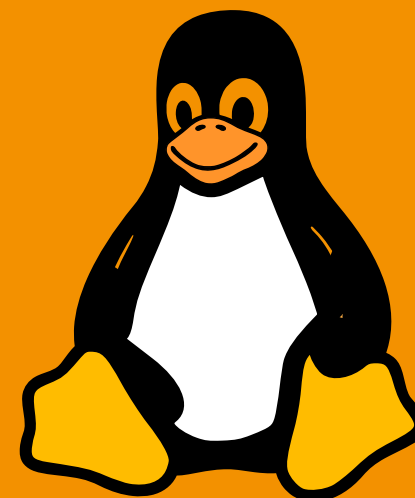




# Prof. Dr. Florian Künzner

Technical University of Applied Sciences Rosenheim, Computer Science

## OS 9 – Communication 2

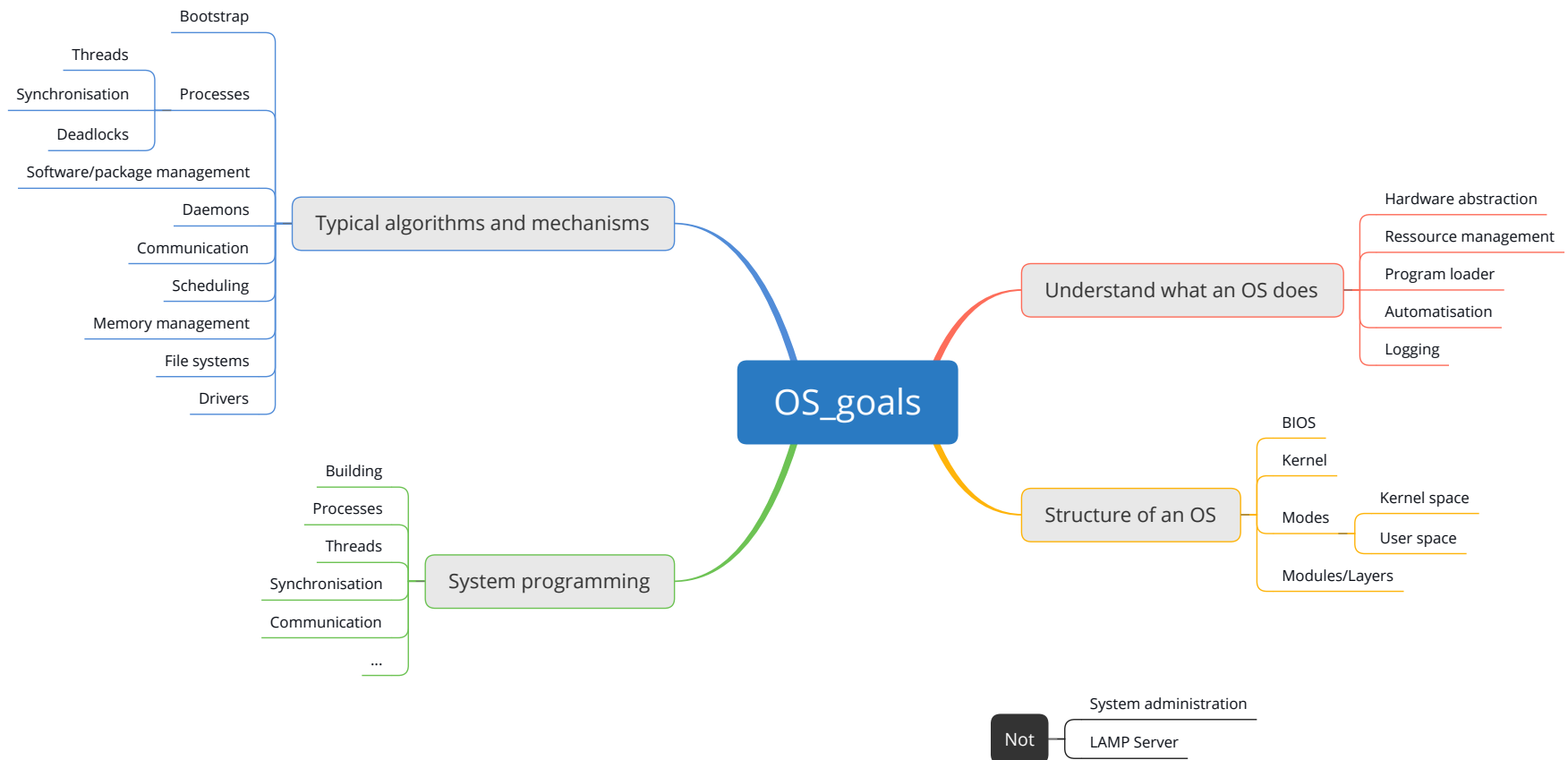


source: [icons.png.com](https://icons.png.com)

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



# Goal

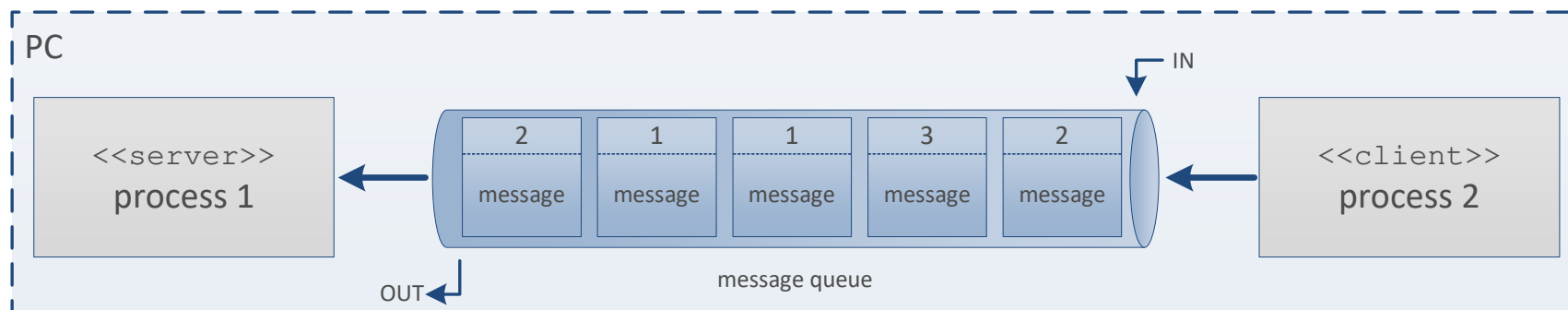


# Goal

## OS::Communication

- Message queue
- Shared memory
- Process communication summary

# Message queue



# 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).

# Message queue

## Message structure

```
1 struct message {  
2     long priority;    //priority or type  
3     char message[64]; //buffer for message bytes  
4 };
```

## 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

```
1 struct message { //structure for messages
2     long priority;
3     char message[64];
4 };
```

```
5 void receiver() {
6     //create message queue
7     msgget(...);
8
9
10
11     //receive message
12     //blocks if message queue is empty
13     msgrcv(...);
14
15     //... work with message
16
17     //remove message queue
18     msgctl(...);
19 }
```

```
20 void sender() {
21     //open existing message queue
22     msgget(...);
23
24     //prepare message
25
26     //send message
27     //blocks if message queue is full
28     msgsnd(...);
29
30     //close not needed
31
32
33
34 }
```



# Message queue: Linux commands

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



# Message queue

## C example

---

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

# Questions?

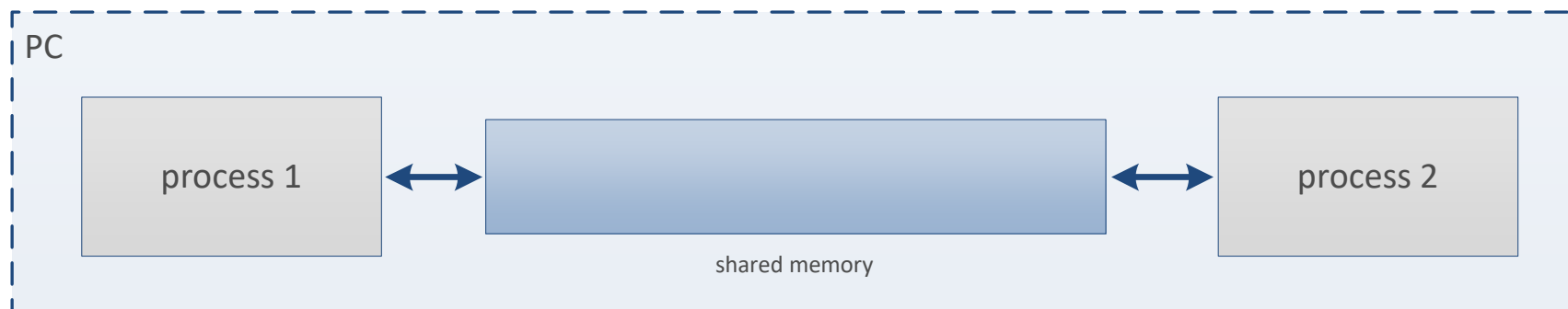
All right?  $\Rightarrow$  

Question?  $\Rightarrow$   and use **chat**

or

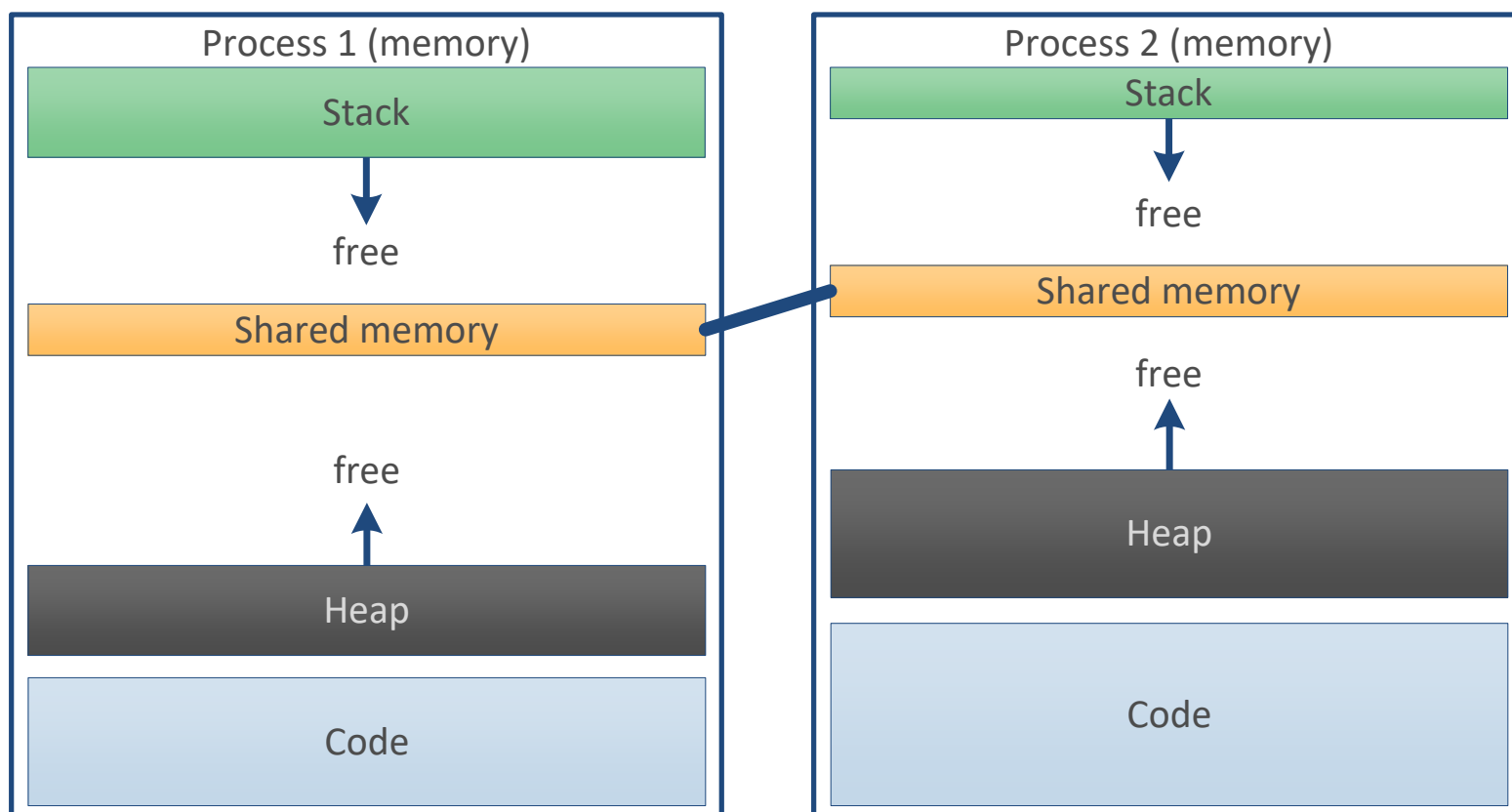
**speak** *after* I  
ask you to

# Shared memory





# Shared memory



# 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

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

3 void receiver() {
4     //create shared memory
5     shmget(...);
6     //attach the shared memory
7     shared_mem_address = shmat(...);
8
9
10
11
12     //copy data from shared memory
13     P(READY_TO_READ);
14     copy(data, shared_mem_address); //data = sm
15     V(READY_TO_WRITE);
16
17     //... work with data
18     work_with(data);
19
20     //detach shared memory
21     shmdt(...);
22     //remove shared memory
23     shmctl(...);
24 }

25 void sender() {
26     //get existing shared memory
27     shmget(...);
28     //attach the shared memory
29     shared_mem_address = shmat(...);
30
31     //... prepare data
32     data = prepare_data();
33
34     //copy data into shared memory
35     P(READY_TO_WRITE);
36     copy(shared_mem_address, data); //sm = data
37     V(READY_TO_READ);
38
39
40
41
42     //detach shared memory
43     shmdt(...);
44
45
46 }
```



# Shared memory: Linux commands

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

# Shared memory

## C example

---

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



# Questions?

All right?  $\Rightarrow$  

Question?  $\Rightarrow$   and use **chat**

or

**speak** *after* I  
ask you to

# 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			
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/MT00AzMwATI.html>

# Questions?

All right?



Question?



and use **chat**

or

**speak** *after* I  
ask you to

# Summary and outlook

## Summary

- Message queue
- Shared memory
- Process communication summary

## Outlook

- Deadlocks
- Deadlock analysis
- Deadlock prevention