Personal Computing und Client/Server (1980/90, e.g. Apple Lisa, MS Windows, Linux, Solaris, HP-UX, ...)
  - Intelligent workstations
  - GUI / Window mode



# Protection Rings

- Hardware provides hierarchical privilege levels
  - Inner rings have access to outer ring's resources
  - Outer rings may access inner rings through predefined gateways

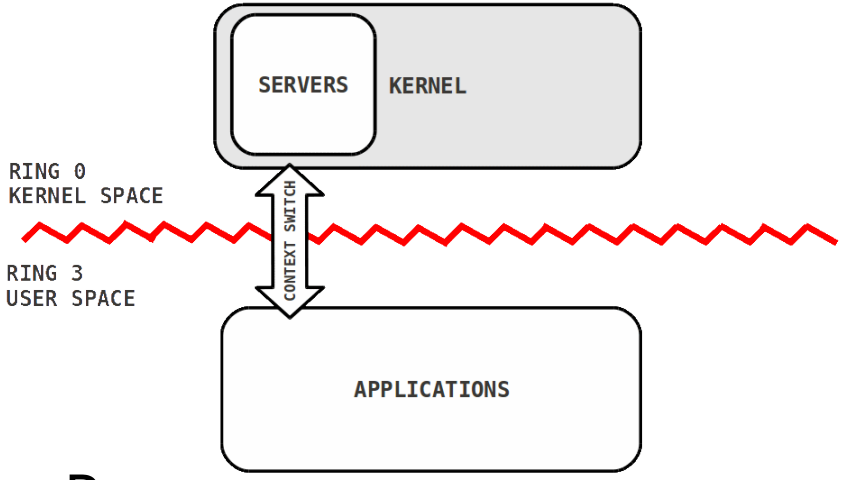


# Operating System Kernel

- Kernel implements basic layer of abstraction
- Runs with full access to hardware (Ring 0)
- Context Switch: switching from one process to another
  - a certain amount of time is required for doing the administration, e.g saving and loading registers

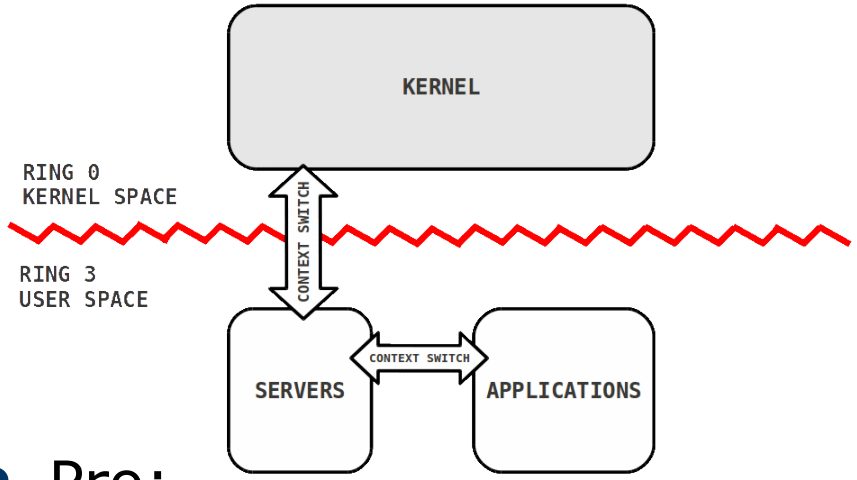
# Monolithic versus Microkernel

- Monolithic Kernel



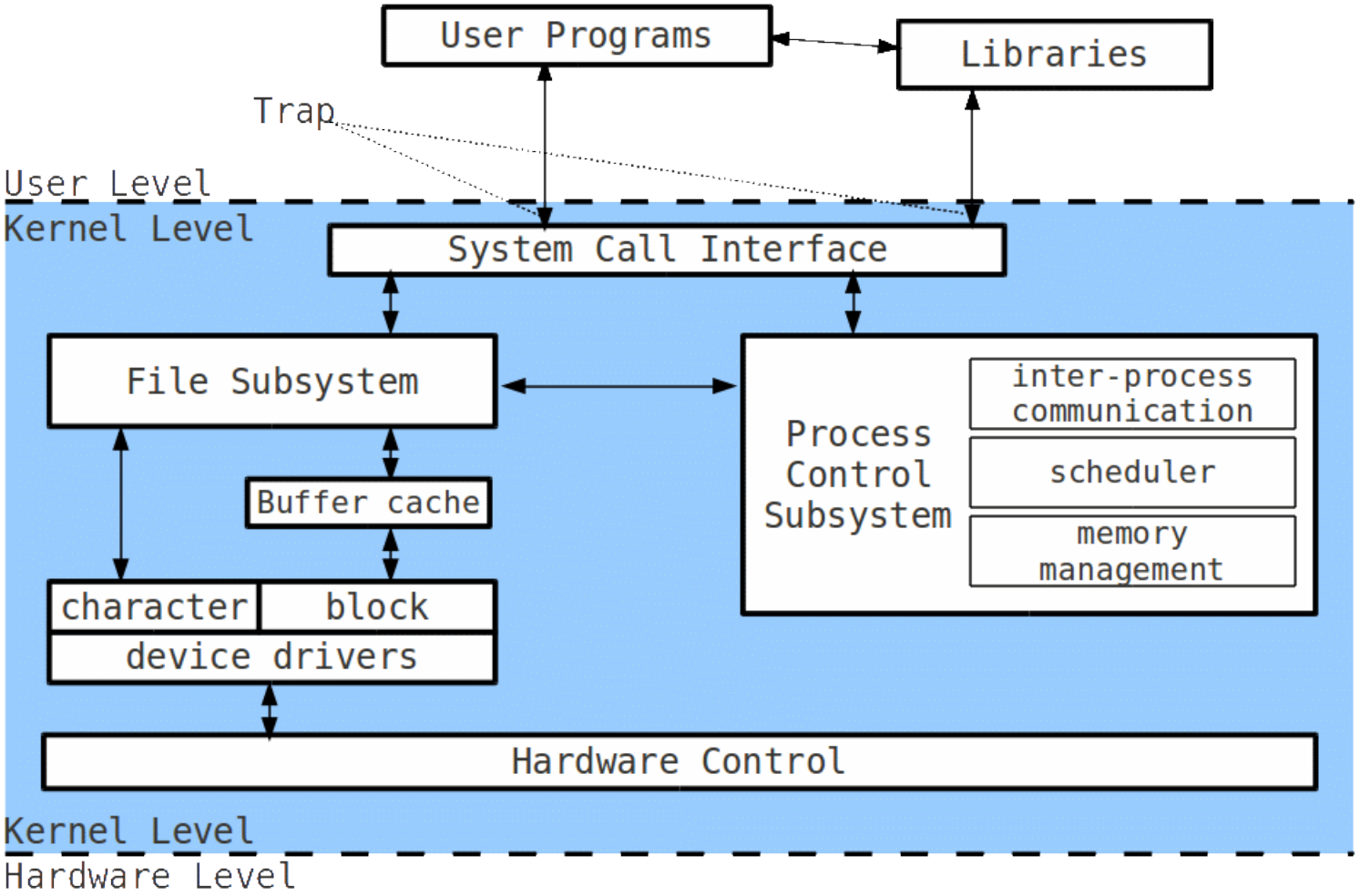

  - Pro:
    - less context switches
    - no expensive communication
  - Contra:
    - complications when exchanging functionality

- Microkernel

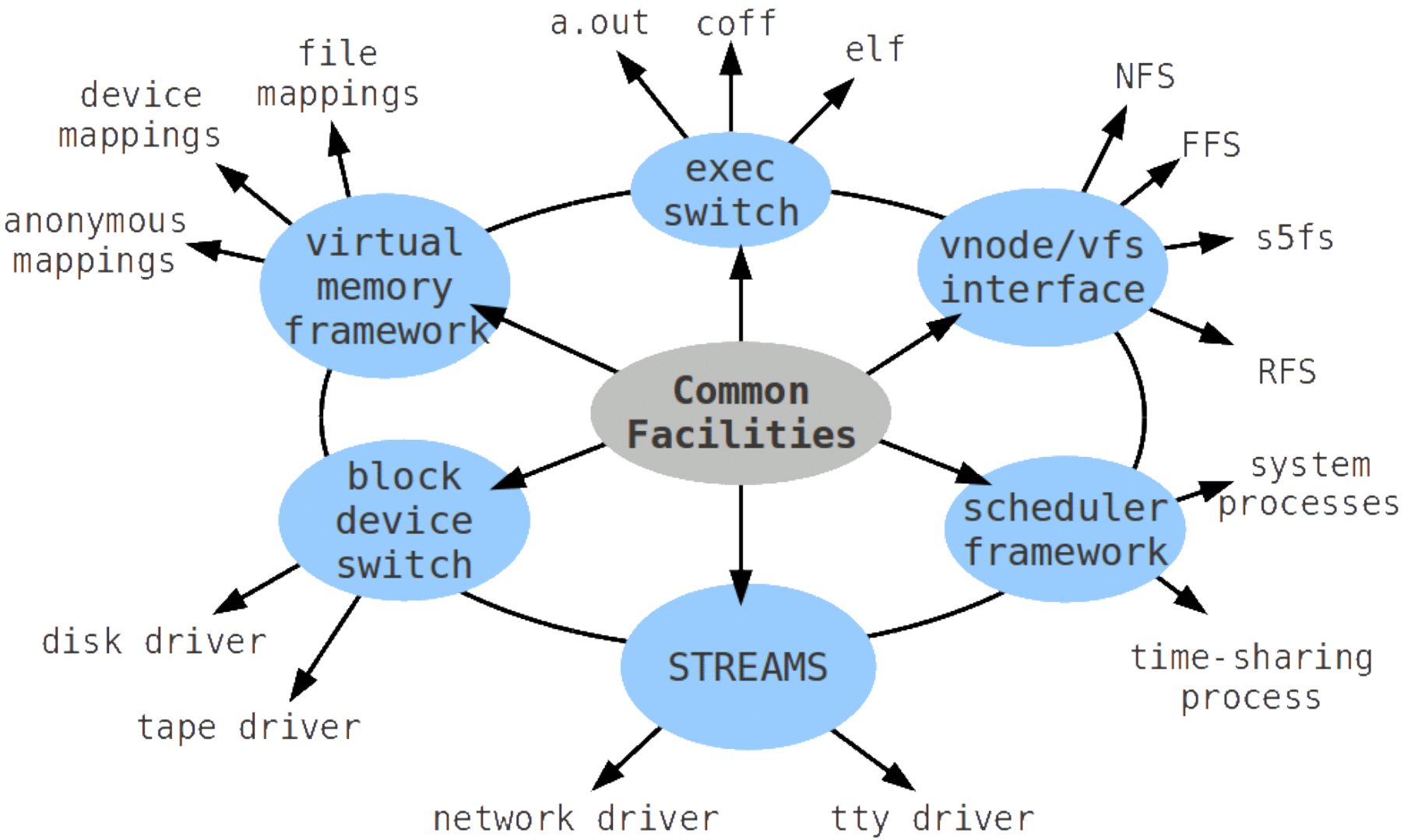



  - Pro:
    - strict interfaces
    - less complexity, clear structure
  - Contra:
    - speed
    - synchronization

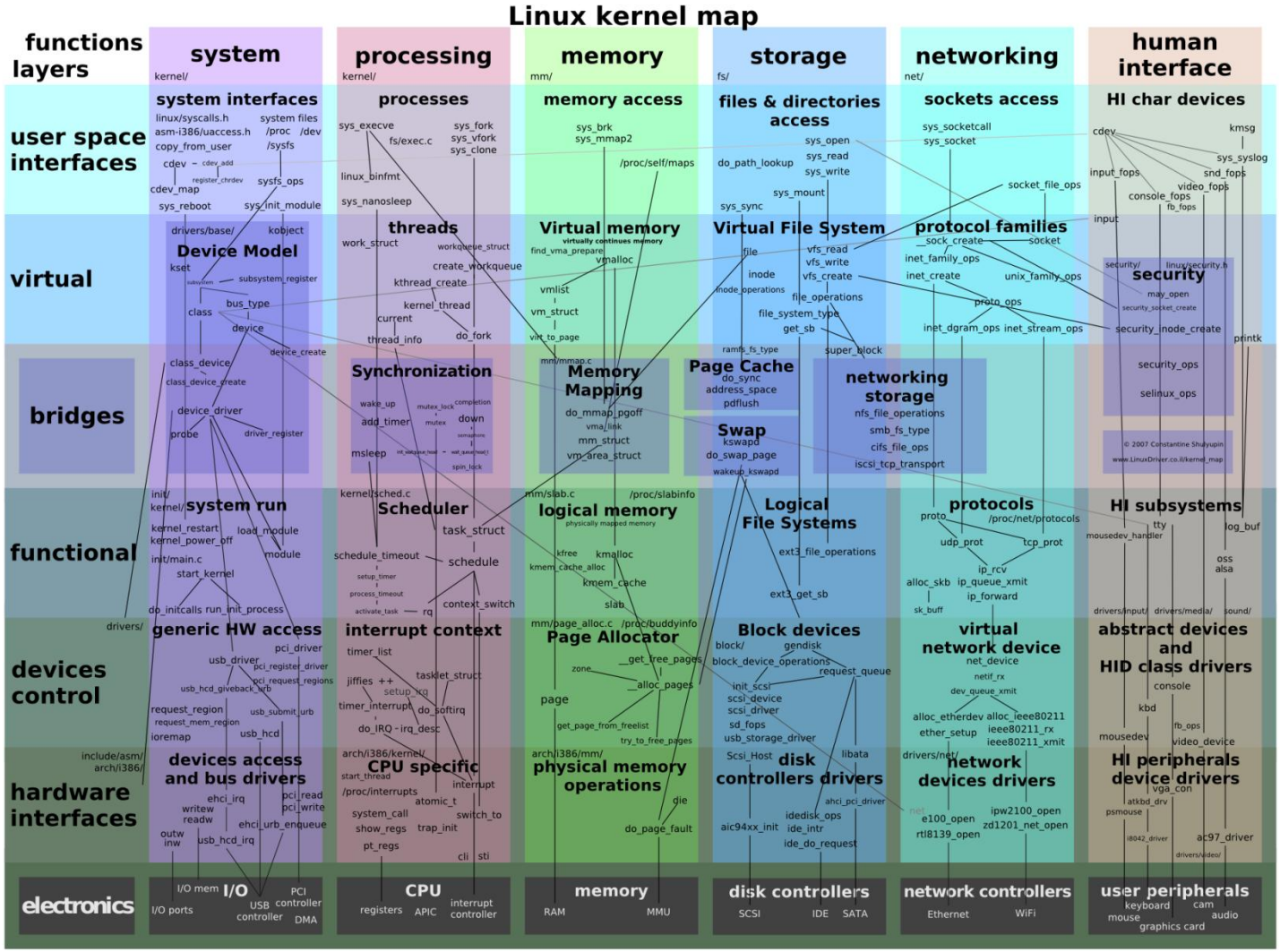
# Examples – UNIX



# Examples – UNIX by Services



# Examples – Linux




Source: http://www.makelinux.net/kernel_map

# A Word About Synchronization

Concurrency handling is outside the scope of this lecture.

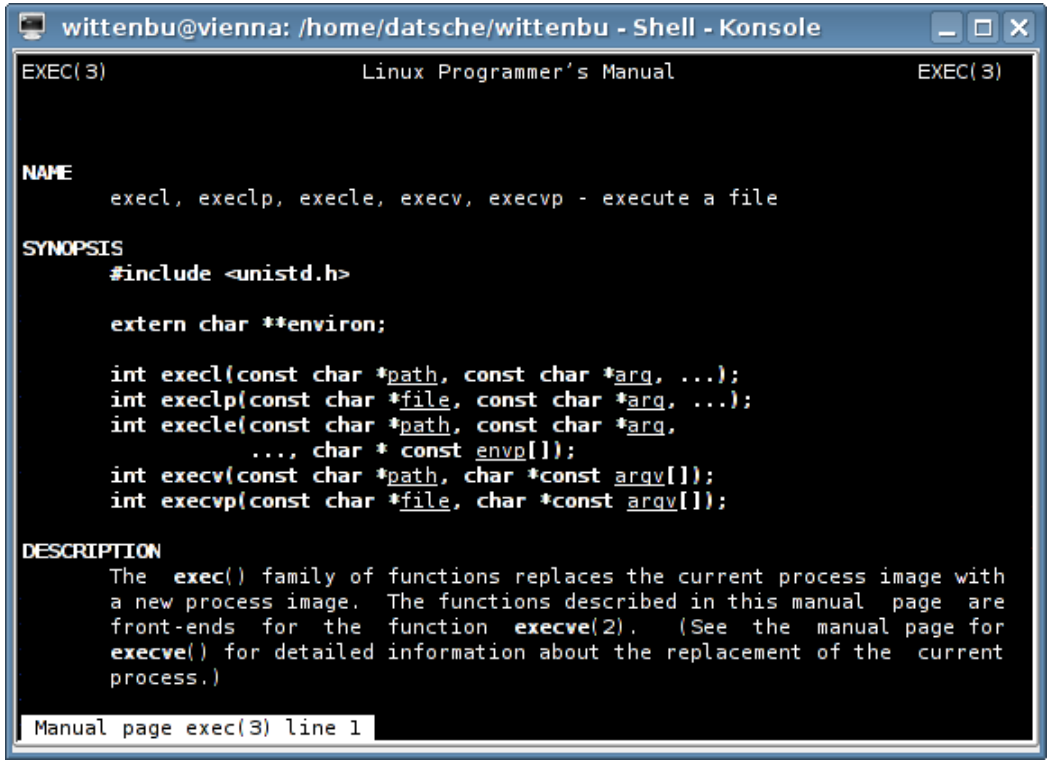
- Course "Nichtsequentielle Programmierung" in summer term

Some pointers/methods/ideas:

- In hardware:
  - Atomic operations:
    - ISA instructions that are guaranteed by design to run to completion
  - Interrupts:
    - Enable/disable interrupts via special ISA instructions
    - Allows other interrupt handlers to run to completion
- In software:
  - Spinlocks (busy waiting):
    - Short-term synchronization mechanism
    - Low overhead, avoid re-scheduling, wasteful on resources
  - Semaphores (wait queues):
    - Long-term synchronization mechanism
    - Synchronize for events on special purpose data structures

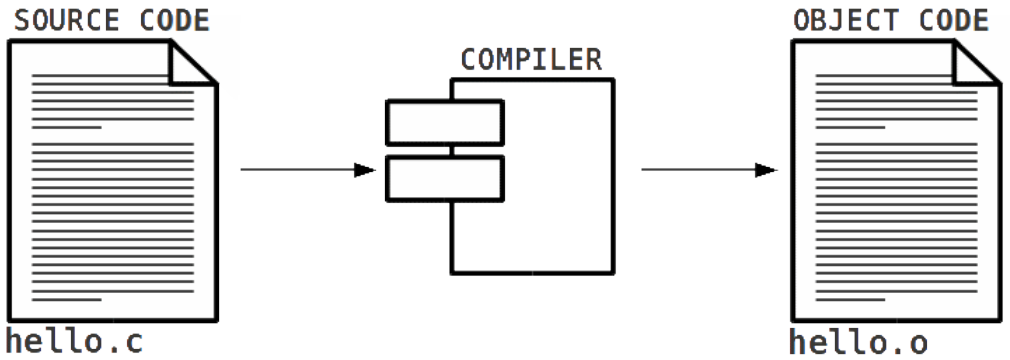
# Manual Pages

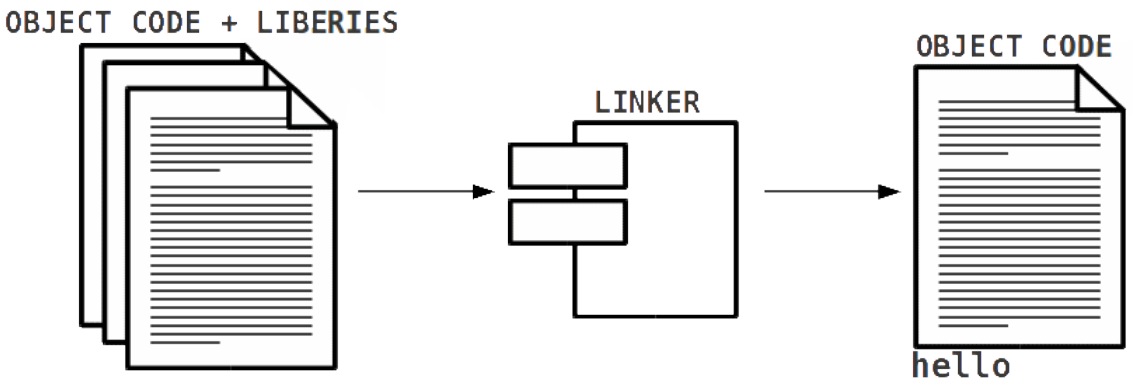
- UNIX-utility **man**
- E.g. **man exec**:



# Program Compilation

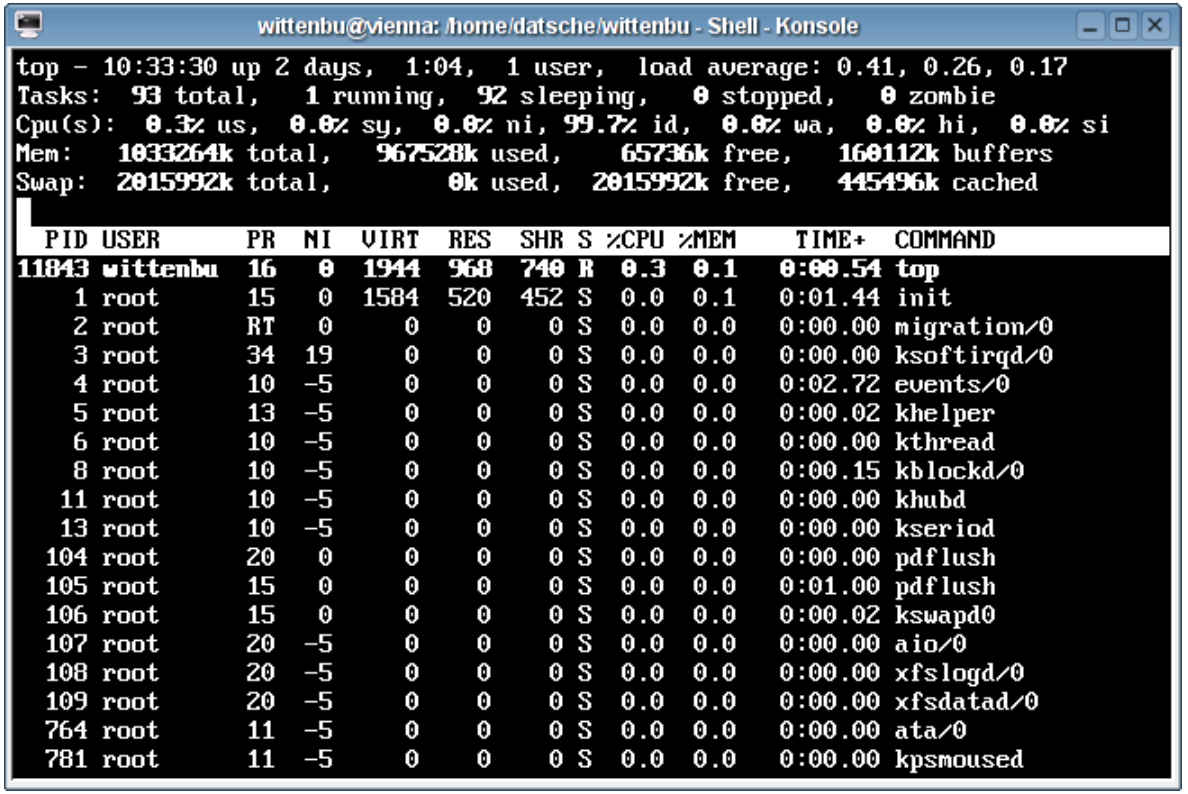
- Toolchain: set of programming tools that are used to create a product (executable)



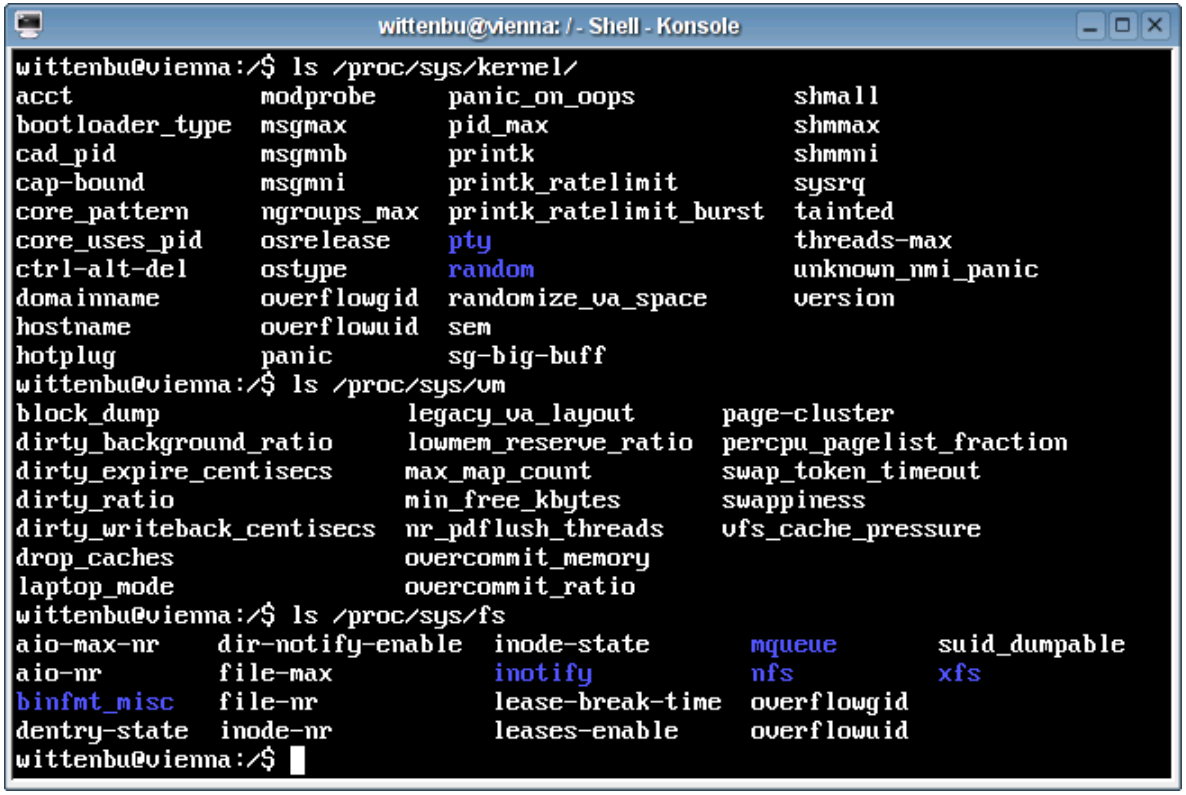
# Process Monitor

- UNIX utility **top**:



# Kernel Parameters

- directories **/proc** and **/sys**
  - virtual directories that reflect general kernel behaviors („everything is a file“)



# Content (1)