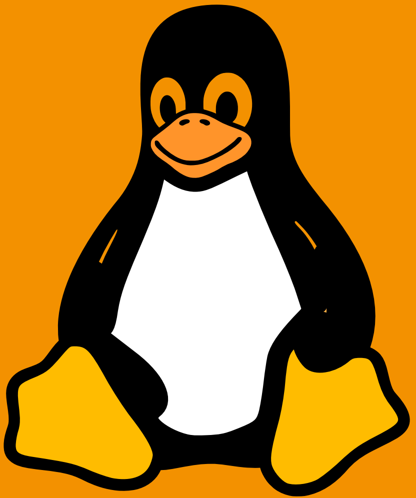
Technische Hochschule Rosenheim
Technical University of Applied Sciences

Prof. Florian Künzner

Technical University of Applied Sciences Rosenheim, Computer Science

OS 9 – Communication 2



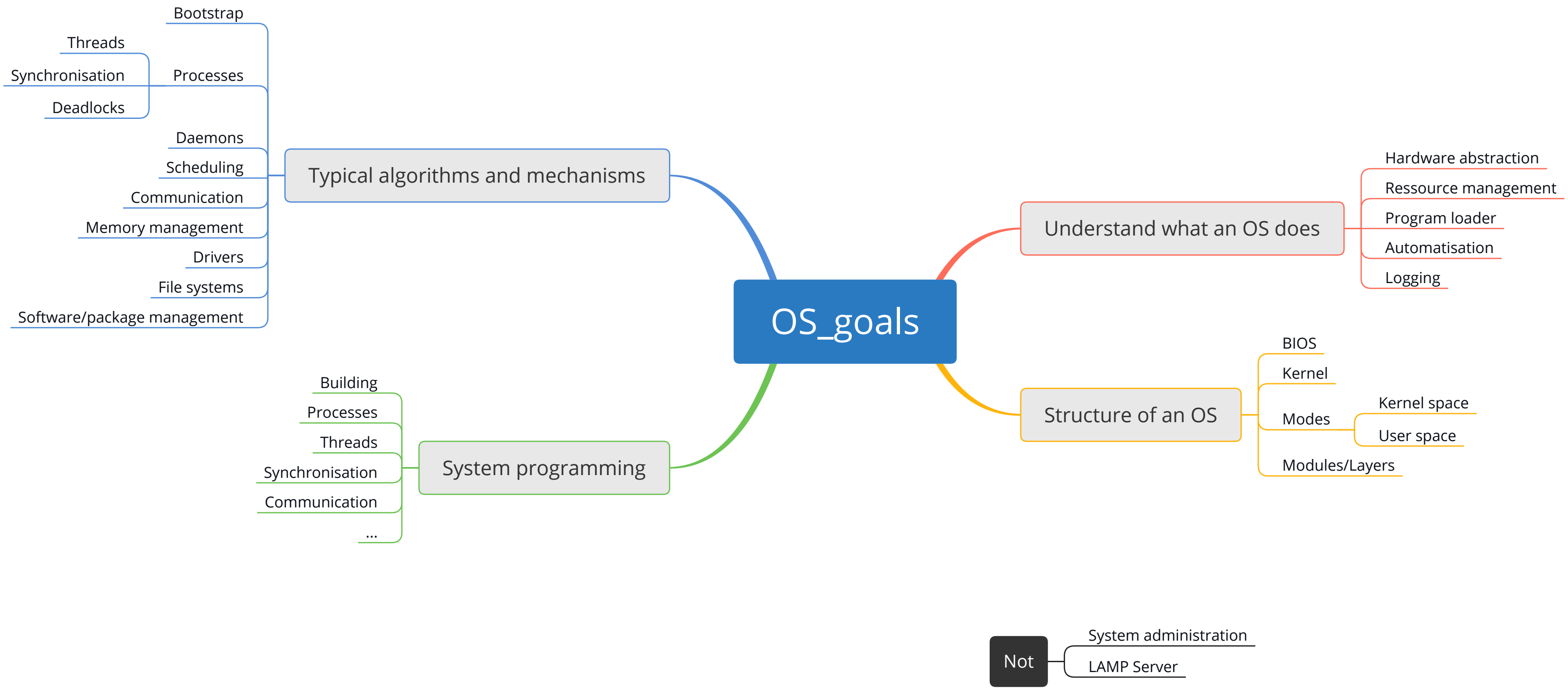
source: iconspng.com

The lecture is based on the work and the documents of Prof. Dr. Ludwig Frank

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Goal



- OS_goals
  - Typical algorithms and mechanisms
    - Bootstrap
    - Processes
      - Threads
      - Synchronisation
      - Deadlocks
    - Daemons
    - Scheduling
    - Communication
    - Memory management
    - Drivers
    - File systems
    - Software/package management
  - System programming
    - Building
    - Processes
    - Threads
    - Synchronisation
    - Communication
    - ...
  - Understand what an OS does
    - Hardware abstraction
    - Ressource management
    - Program loader
    - Automatisation
    - Logging
  - Structure of an OS
    - BIOS
    - Kernel
    - Modes
      - Kernel space
      - User space
    - Modules/Layers
- Not
  - System administration
  - LAMP Server

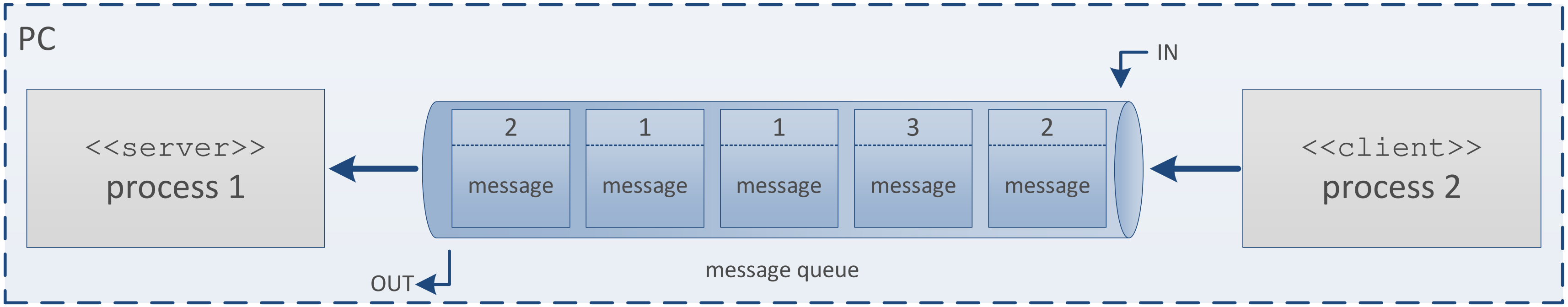
CAMPUS Rosenheim
Computer Science

Goal

OS::Communication

- Message queue
- Shared memory
- Process communication summary

Message queue



CAMPUS Rosenheim
Computer Science

Message queue

Message queue concept

- Queue to **store messages**
- Inter-process communication (IPC) between processes on one PC.
- Messages have **priority/type**.
- Internal stored as a linked list.
- **Send** into queue **does not require** an **active receiver**
- **Read** from queue **does not require** an **active sender**
- Max queue size (default: 16 KiB on Linux).
- Max message size (default: 8 KiB on Linux).

CAMPUS Rosenheim
Computer Science

Message queue

Message structure

```
struct message {
    long priority;    //priority or type
    char message[64]; //buffer for message bytes
};
```

Message priority/type

- The lower the number, the higher the priority
- The priority can be interpreted as a type (each type has its own number)

Message queue usage scenarios (`PRIO_FETCH_FLAG`)

- Read message after message (FIFO principle) (`priority == 0`)
- Read only message with specific priority/type (`priority == N`)
- Read the messages with the highest priority/type first, up to a certain number (`priority == -N`)

Message queue: Pseudo C code

```
struct message { //structure for messages
  long priority;
  char message[64];
};

void receiver() {
  //create message queue
  msgget(...);



  //receive message
  //blocks if message queue is empty
  msgrcv(...);

  //... work with message

  //remove message queue
  msgctl(...);
}
```

```
void sender() {
  //open existing message queue
  msgget(...);

  //prepare message

  //send message
  //blocks if message queue is full
  msgsnd(...);

  //close not needed



}
```

Message queue: Linux commands

Command	Description
`ipcs`	Show information on IPC facilities
`ipcs -q`	Shows active message queues in the system
`ipcmk`	Make various IPC resources
`ipcmk -Q`	Create a message queue
`ipcrm`	Remove certain IPC resources
`ipcrm -q 1`	Remove message queue with id 1
`ipcrm -Q 2`	Remove message queue with key 2

CAMPUS Rosenheim
Computer Science

Message queue

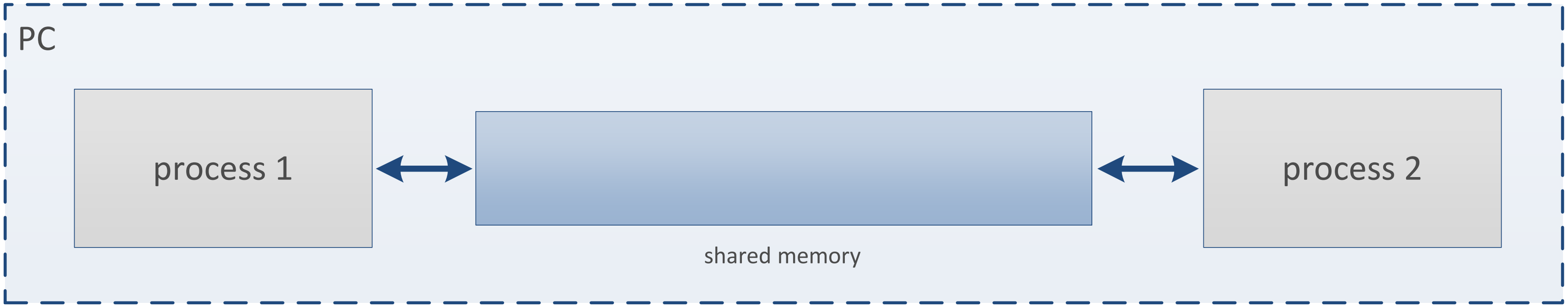
C example

*Please find the source file(s) in the repository.

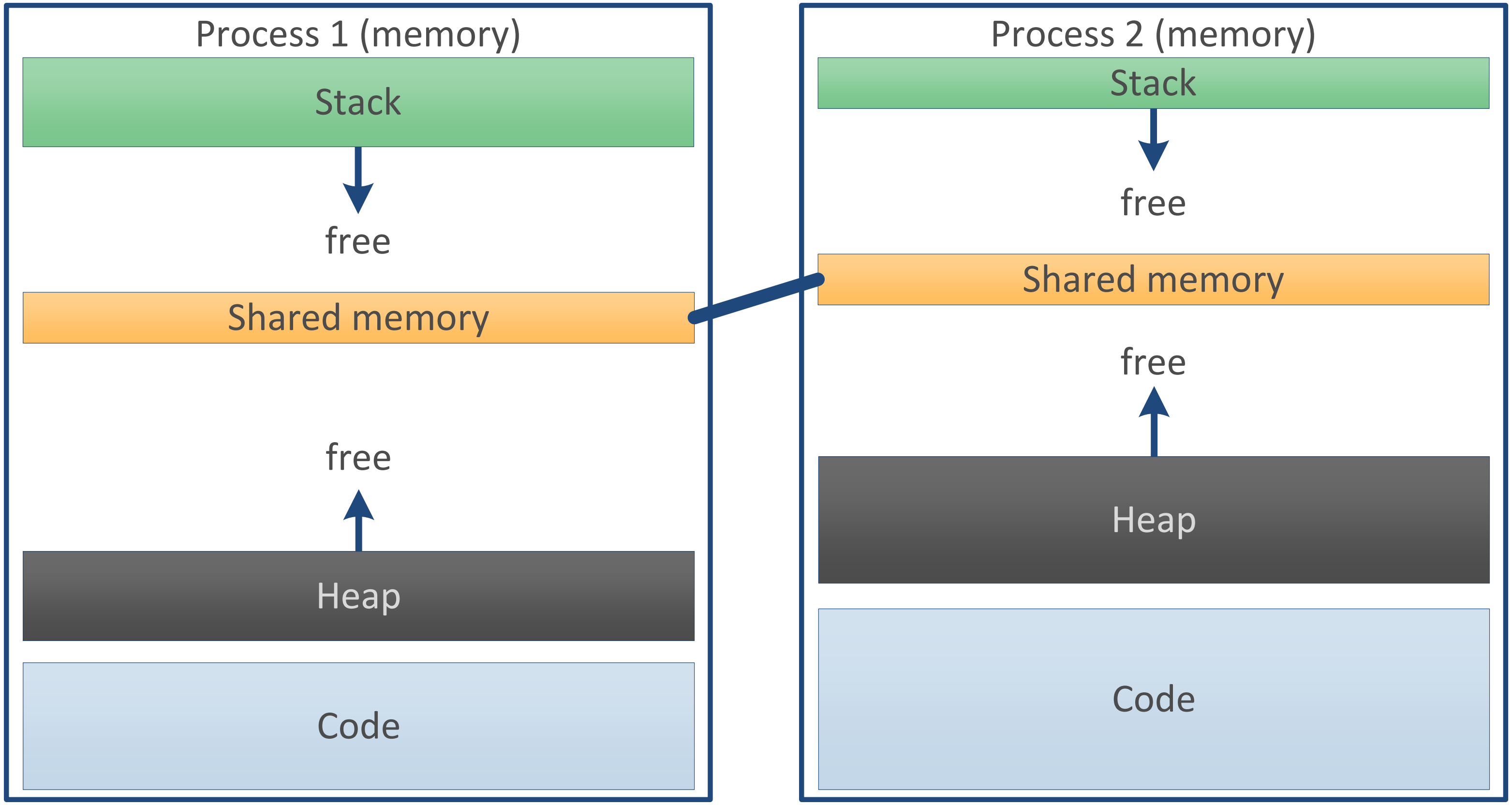
CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Shared memory



Shared memory



CAMPUS Rosenheim
Computer Science

Shared memory

Shared memory concept

- Shared memory area between processes
- Inter-process communication (IPC) between processes on one PC.
- It is a **plain memory area** with a certain size
- Access **needs to be synchronised** (e.g. semaphore)
- Access is **very fast** (comparable with own memory access)

Shared memory: Pseudo C code

```
seminit(READY_TO_WRITE, 1); //declare and initialise semaphore
seminit(READY_TO_READ,  0); //declare and initialise semaphore

void receiver() {
  //create shared memory
  shmget(...);
  //attach the shared memory
  shared_mem_address = shmat(...);




  //copy data from shared memory
  P(READY_TO_READ);
  copy(data, shared_mem_address); //data = sm
  V(READY_TO_WRITE);

  //... work with data
  work_with(data);

  //detach shared memory
  shmdt(...);
  //remove shared memory
  shmctl(...);
}
```

```
void sender() {
  //get existing shared memory
  shmget(...);
  //attach the shared memory
  shared_mem_address = shmat(...);

  //... prepare data
  data = prepare_data();

  //copy data into shared memory
  P(READY_TO_WRITE);
  copy(shared_mem_address, data); //sm = data
  V(READY_TO_READ);




  //detach shared memory
  shmdt(...);



}
```

Shared memory: Linux commands

Command	Description
`ipcs`	Show information on IPC facilities
`ipcs -m`	Shows active shared memory in the system
`ipcmk`	Make various IPC resources
`ipcmk -M 8`	Create a shared memory with 8 bytes
`ipcrm`	Remove certain IPC resources
`ipcrm -m 1`	Remove shared memory with id 1
`ipcrm -M 2`	Remove shared memory with key 2

CAMPUS Rosenheim
Computer Science

Shared memory

C example

*Please find the source file(s) in the repository.

Process communication summary

Mechanism	data	store	access contr.	remote	bidirect.	fast	prio.	sync req.
signal						X		
unix socket	X		X		X	X		
network socket	X			X	X			
pipe	X	_x_	X			X		
message queue	X	X	X			X	X	
shared memory	XX	XX	X		X	XX		X

Comparison of different communication mechanism: `https://www.programering.com/a/MTO0AzMwATI.html`

CAMPUS Rosenheim
Computer Science

Summary and outlook

Summary

- Message queue
- Shared memory
- Process communication summary

Outlook

- Deadlocks
- Deadlock analysis
- Deadlock prevention