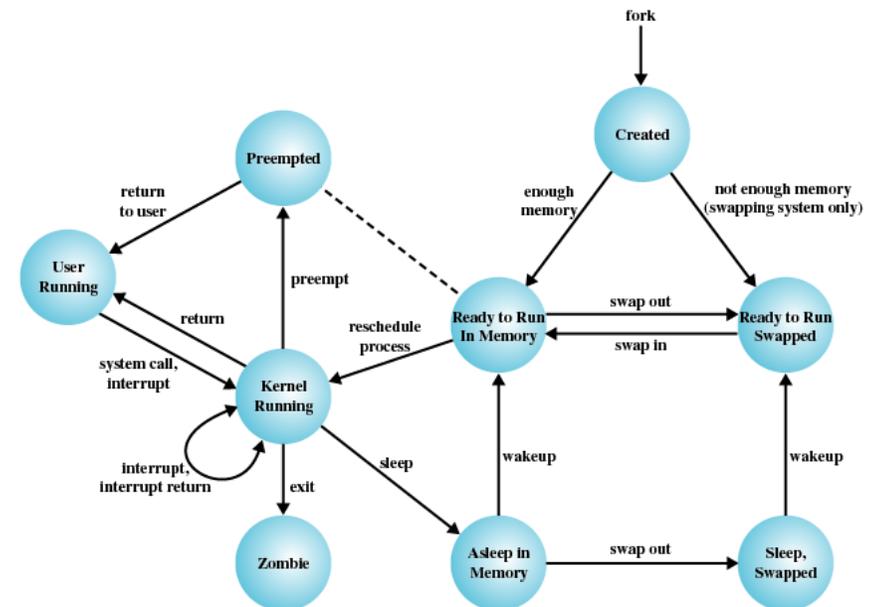


# TI III: Operating Systems & Computer Networks

## Introduction and Motivation

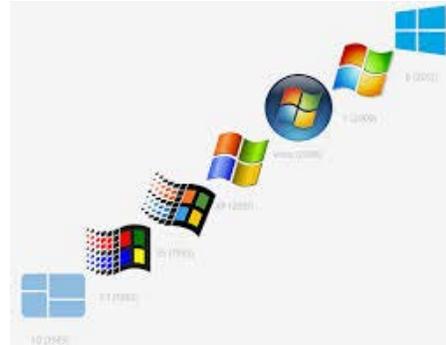
Prof. Dr.-Ing. Jochen Schiller  
Computer Systems & Telematics  
Freie Universität Berlin, Germany

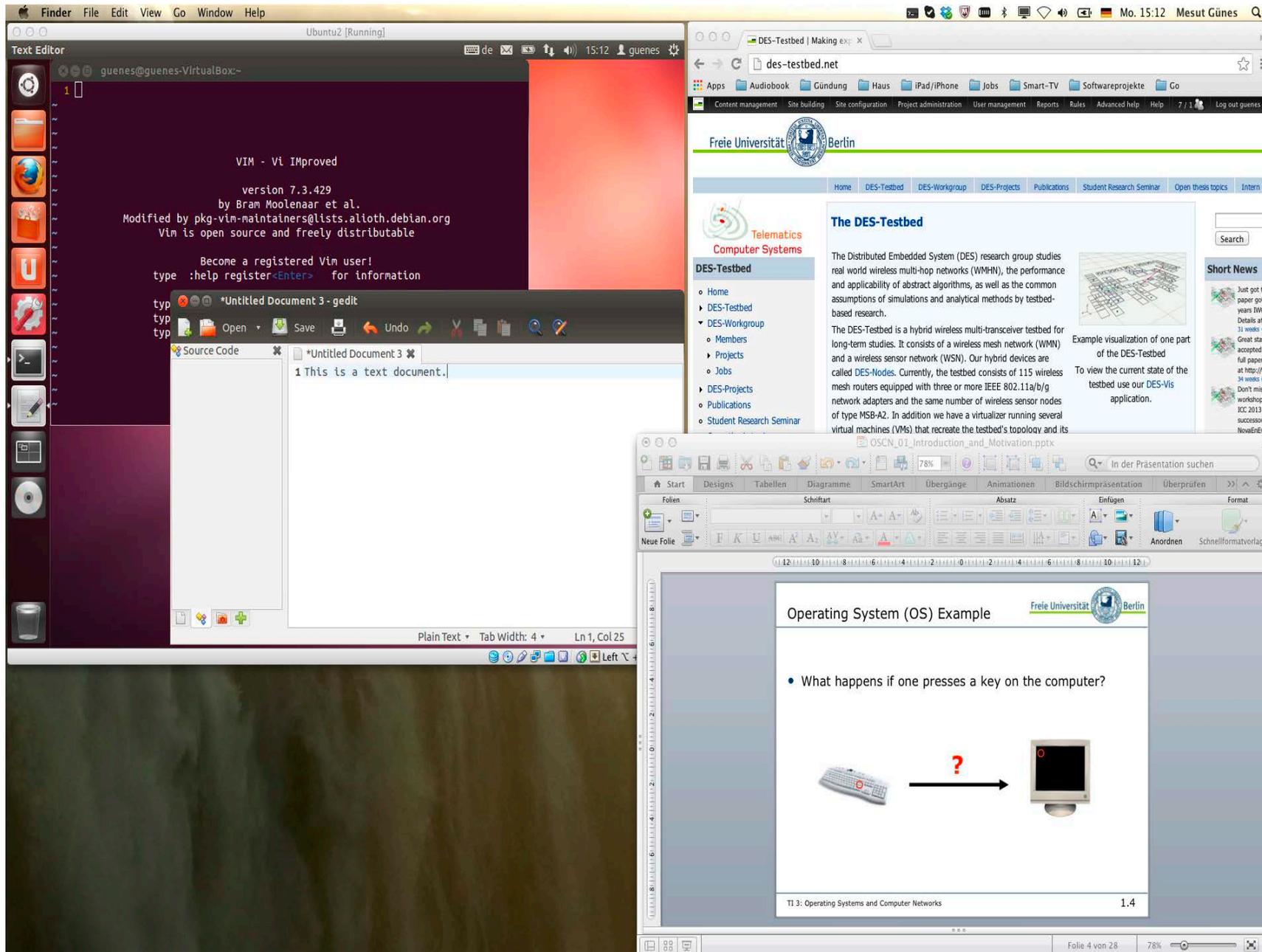


# Content

- 1. Introduction and Motivation**
2. Subsystems, Interrupts and System Calls
3. Processes
4. Memory
5. Scheduling
6. I/O and File System
7. Booting, Services, and Security

# Motivation





The screenshot displays a virtual machine environment with three main windows:

- Text Editor (VIM):** Shows the VIM help screen with the text:
 

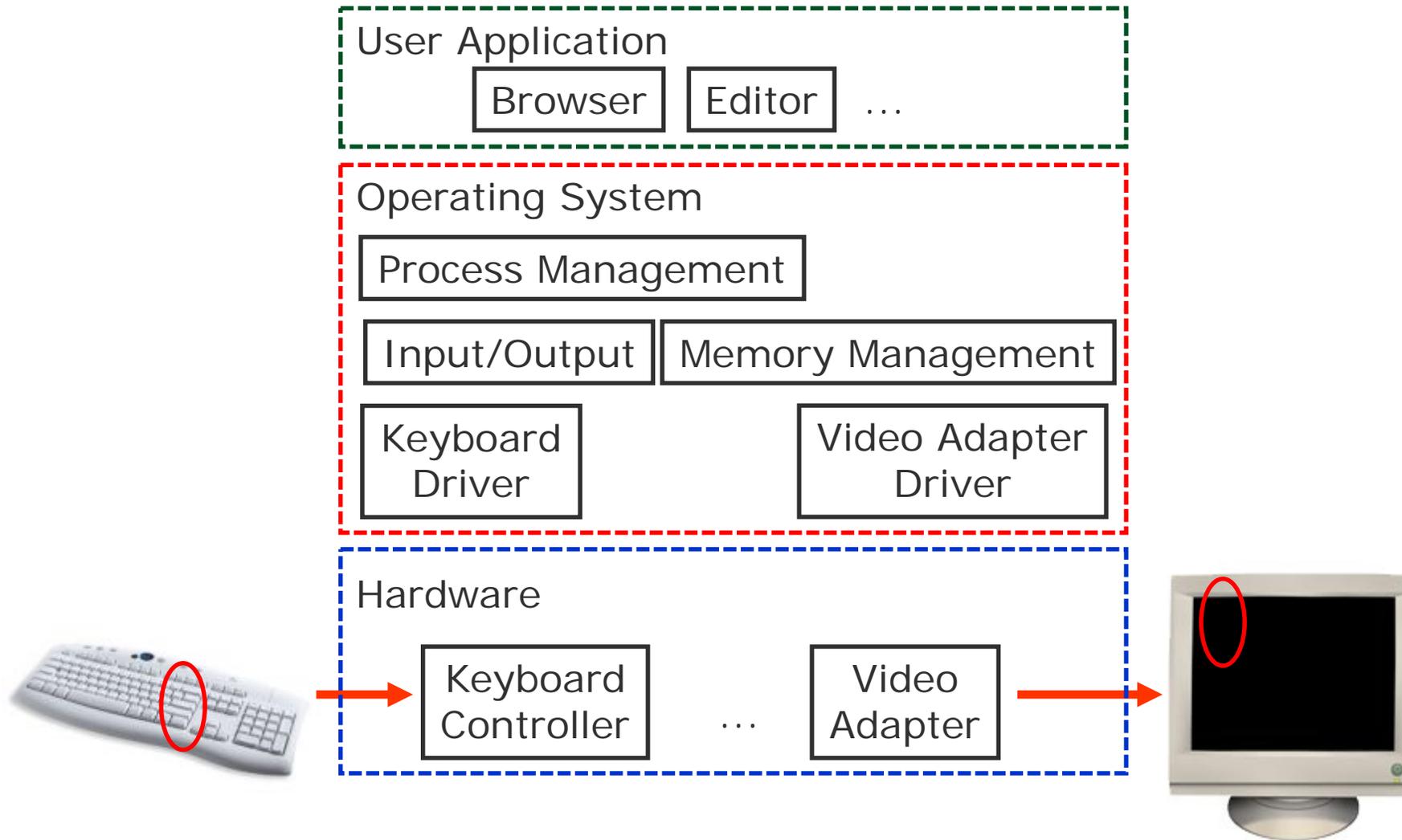
```
VIM - Vi Improved
version 7.3.429
by Bram Moolenaar et al.
Modified by pkg-vim-maintainers@lists.ubuntu.com
Vim is open source and freely distributable
Become a registered Vim user!
type :help register<Enter> for information
```
- Web Browser:** Displays the website for the DES-Testbed at Freie Universität Berlin. The page title is "The DES-Testbed" and it describes the Distributed Embedded System (DES) research group's work on wireless multi-hop networks. It mentions that the testbed consists of 115 wireless mesh routers and 115 wireless sensor nodes.
- Presentation (Beamer):** Shows a slide titled "Operating System (OS) Example" from the "TI 3: Operating Systems and Computer Networks" course. The slide contains a bullet point: "What happens if one presses a key on the computer?". Below the text is a diagram showing a keyboard on the left, an arrow pointing to a computer monitor on the right, with a red question mark above the arrow.

## Operating System (OS) Example

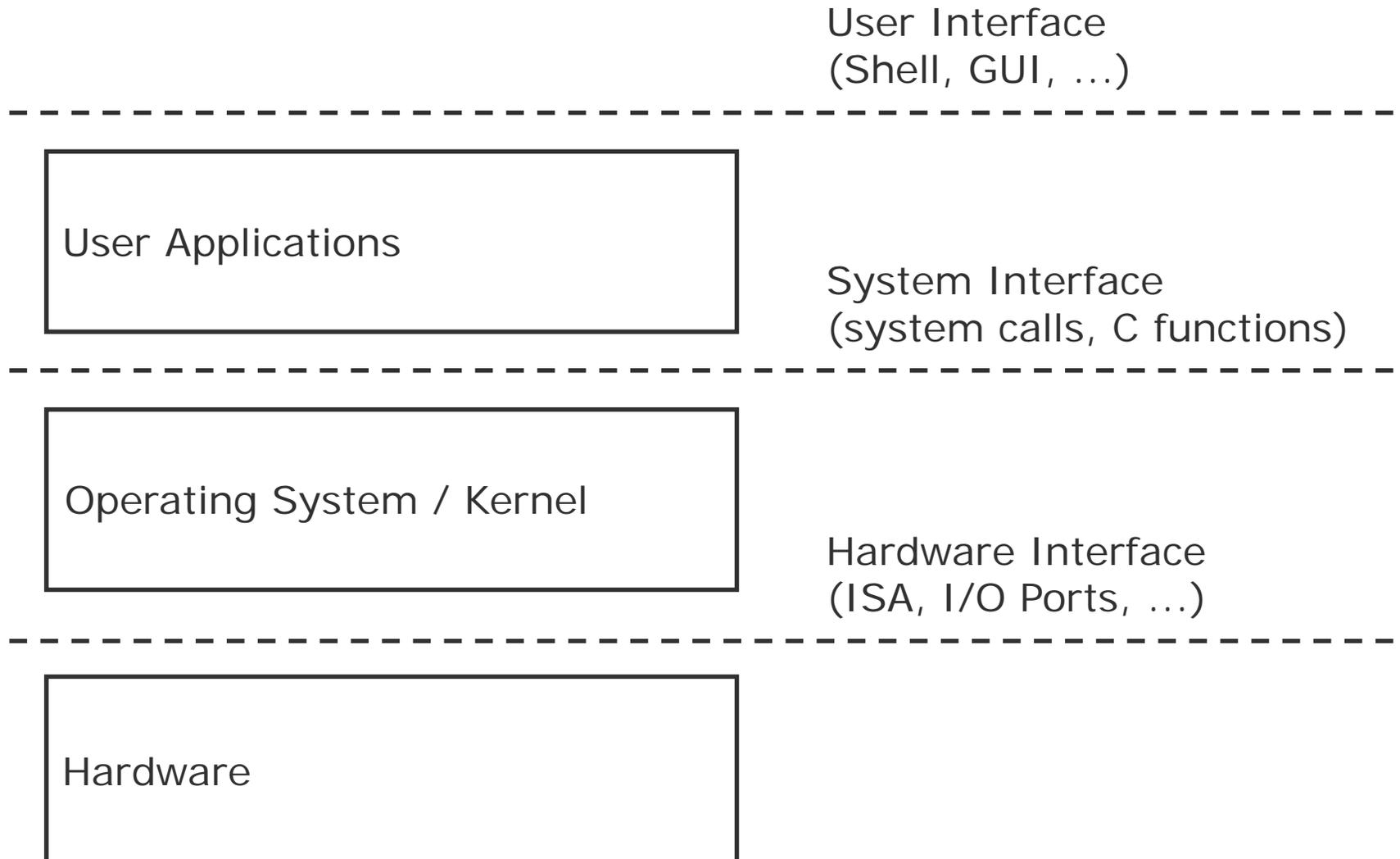
What happens if one presses a key on the computer?



# Operating System Example

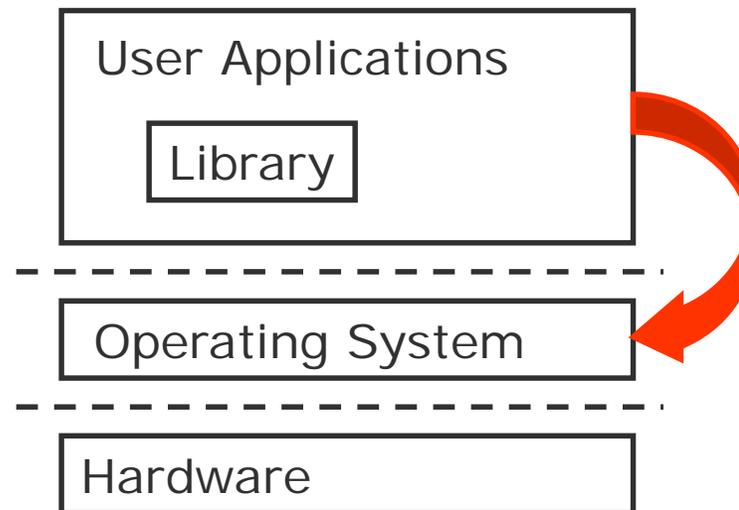


# Layers of Abstraction



## System Interface and 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) → POSIX.



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

# POSIX

Portable Operating System Interface (POSIX)

- E.g. [https://standards.ieee.org/standard/1003\\_1,2013Edition.html](https://standards.ieee.org/standard/1003_1,2013Edition.html)

POSIX defines

- Application programming interface (API)
- Command line shells
- Utilities

UNIX like Operating Systems

POSIX oriented operating systems

- Unix
- Linux
- Windows
- Mac OS X
- ...

## Questions & Tasks

- Why are there so many different operating systems? What are differences?
- What are benefits of layering? Are there any drawbacks?
- Do you know other examples of layering?
- Check out your favorite programming language for system calls. Are there any?
  - That's one of the reasons we use the language C here ...
- Read about POSIX
  - Check IEEE, e.g. [https://standards.ieee.org/standard/1003\\_1,2013Edition.html](https://standards.ieee.org/standard/1003_1,2013Edition.html)
  - or at least Wikipedia...
- Go to RIOT and check out an OS for the IoT
  - <https://riot-os.org/>

# 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

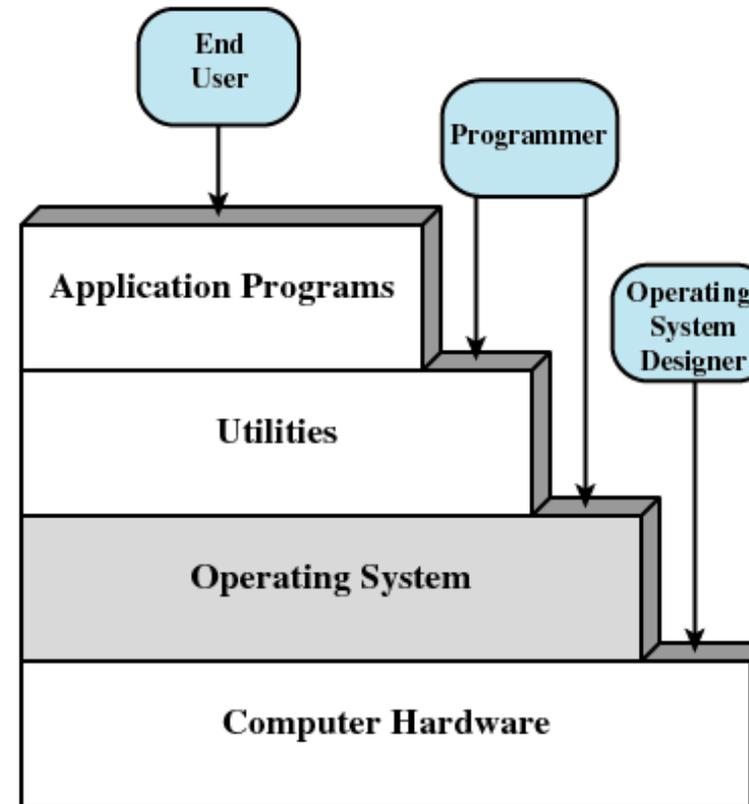
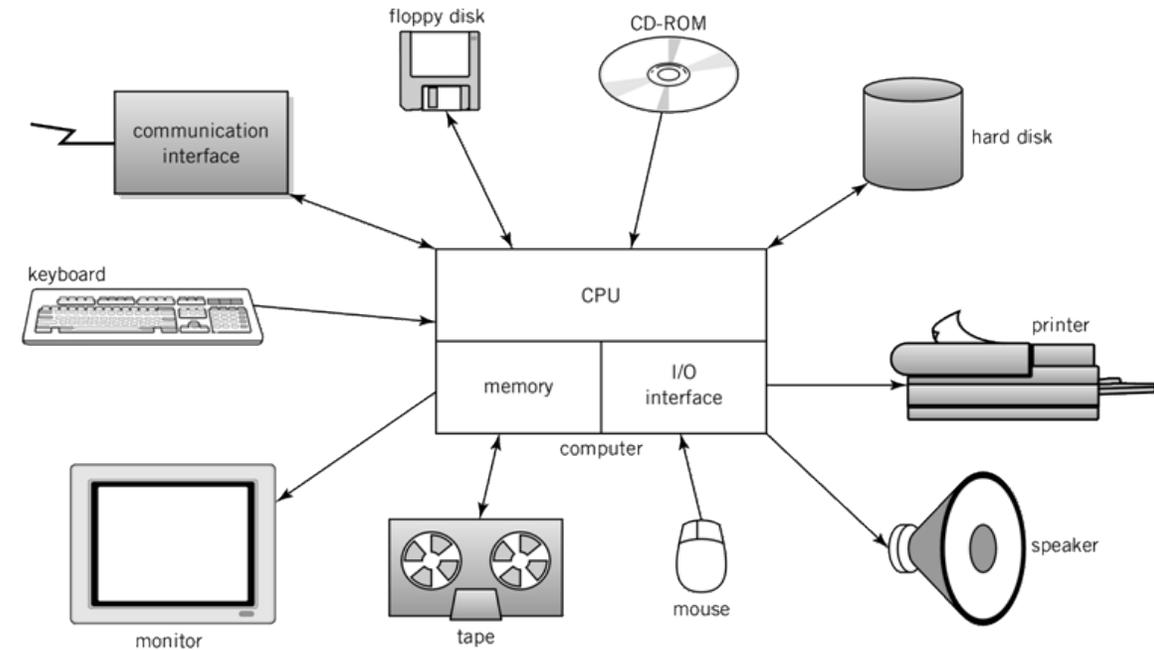


Figure 2.1 Layers and Views of a Computer System

## Managing Resources

Hardware provides the basic computing resources such as

- Processor(s)
- Memory
- Persistent storage
- Network connection



Englander: The Architecture of Computer  
Hardware and Systems Software, 2nd edition  
Chapter 1, Figure 01-06

OS **virtualizes** resources to permit controlled sharing and isolation

- virtual instances of a resource are created

OS provides virtual resources for user applications

# Computer Components

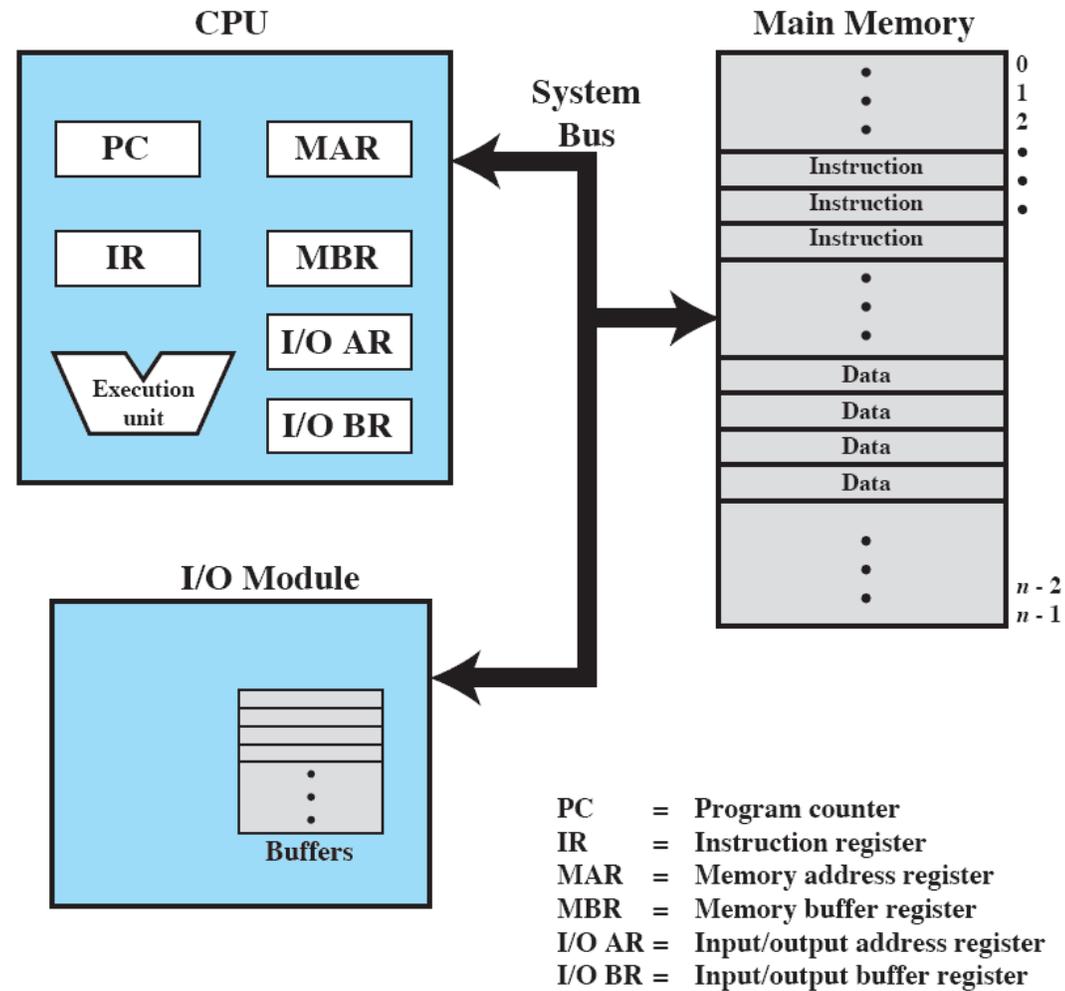


Figure 1.1 Computer Components: Top-Level View

# 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**

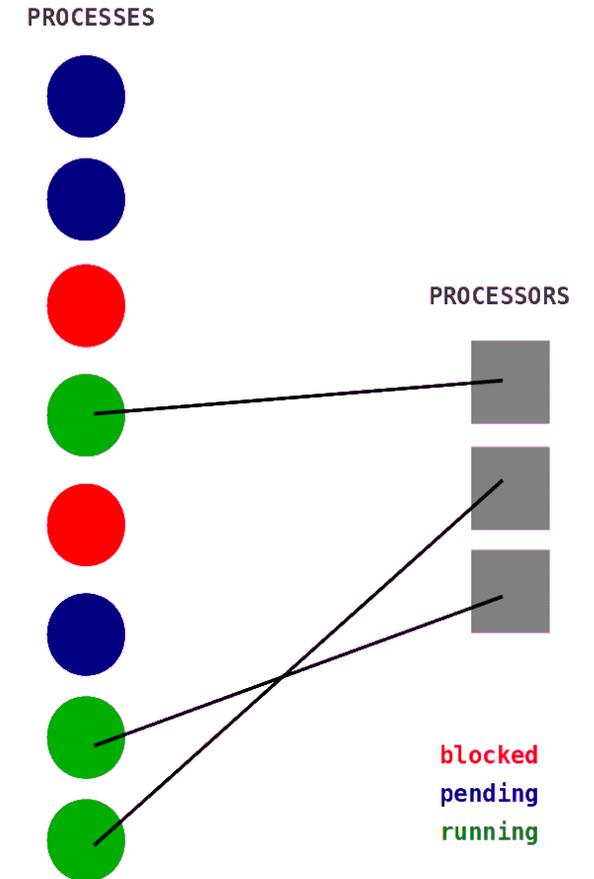
Processor is used efficiently:

- 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:

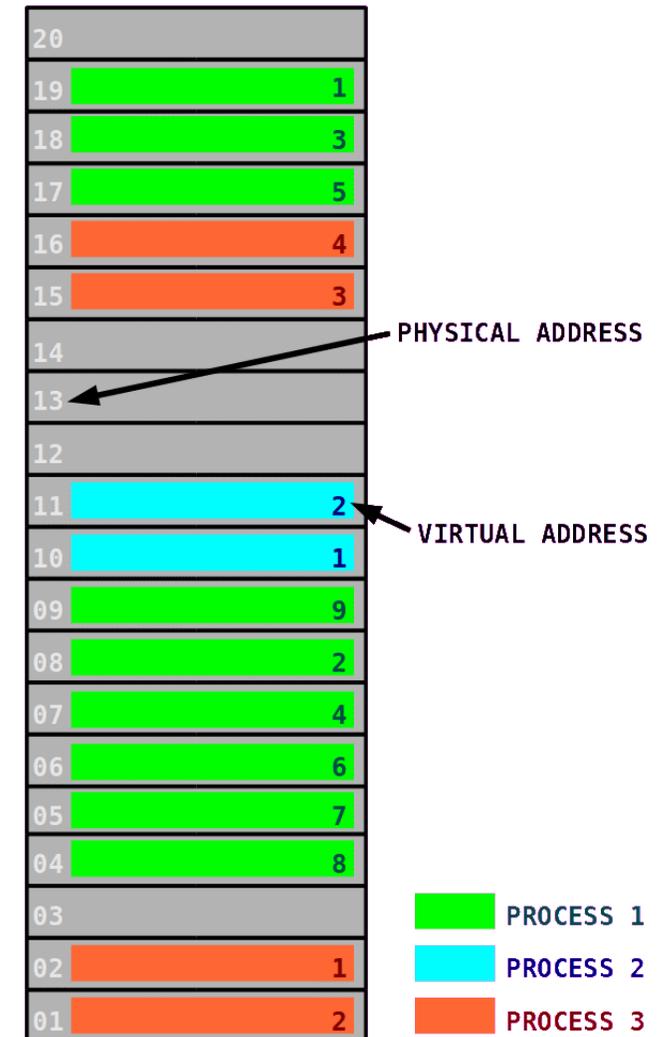
- position independent code – program does not depend on memory architecture

Security:

- memory access is restricted to memory units “owned” by a process

Efficiency:

- external fragmentation is avoided



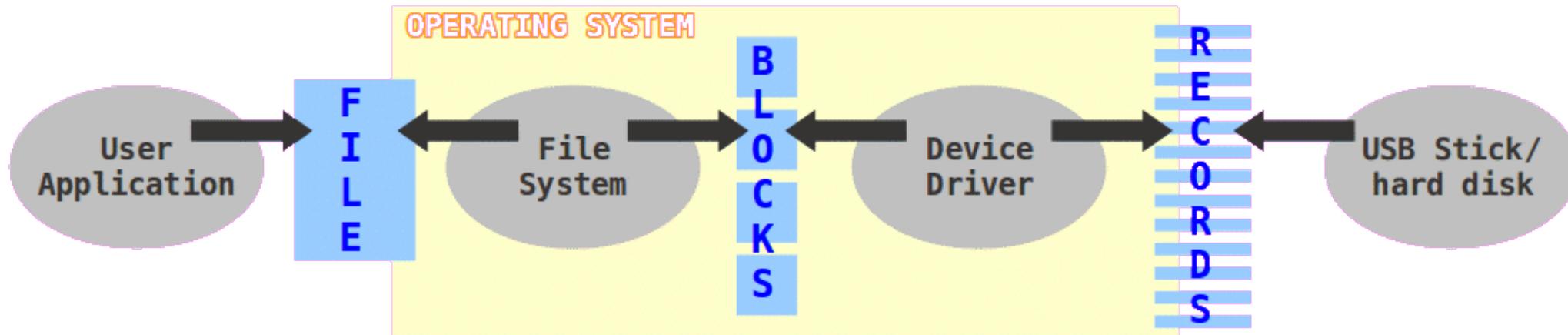
# Files

Managed by a file system

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 address



# History of Operating Systems

Development of operating systems follows changes in computer architecture

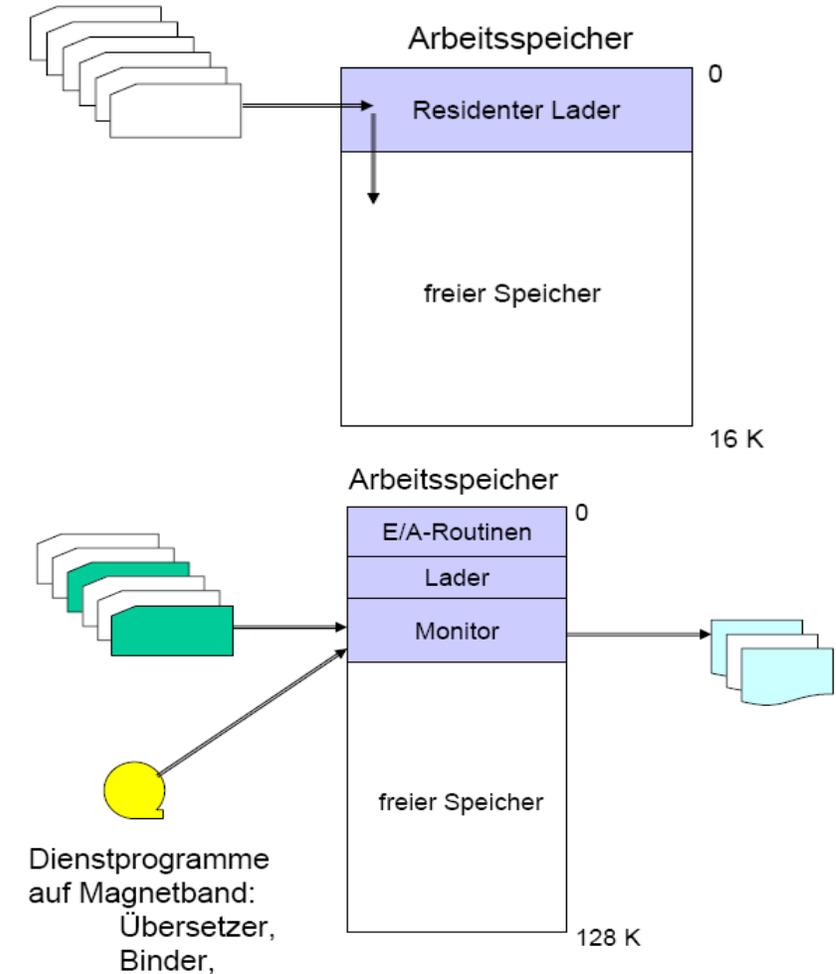
## Loader (1950, IBM 704)

- Loads programs into memory

## Batch System

(1960, IBM 7090, Zuse Z 23, Telefunken TR4)

- Processing of jobs stored on punch cards
- Manual job control by human operator

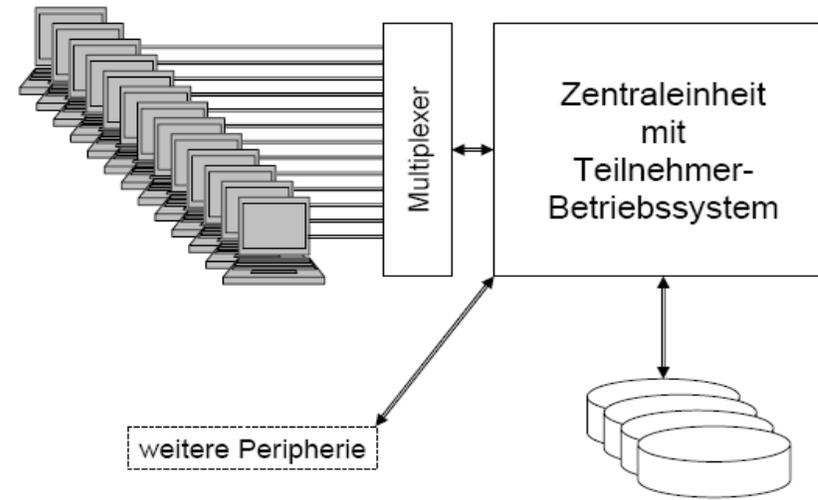


# History of Operating Systems

## Multi-User / Time Sharing Systems

(1970, IBM OS/360, TSS, T.H.E., Multics, UNIX)

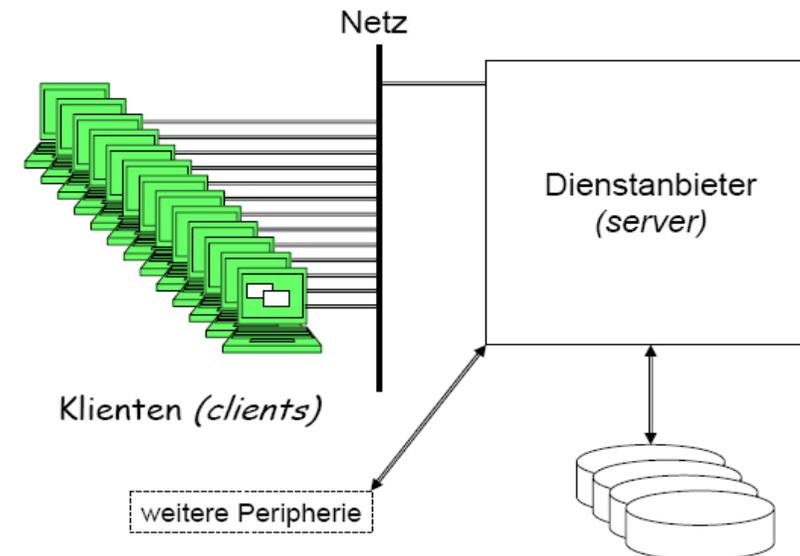
- Many terminals connected to one computer
- Interactive control for users
- Multitasking



## Personal Computing und Client/Server

(1980/90, Apple Lisa, MS Windows, Linux, Solaris, HP-UX)

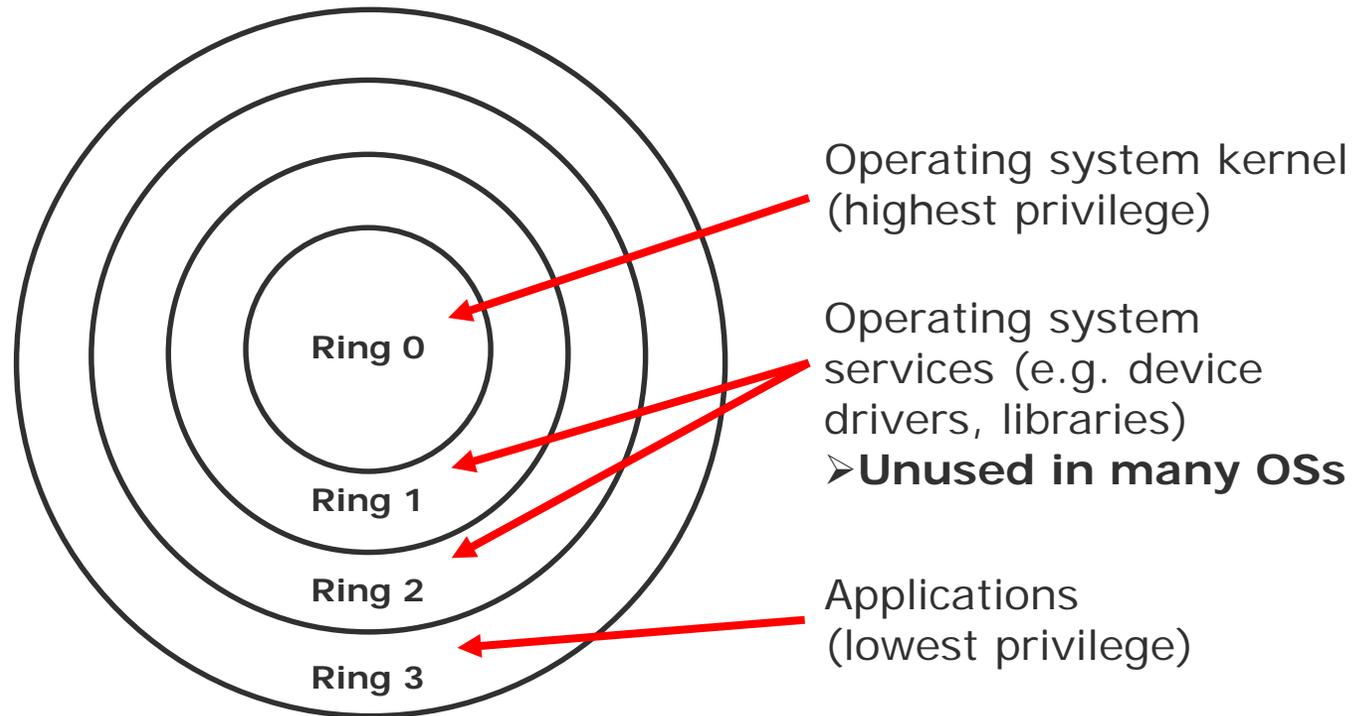
- Intelligent workstations
- GUI / Window mode



# Protection Rings

Hardware provides hierarchical privilege levels

- Inner rings have access to outer rings' resources
- Outer rings may access inner rings through predefined gateways



# Operating System Kernel

Kernel implements basic layer of abstraction

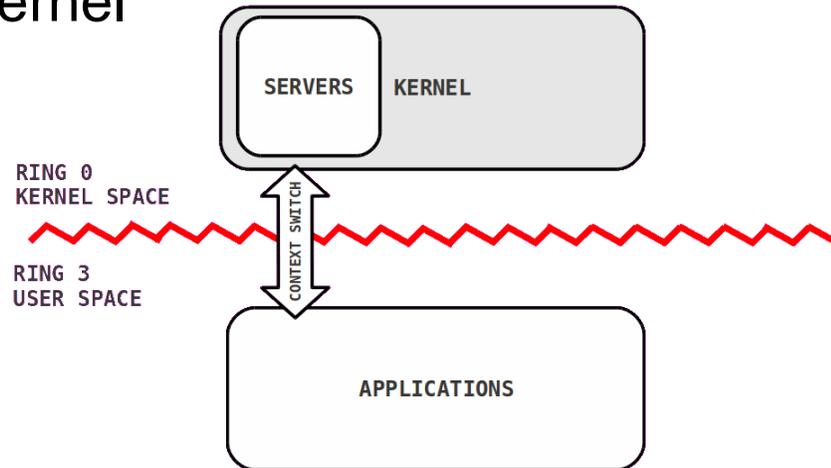
Runs with full access to hardware (Ring 0)

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



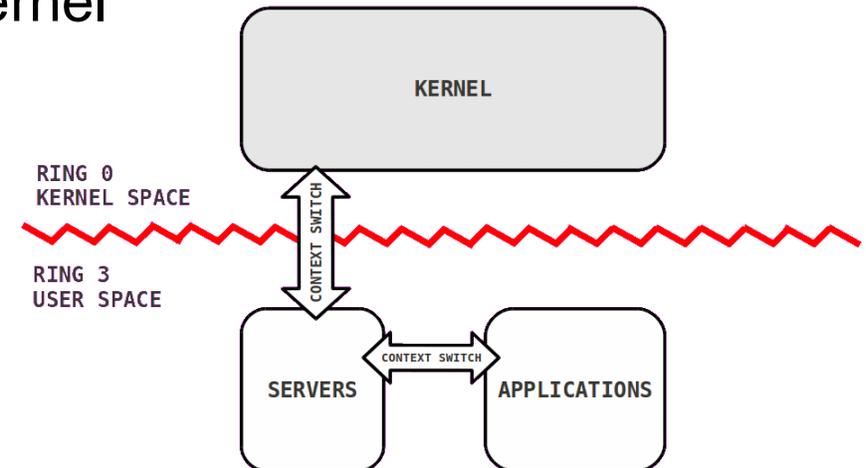
Pro:

- less context switches
- no expensive communication

Contra:

- complications when exchanging functionality

## Microkernel



Pro:

- strict interfaces
- less complexity, clear structure

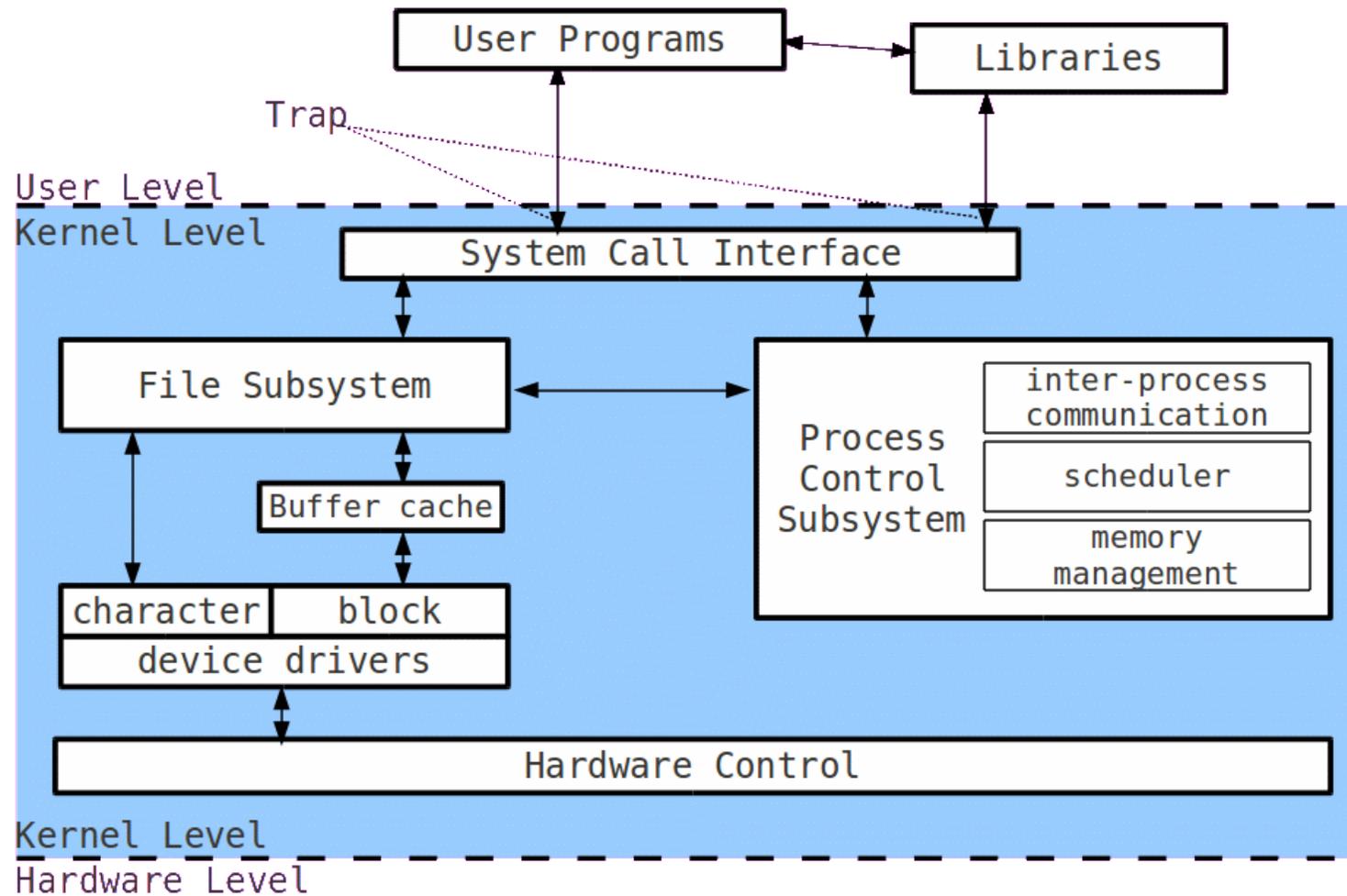
Contra:

- speed
- synchronization

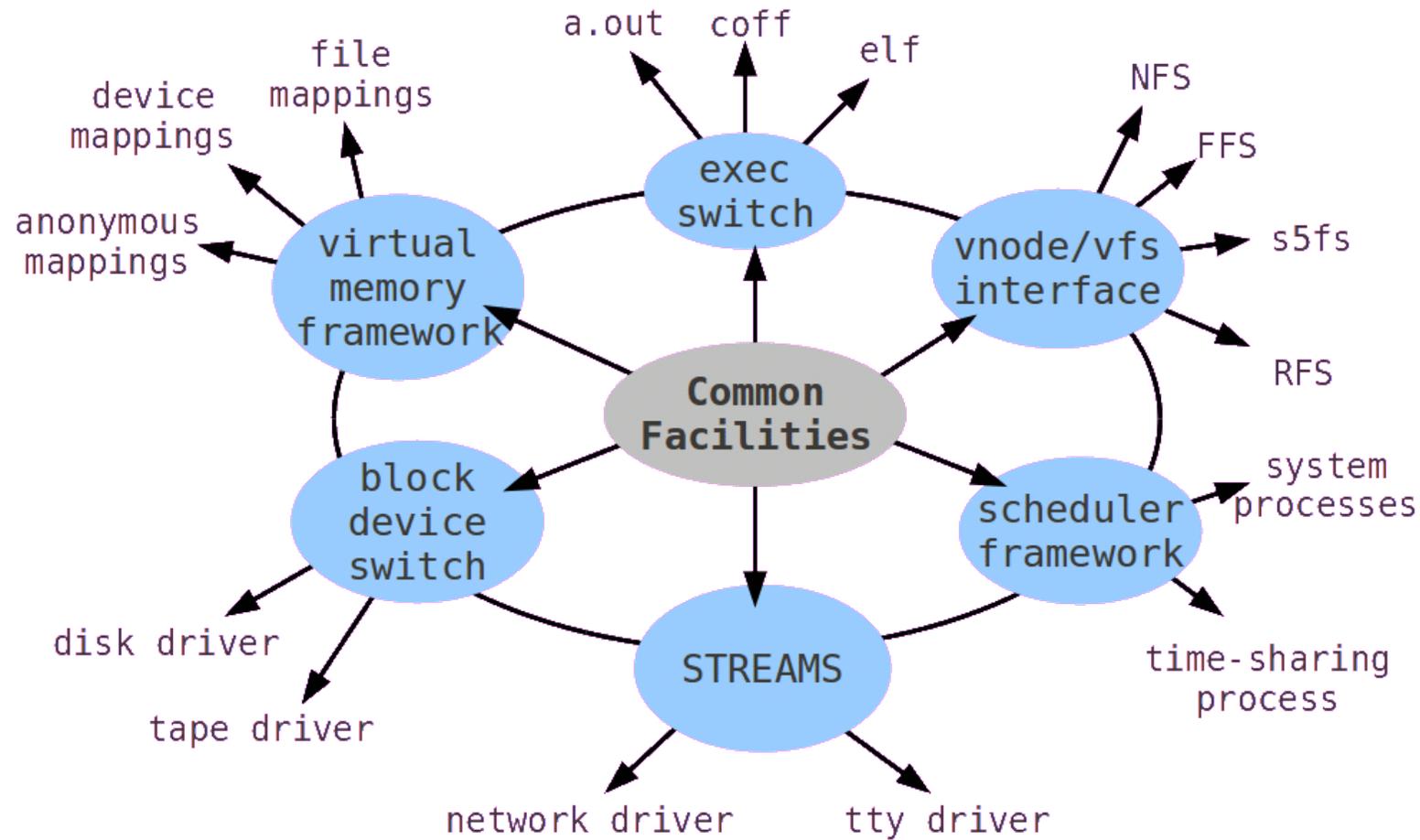
## Questions & Tasks

- What is the basic idea of an OS?
- Can we run a computer without an OS?
- How could a VERY simple OS look like?
- What does virtualization in the context of an OS mean? How does it help? Where else do computers use virtualization?
- Check the number of processes, the resources they use, the memory mapping etc. on your computer. Depending on the OS there are many tools for that!
- What could be reasons for not using that many protection rings?
- What do we need to protect the OS? Think of memory protection, HW support, processor mode etc.
- Do we have pure micro kernel OSs today? Do we at least use ideas from micro kernels? Why are many of the famous OSs today rather big monolithic systems?

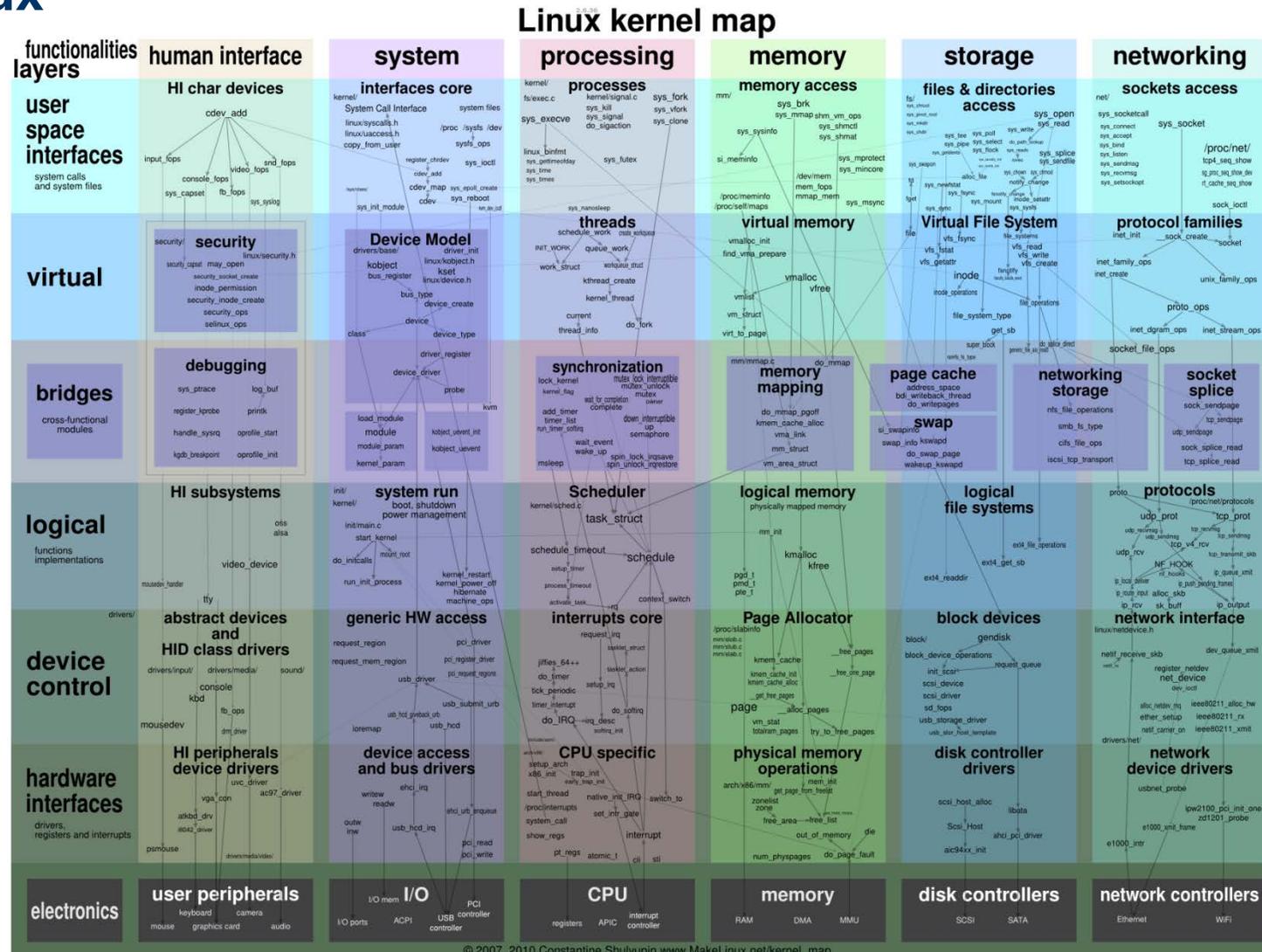
# Examples – UNIX



## Examples – UNIX by Services



# Examples – Linux



Source: [http://www.makelinux.net/kernel\\_map](http://www.makelinux.net/kernel_map)

## A Word About Synchronization

Concurrency handling is outside the scope of this lecture – see “Non-sequential programming”

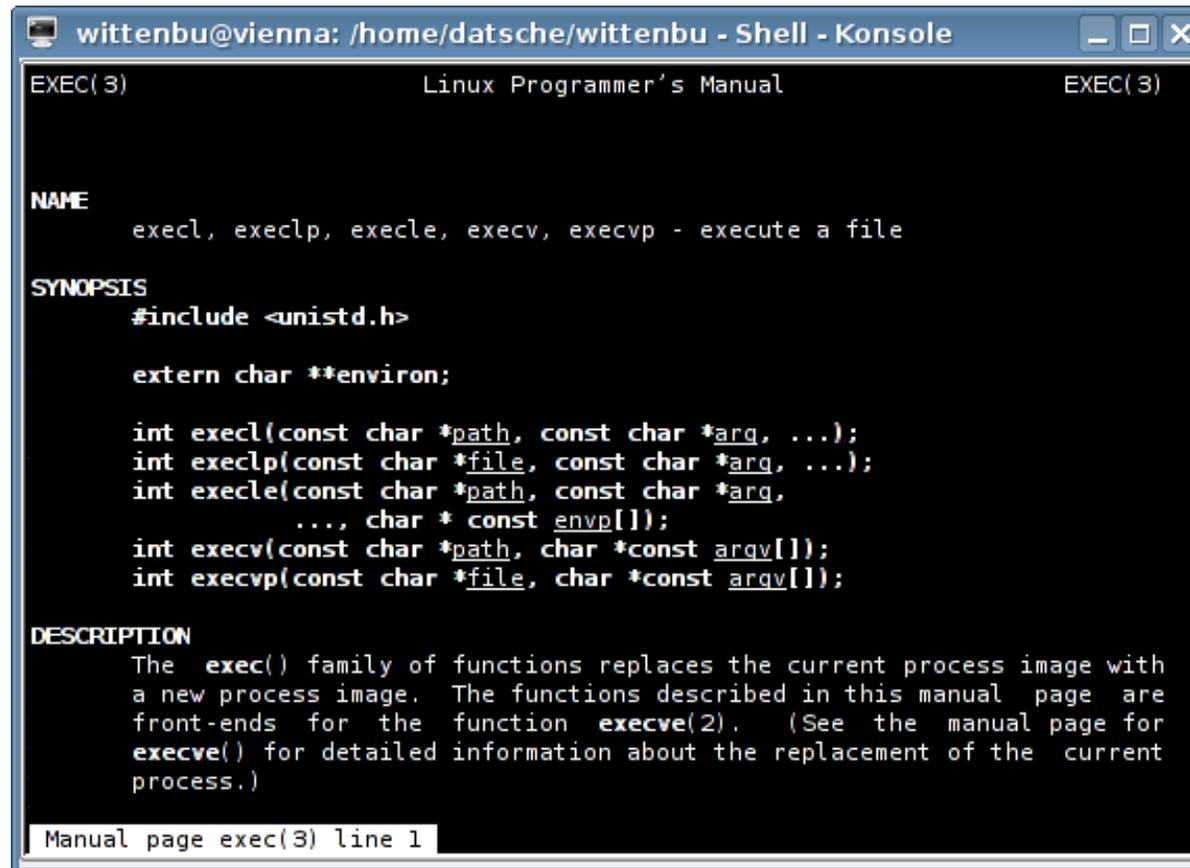
Some pointers/methods/ideas

- 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**



```
wittenbu@vienna: /home/datsche/wittenbu - Shell - Konsole
EXEC(3)                               Linux Programmer's Manual                               EXEC(3)

NAME
    execl, execlp, execl, execv, execvp - execute a file

SYNOPSIS
    #include <unistd.h>

    extern char **environ;

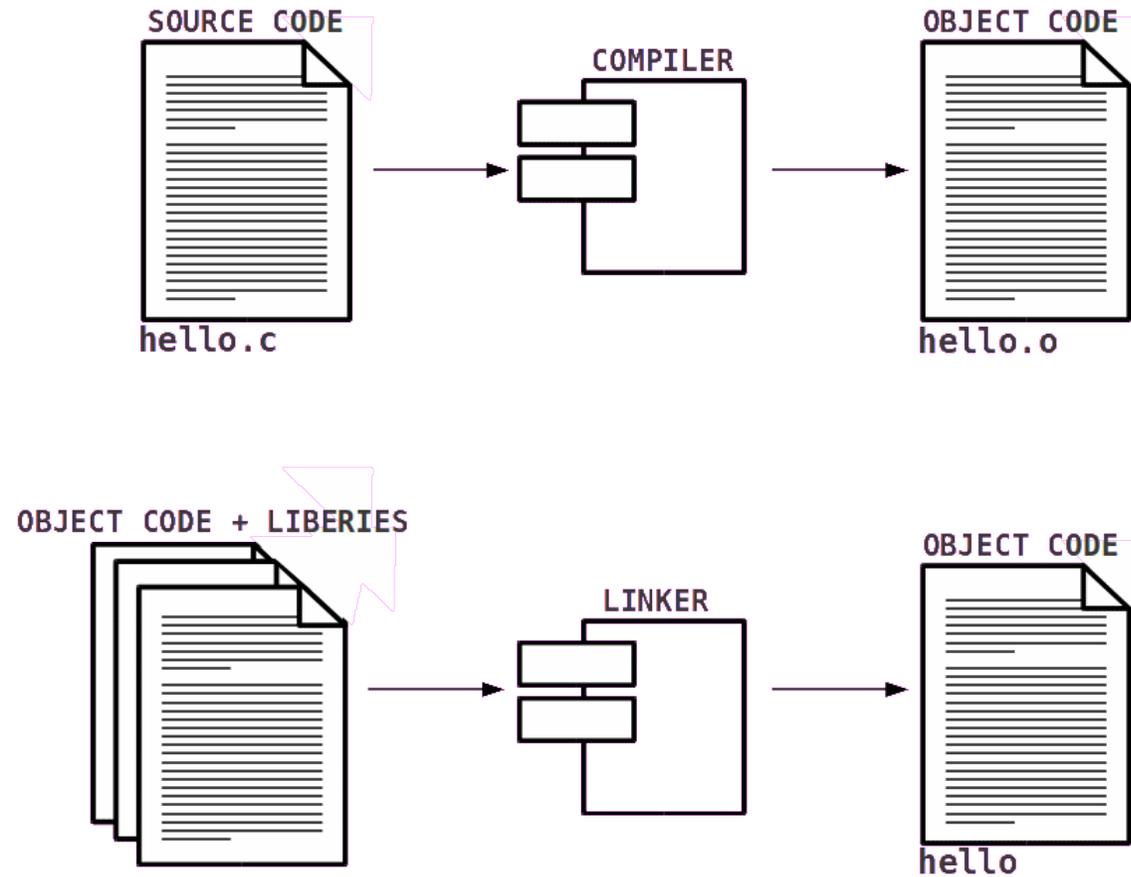
    int execl(const char *path, const char *arg, ...);
    int execlp(const char *file, const char *arg, ...);
    int execl(const char *path, const char *arg,
              ..., char * const envp[]);
    int execv(const char *path, char *const argv[]);
    int execvp(const char *file, char *const argv[]);

DESCRIPTION
    The exec() family of functions replaces the current process image with
    a new process image. The functions described in this manual page are
    front-ends for the function execve(2). (See the manual page for
execve() for detailed information about the replacement of the current
    process.)

Manual page exec(3) line 1
```

# Program Building

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



# Process Monitor

UNIX utility **top**

```
wittenbu@vienna: /home/datsche/wittenbu - Shell - Konsole
top - 10:33:30 up 2 days, 1:04, 1 user, load average: 0.41, 0.26, 0.17
Tasks: 93 total, 1 running, 92 sleeping, 0 stopped, 0 zombie
Cpu(s):  0.3% us,  0.0% sy,  0.0% ni, 99.7% id,  0.0% wa,  0.0% hi,  0.0% si
Mem:  1033264k total,  967528k used,  65736k free,  160112k buffers
Swap: 2015992k total,    0k used, 2015992k free,  445496k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
11843	wittenbu	16	0	1944	968	740	R	0.3	0.1	0:00.54	top
1	root	15	0	1584	520	452	S	0.0	0.1	0:01.44	init
2	root	RT	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
3	root	34	19	0	0	0	S	0.0	0.0	0:00.00	ksoftirqd/0
4	root	10	-5	0	0	0	S	0.0	0.0	0:02.72	events/0
5	root	13	-5	0	0	0	S	0.0	0.0	0:00.02	khelper
6	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	kthread
8	root	10	-5	0	0	0	S	0.0	0.0	0:00.15	kblockd/0
11	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	khubd
13	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	kseriod
104	root	20	0	0	0	0	S	0.0	0.0	0:00.00	pdflush
105	root	15	0	0	0	0	S	0.0	0.0	0:01.00	pdflush
106	root	15	0	0	0	0	S	0.0	0.0	0:00.02	kswapd0
107	root	20	-5	0	0	0	S	0.0	0.0	0:00.00	aio/0
108	root	20	-5	0	0	0	S	0.0	0.0	0:00.00	xfsslogd/0
109	root	20	-5	0	0	0	S	0.0	0.0	0:00.00	xfssdatad/0
764	root	11	-5	0	0	0	S	0.0	0.0	0:00.00	ata/0
781	root	11	-5	0	0	0	S	0.0	0.0	0:00.00	kpsmoused

# Kernel Parameters

## Directories `/proc` and `/sys`

- virtual directories that reflect general kernel behaviors (“everything is a file”)

```
wittenbu@vienna: / - Shell - Konsole
wittenbu@vienna:/$ ls /proc/sys/kernel/
acct                modprobe           panic_on_oops      shmall
bootloader_type    msgmax             pid_max            shmmax
cad_pid            msgmnb             printk              shmmni
cap-bound          msgmni             printk_ratelimit    sysrq
core_pattern       ngroups_max       printk_ratelimit_burst  tainted
core_uses_pid      osrelease          pty                 threads-max
ctrl-alt-del       ostype             random              unknown_nmi_panic
domainname         overflowgid        randomize_va_space    version
hostname           overflowuid        sem
hotplug            panic              sg-big-buff
wittenbu@vienna:/$ ls /proc/sys/vm
block_dump          legacy_va_layout   page-cluster
dirty_background_ratio  lowmem_reserve_ratio  percpu_pagelist_fraction
dirty_expire_centisecs  max_map_count         swap_token_timeout
dirty_ratio           min_free_kbytes       swappiness
dirty_writeback_centisecs  nr_pdflush_threads    vfs_cache_pressure
drop_caches          overcommit_memory
laptop_mode          overcommit_ratio
wittenbu@vienna:/$ ls /proc/sys/fs
aio-max-nr          dir-notify-enable  inode-state         mqueue              suid_dumpable
aio-nr              file-max            inotify              nfs                  xfs
binfmt_misc         file-nr             lease-break-time    overflowgid
dentry-state        inode-nr            leases-enable        overflowuid
wittenbu@vienna:/$
```

# Content

- 1. Introduction and Motivation**
2. Subsystems, Interrupts and System Calls
3. Processes
4. Memory
5. Scheduling
6. I/O and File System
7. Booting, Services, and Security