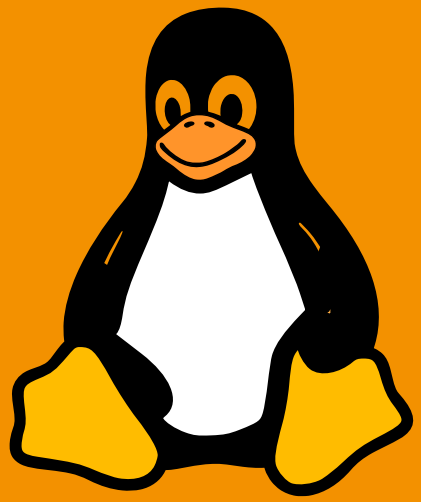
Technische Hochschule Rosenheim
Technical University of Applied Sciences

Prof. Florian Künzner
Technical University of Applied Sciences Rosenheim, Computer Science

OS 5 – Process/Thread



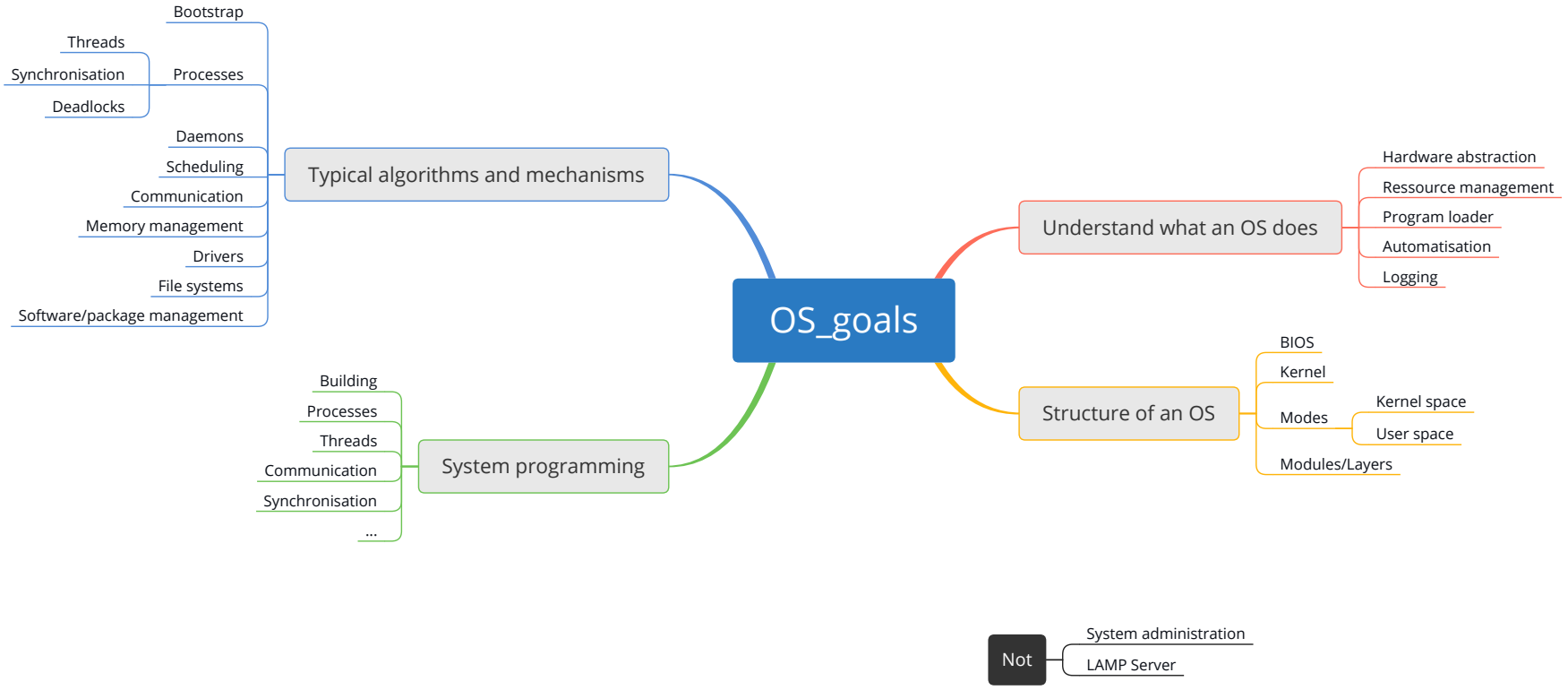
source: iconspng.com

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



CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Goal

OS::Process/Thread

- What is a process/thread
- Process hierarchy
- Processes management
- Thread management
- Parallelisation

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Intro

What is a process?

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

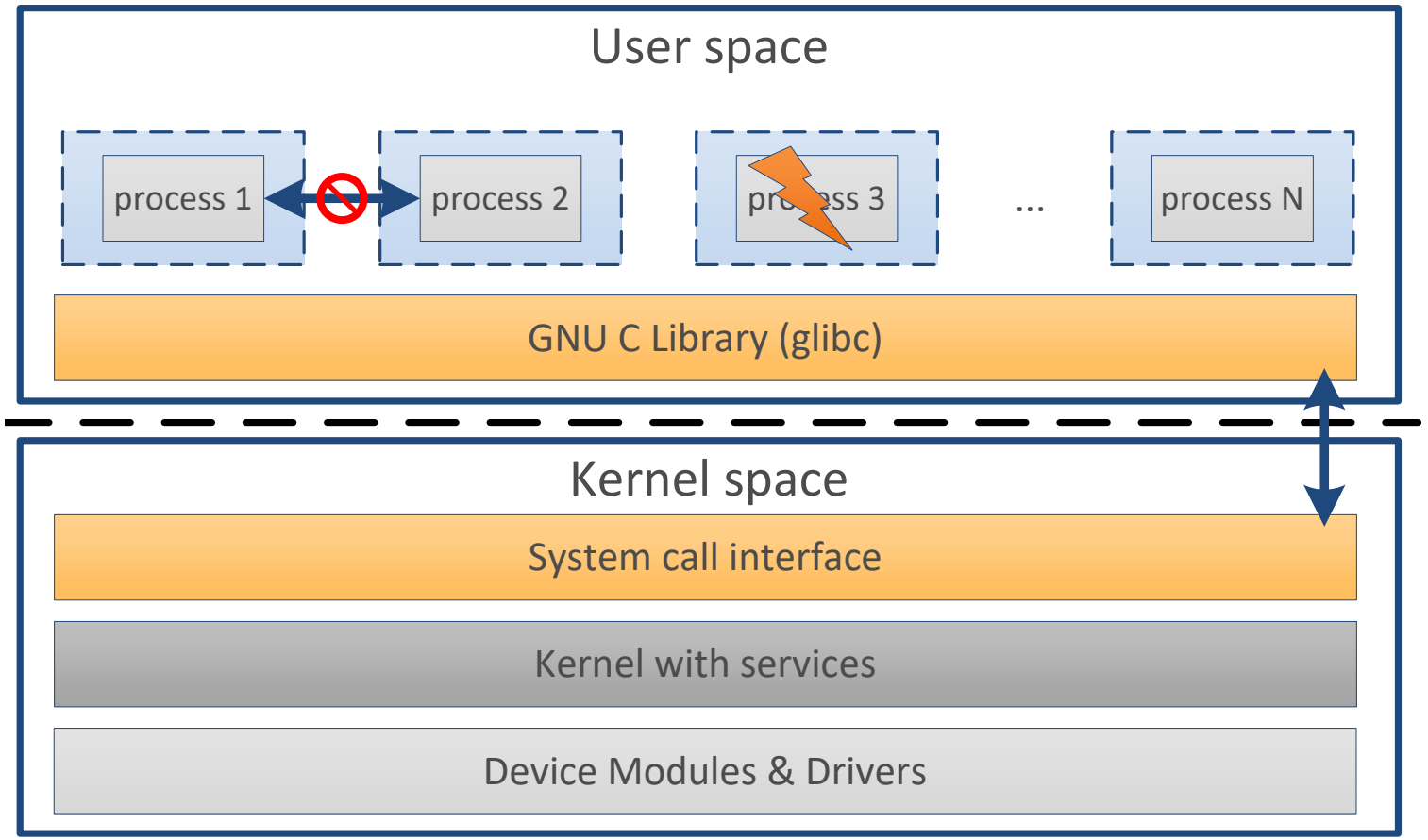
Process definition

A process is an **instance of a computer program** that is being executed.

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

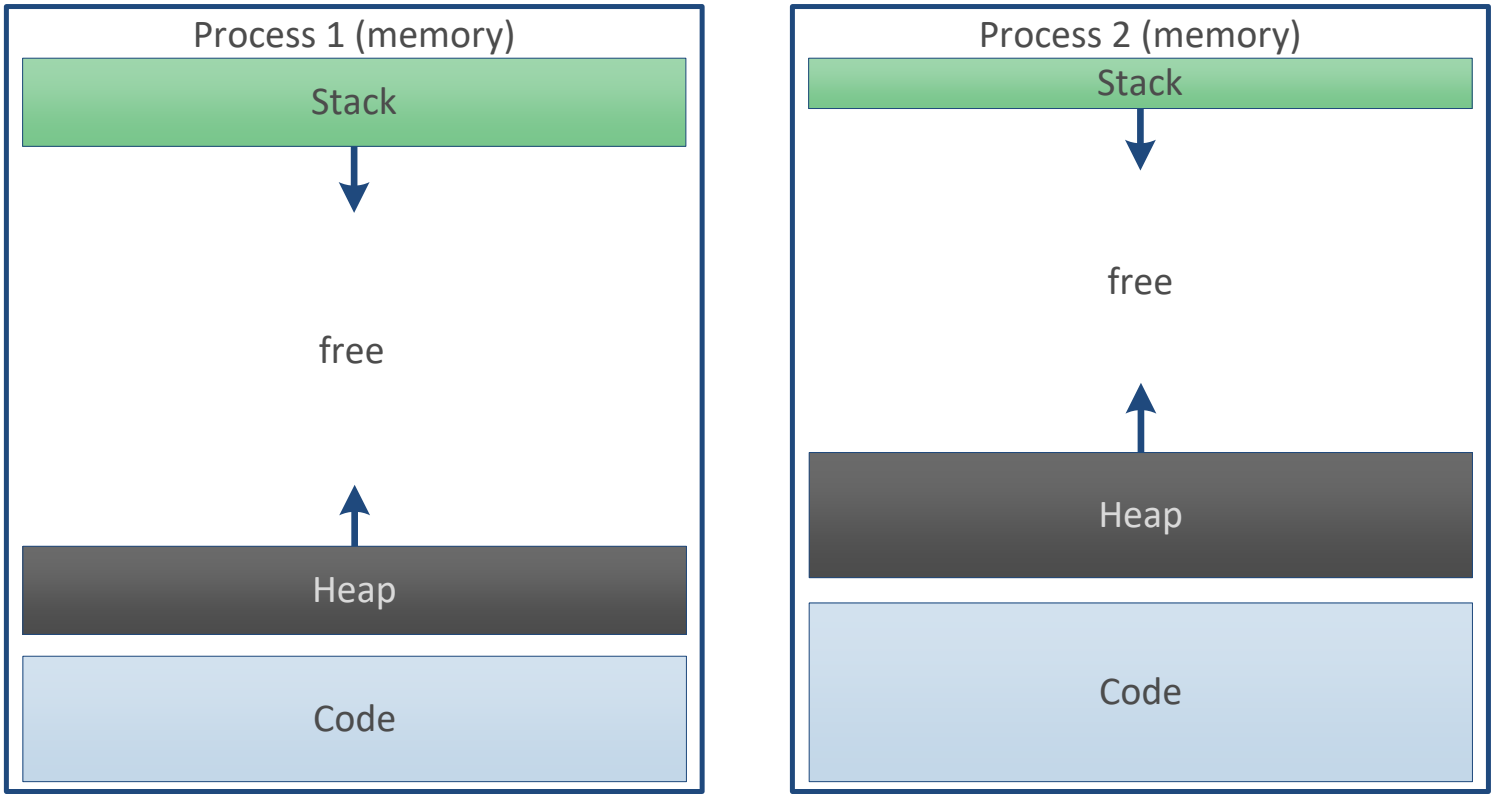
Process isolation



CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

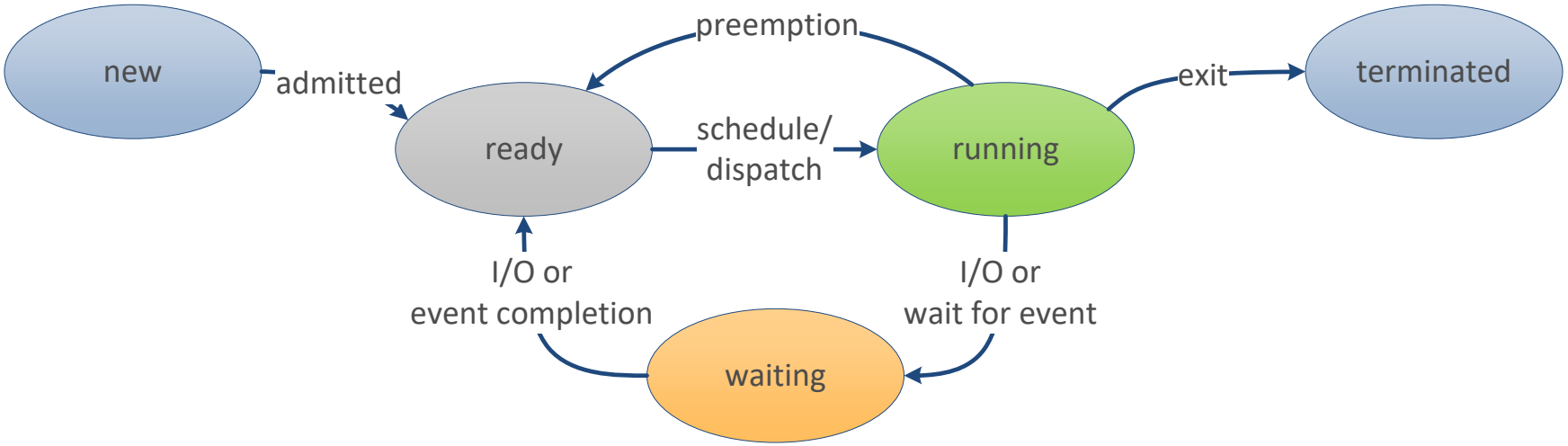
Process memory view



CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Process states

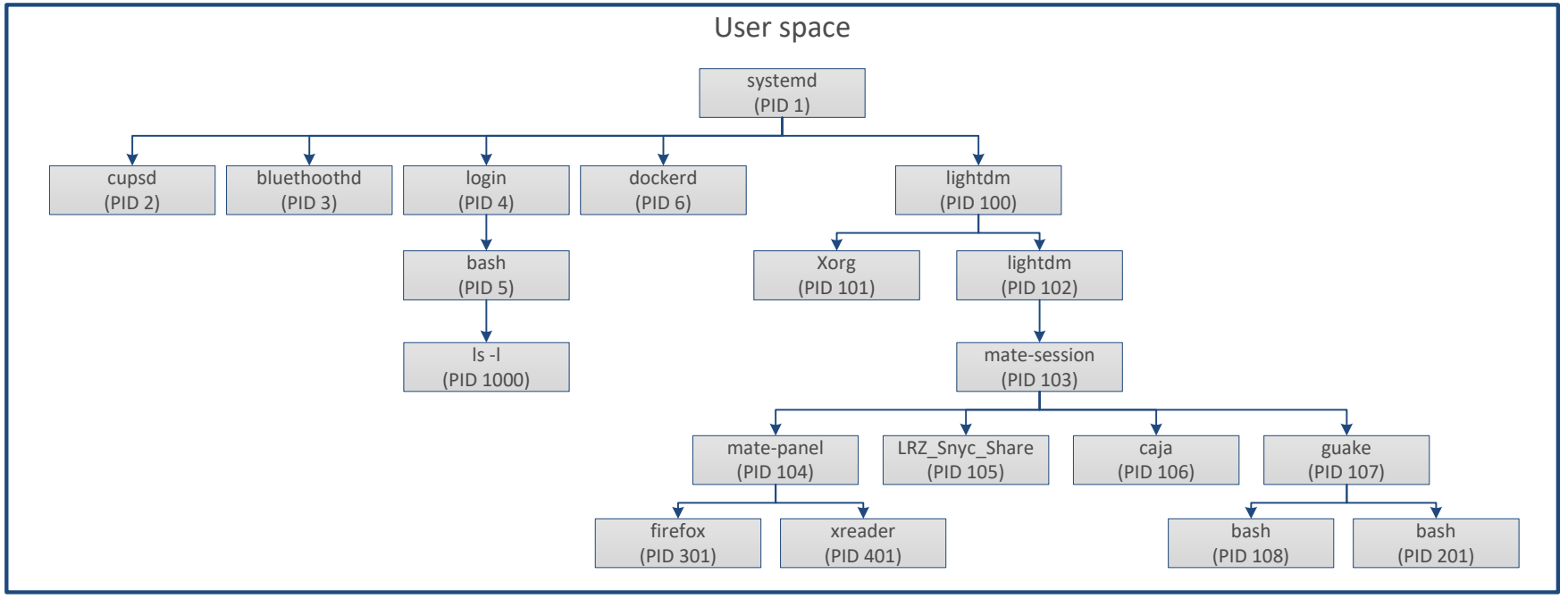


It is still a simplified view!

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Process hierarchy



CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Daemon: a special kind of a process

Daemon - disk and execution monitor (bacronym)

- A daemon is a process
- Operates in the background
- A daemons parent PID is 1 -> `systemd`
- Usually started from `systemd` as part of the boot procedure
- A daemon has no direct interaction (shell, keyboard, mouse) with the user
- Communication with a daemon: network, signals, pipes, shared memory, ...
- Working directory is /
- Usually uses a logfile to log events and errors

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Process specific properties

Process management

- Process id (PID)
- Parent (PID)
- State
- Register entries
- Priority
- Signals
- Start time

Process memory

- Code (pointer)
- Stack (pointer)
- Heap (pointer)

File management

- Root directory (/)
- Working directory
- File descriptors
- User id
- Group id

Advantages of multiple processes

- **Independent start** of different processes
- Can be **executed** in **parallel**
- If a process **crashes** the others can **continue** their work
- **No overwriting** of the memory
- Security: **No read** of another process memory possible
- **Independent development**!
- **Independent dependencies**!
- **Each user** can have its **own processes**.

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

OS process table

Process control block (PCB)

- The **kernel manages** the different **processes**
- Each process has its own **process control block (PCB)**
  - Contains all **process specific properties**
  - The process table contains all PCBs
- In Linux: `struct task_struct {...}`
- `https://github.com/torvalds/linux/blob/master/include/linux/sched.h`

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Programmatic: execute a command

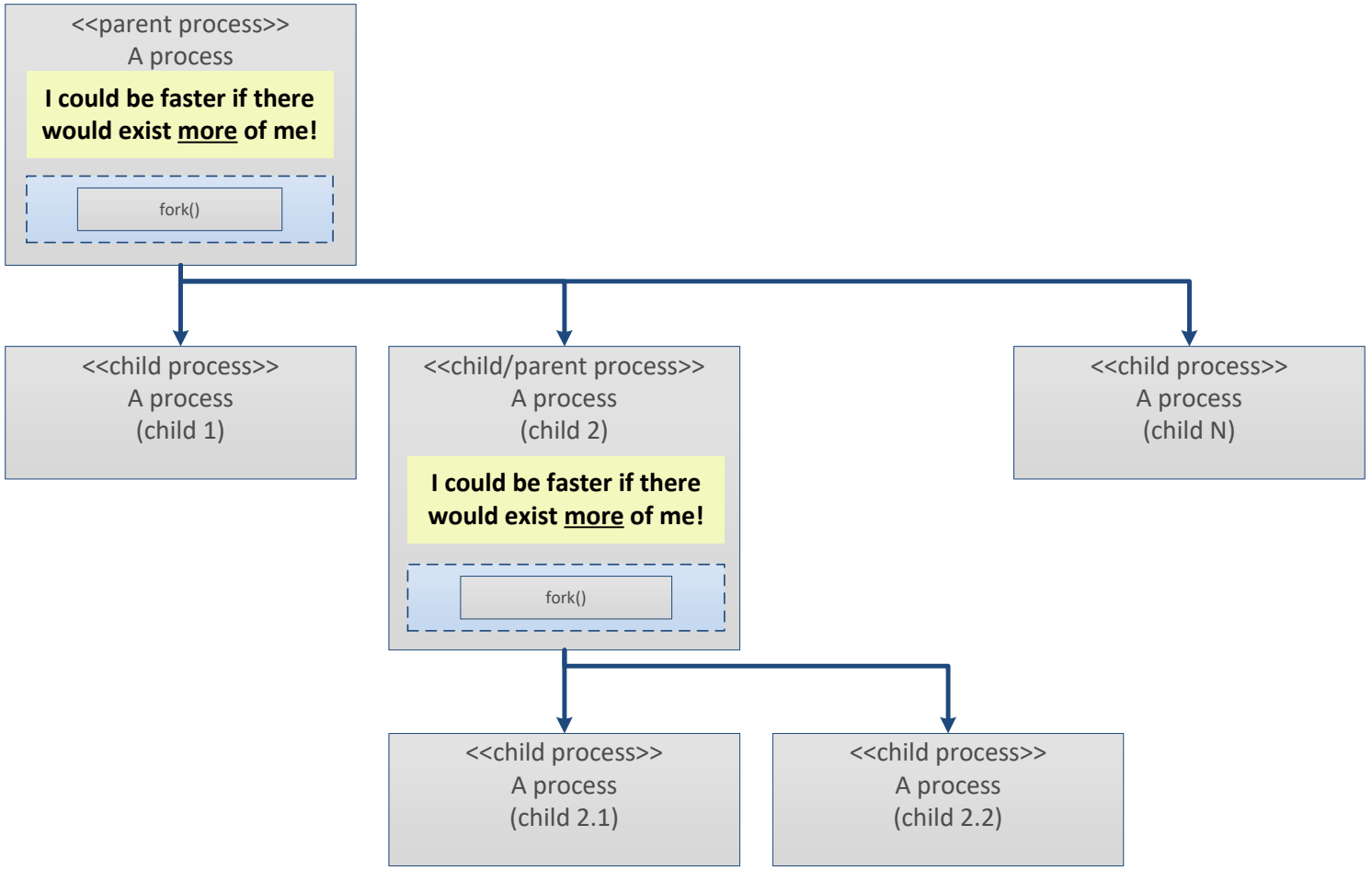
```c
#include <stdio.h>      //printf
#include <stdlib.h>     //EXIT_SUCCESS, system

int main(int argc, char** argv)
{
    //executes a command specified in command by calling /bin/sh -c command
    const char* const command = "ls -l /";
    int exit_status = system(command);

    if(exit_status == -1) {
        printf("%s can't be started.\n", command);
    } else {
        printf("%s exited with status: %d.\n", command, exit_status);
    }

    return EXIT_SUCCESS;
}
```

system (man): http://man7.org/linux/man-pages/man3/system.3.html

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Programmatic: fork idea



CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Programmatic: fork example

```c
#include <stdio.h>     //printf
#include <stdlib.h>    //EXIT_SUCCESS, system
#include <unistd.h>    //fork
#include <sys/wait.h>  //waitpid

int main(int argc, char** argv)
{
    pid_t pid = fork();

    switch(pid){
        case -1: //error
            printf("Error: fork failed.\n");
            exit(EXIT_FAILURE);
            break;
        case 0:  //child
            printf("Hi, I'm the fork with the PID %d!\n", getpid());
            break;
        default: //parent
            printf("Parent waits until child process with PID %d ends.\n", pid);
            waitpid(pid, NULL, 0);
            printf("Child process with PID %d exited.\n", pid);
            break;
    }

    return EXIT_SUCCESS;
}
```

fork (man): http://man7.org/linux/man-pages/man2/fork.2.html

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Processes management on shell

cmd	Description
`./command`	**Start** a process.
`kill`	**Stop** (exit) a process.
`wait`	**Wait** until a child process has stopped.
`ps aux`	**Show** information about started processes.
`top`	**Show live** information about processes.
`pstree`	**Show** the process **hierarchy**.
`renice`	**Change** the priority of the process.

Thread definition

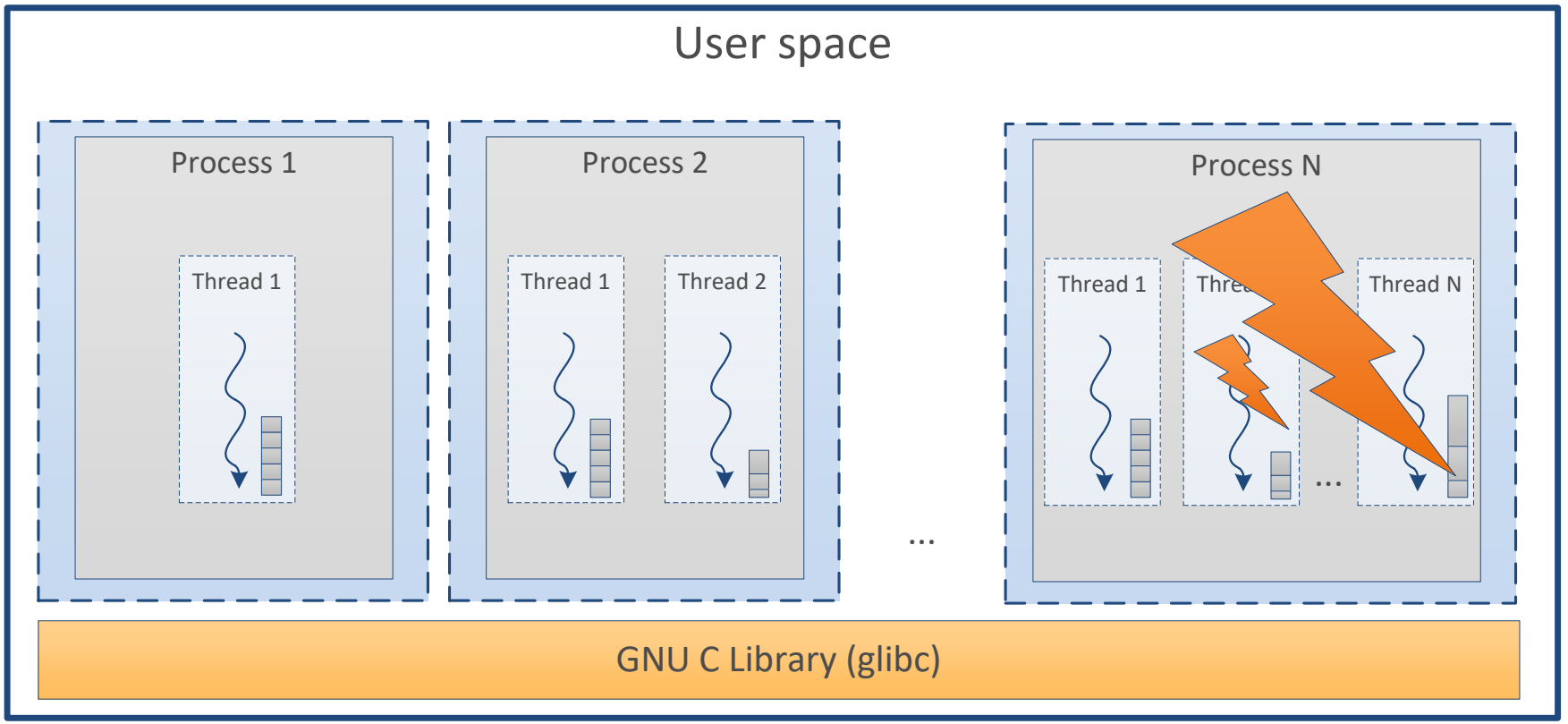
A thread is an **entity within a process** that can be **scheduled independently** of other threads for execution.

In Linux: A thread is a **lightweight process**!

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Thread illustration



CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Thread properties

Properties for each thread

- Thread id (TID)
- State
- Register entries
- Stack (pointer)

Shared process properties

- Address space (code/heap)
- Global variables
- Opened files
- Child processes
- Signals
- Working directory

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

OS thread/process table

Thread control block (TCB)

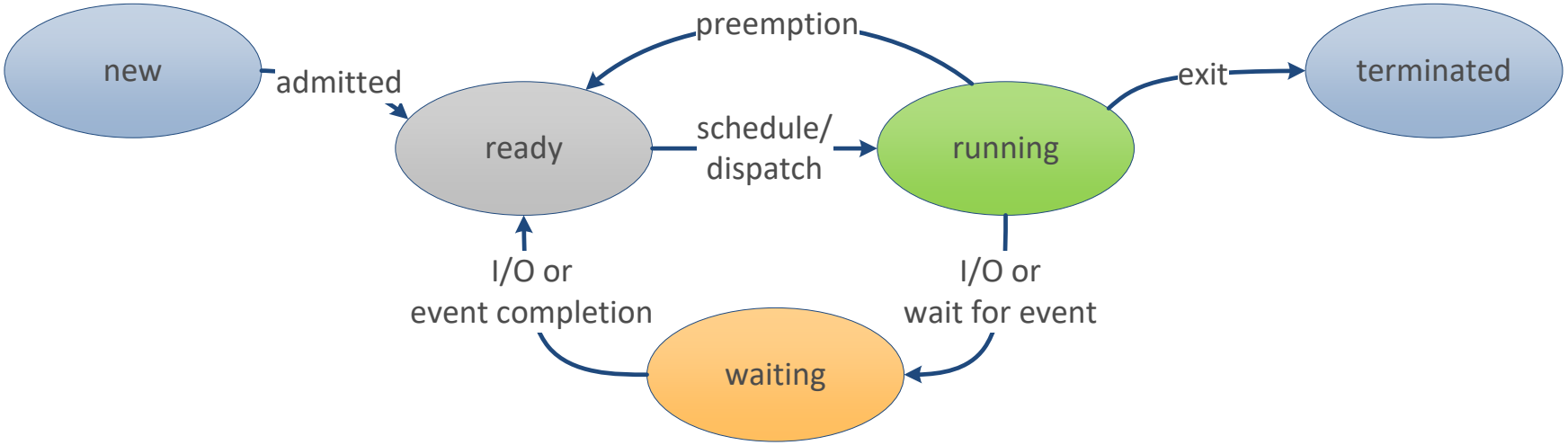
- The **kernel manages** the different **threads** (similar to a process)
- Each thread has its own **thread control block (TCB)**
  - Contains all **thread specific properties**
  - Each process knows the thread TCBs in its PCB
- In Linux:
  - `struct task_struct {...}`
  - It is **used** for **processes and threads**
  - This is the reason for the term: "**lightweight process**"
  - `https://github.com/torvalds/linux/blob/master/include/linux/sched.h`

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Thread states

Similar to the process states.



It is still a simplified view!

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Programmatic: pthread example

```c
#include <stdio.h>     //printf
#include <stdlib.h>    //EXIT_SUCCESS, system
#include <pthread.h>   //pthread_create, pthread_join

void* thread_work() {
    printf("Hi, I'm the thread with the TID %lu!\n", pthread_self());
    return NULL; //the thread ends here
}

int main(int argc, char** argv) {
    //create thread
    pthread_t thread_id;
    int thread_create_state = pthread_create(&thread_id, NULL, &thread_work, NULL);
    if( thread_create_state != 0) {
        printf("Failed creating thread\n");
        exit(EXIT_FAILURE);
    }

    //join thread (wait until the thread has finished)
    printf("Main thread waits until child thread with TID %lu ended.\n", thread_id);
    int thread_exit_state = pthread_join(thread_id, NULL);
    if(thread_exit_state != 0){
        printf("Thread exited with failure %d.\n", thread_exit_state);
    }

    printf("Child thread with TID %lu exited.\n", thread_id);
    return EXIT_SUCCESS;
}
```

pthreads (man): http://man7.org/linux/man-pages/man7/pthreads.7.html

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Process vs. thread

Quiz: When to use a process or a thread?

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Summary and outlook

Summary

- What is a process/thread
- Process hierarchy
- Processes management
- Thread management
- Parallelisation

Outlook

- Process synchronisation
- Semaphore
- Mutual exclusion