Technische
Hochschule
Rosenheim
Technical University of Applied Sciences

Prof. Dr. Florian Künzner

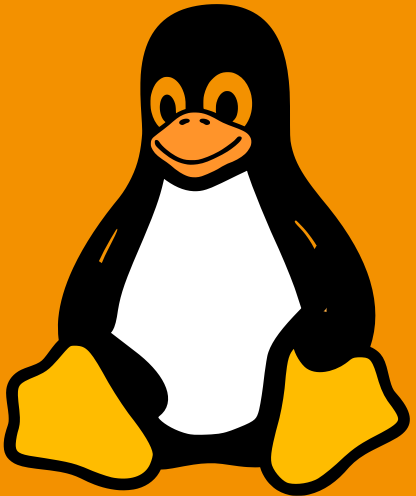
Technical University of Applied Sciences Rosenheim, Computer Science

Start: 8:01

Bitte: Vorname + Nachname

⇒ selbständig umbenennen

OS 8 – Communication 1

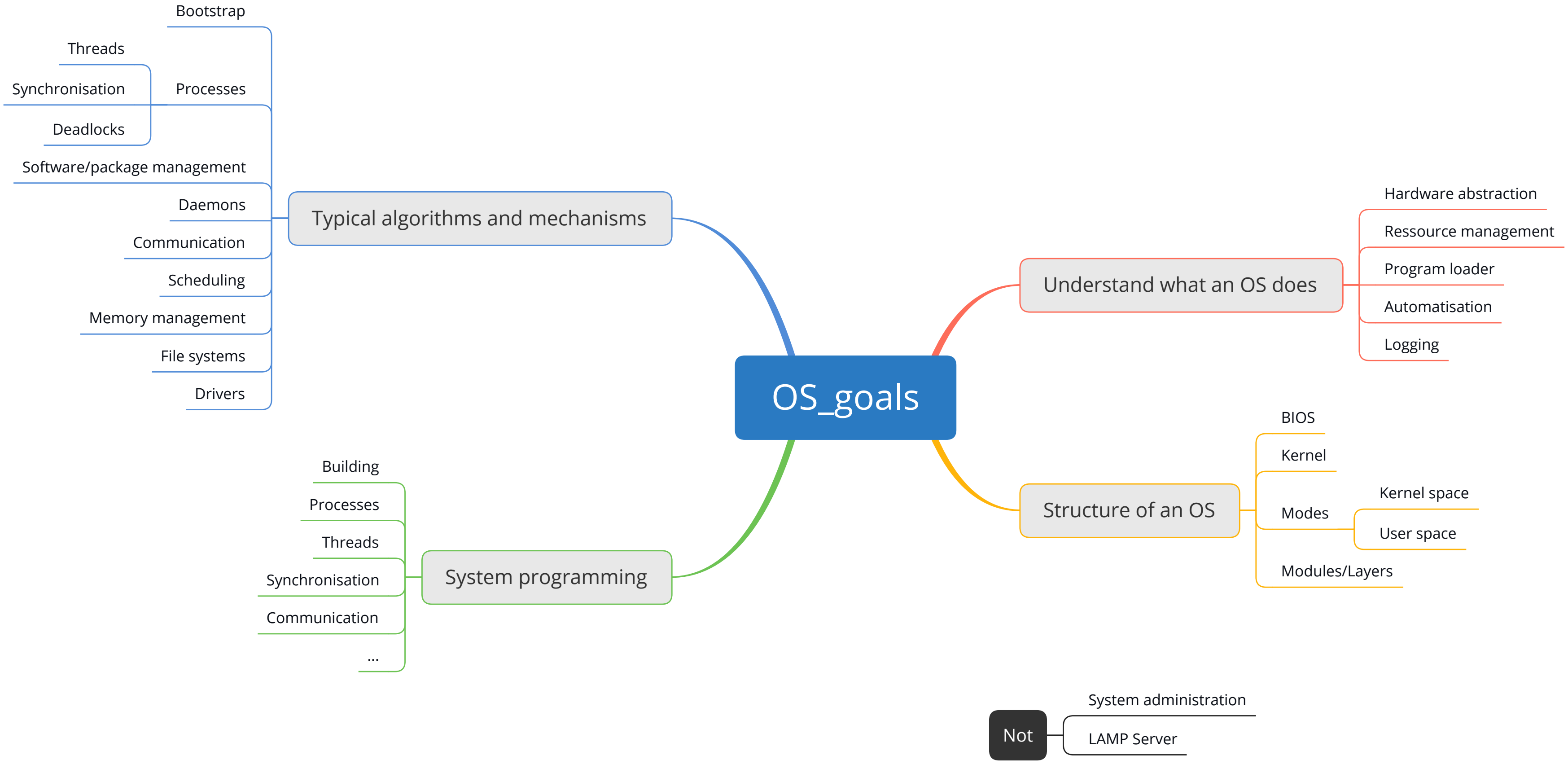


source: iconspng.com

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

CAMPUS Rosenheim
Computer Science

Goal



CAMPUS Rosenheim
Computer Science

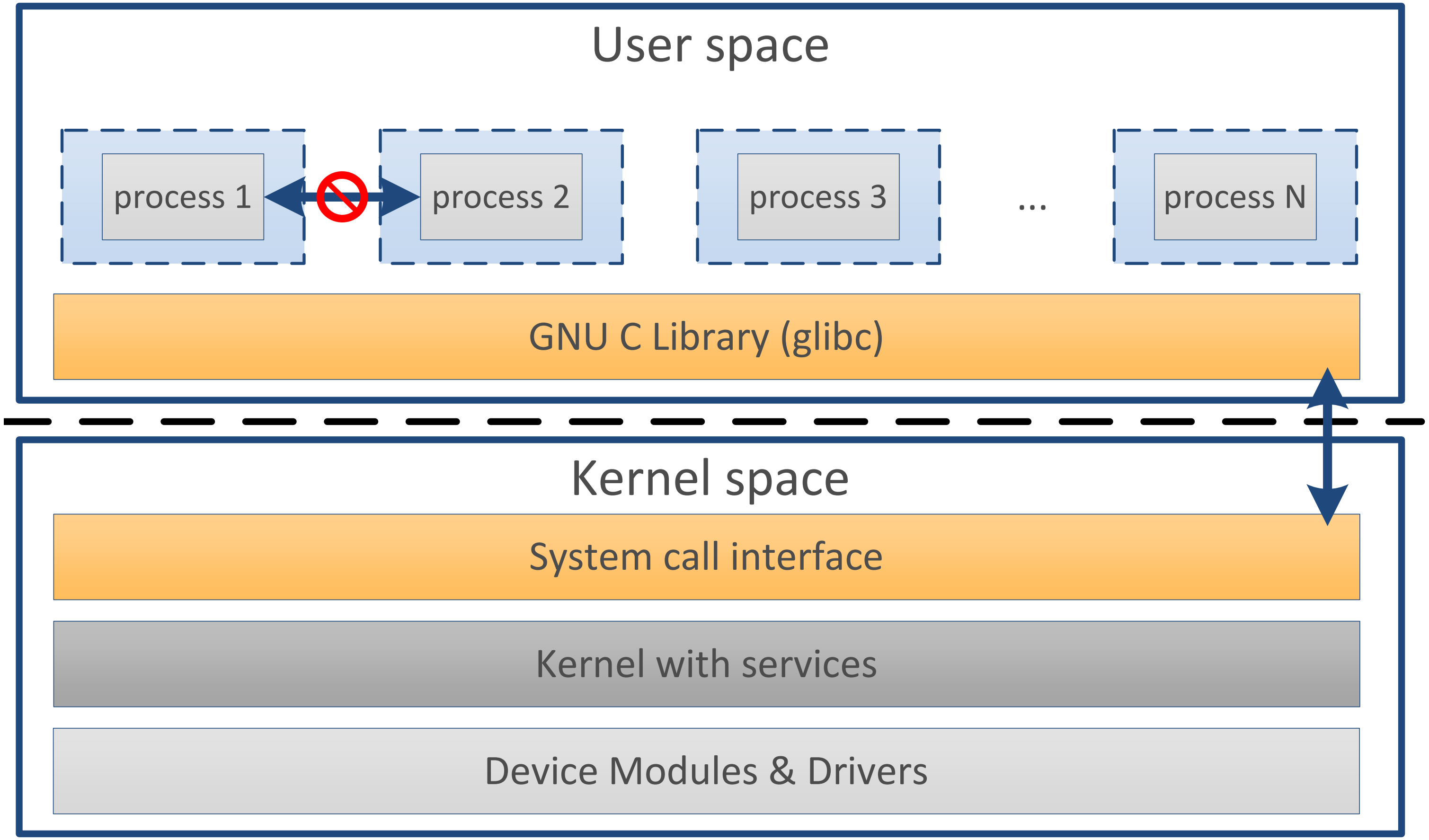
Technische Hochschule Rosenheim
Technical University of Applied Sciences

Goal

OS::Communication

- Process communication concept
- Signals
- Sockets (Unix, network)

Process isolation ⇒ no communication



CAMPUS Rosenheim
Computer Science

Intro

- C-Debugger
- Daemon: PHP → HTML
  ⇒ Browser → Websites
- Zoom
- Youtube
- E-Mail

Why do you want to communicate with a process?

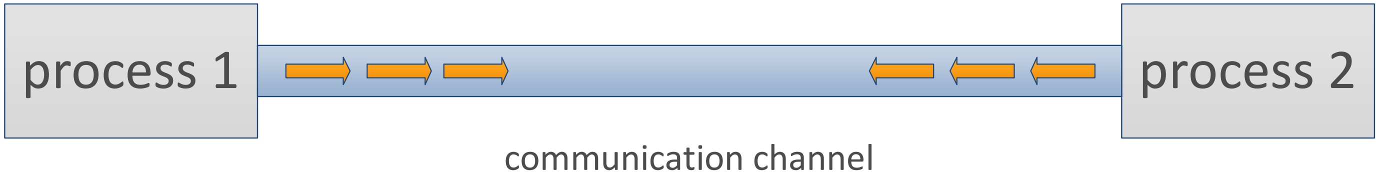
CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Intro

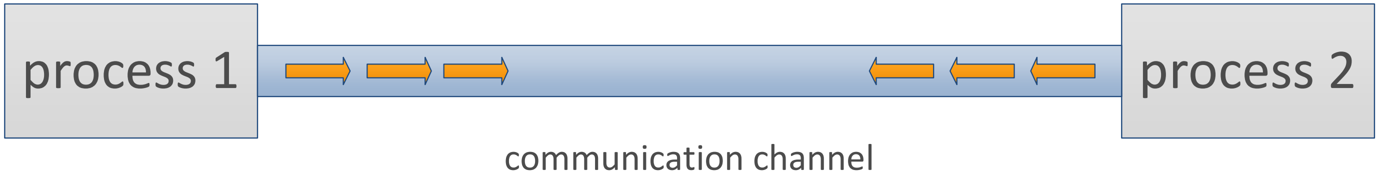
How can we communicate with a process?

Process communication



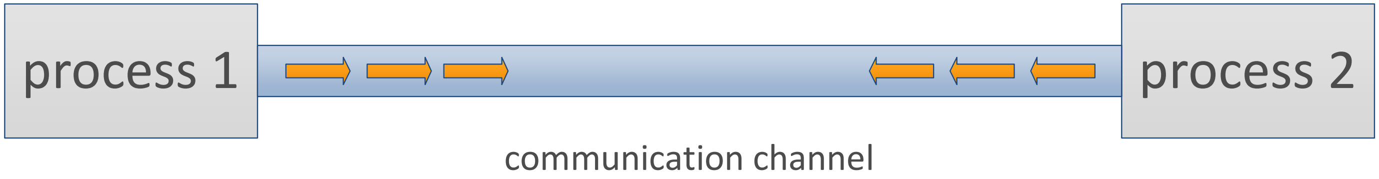
- The communication channel is provided by the OS
- Different types of communication channels exist

Process communication



- The communication channel is provided by the OS
- Different types of communication channels exist

Process communication



- The communication channel is provided by the OS
- Different types of communication channels exist

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Process communication

Important concepts

Function/concept	Description

CAMPUS Rosenheim
Computer Science

Process communication

Important concepts

Function/concept	Description
`send(destination, message)`	**Send** a message **to** the **destination**.

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Process communication

Important concepts

Function/concept	Description
`send(destination, message)`	**Send** a message **to** the **destination**.
`recv(source, &message)`	**Receive** a message **from** the **source**.

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Process communication

Important concepts

Function/concept	Description
`send(destination, message)`	**Send** a message **to** the **destination**.
`recv(source, &message)`	**Receive** a message **from** the **source**.
Blocking/synchron	`send()`/`recv()` **blocks** until the data is fully transferred.

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Process communication

Important concepts

Function/concept	Description
`send(destination, message)`	**Send** a message **to** the **destination**.
`recv(source, &message)`	**Receive** a message **from** the **source**.
Blocking/synchron	`send()/recv()` **blocks** until the data is fully transferred.
Non-blocking/asynchron	`send()/recv()` **immediately returns** and the process can proceed.

CAMPUS Rosenheim
Computer Science

Process communication

Important concepts

Function/concept	Description
`send(destination, message)`	**Send** a message **to** the **destination**.
`recv(source, &message)`	**Receive** a message **from** the **source**.
Blocking/synchron	`send()/recv()` **blocks** until the data is fully transferred.
Non-blocking/asynchron	`send()/recv()` **immediately returns** and the process can proceed.
Protocol required	A **protocol** defines the **order of `send()/recv()`** between processes and the message format.

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Process communication

Important concepts

Function/concept	Description
`send(destination, message)`	**Send** a message **to** the **destination**.
`recv(source, &message)`	**Receive** a message **from** the **source**.
Blocking/synchron	`send()/recv()` **blocks** until the data is fully transferred.
Non-blocking/asynchron	`send()/recv()` **immediately returns** and the process can proceed.
Protocol required	A **protocol** defines the **order of `send()/recv()`** between processes and the message format.
Half-duplex/unidirectional	Communication over a "channel" **only in one** direction.

CAMPUS Rosenheim
Computer Science

Process communication

Important concepts

Function/concept	Description
`send(destination, message)`	**Send** a message **to** the **destination**.
`recv(source, &message)`	**Receive** a message **from** the **source**.
Blocking/synchron	`send()`/`recv()` **blocks** until the data is fully transferred.
Non-blocking/asynchron	`send()`/`recv()` **immediately returns** and the process can proceed.
Protocol required	A **protocol** defines the **order of `send()`/`recv()`** between processes and the message format.
Half-duplex/unidirectional	Communication over a “channel” **only in one** direction.
Full-duplex/bidirectional	Communication over a “channel” **in both** directions.

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Signals

Idea: Signals are **asynchronous events** that interrupt a process.

It is like an **interrupt** request at **process level**.

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Signals

Idea: Signals are **asynchronous events** that interrupt a process.

It is like an **interrupt** request at **process level**.

Signals overview

List of signals: `kill -l`

```
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
```

Signals: some important signals

Nr	Signal	Key	Blockable	Description

More details: `http://man7.org/linux/man-pages/man7/signal.7.html`

Signals: some important signals

Nr	Signal	Key	Blockable	Description
1	SIGHUP		Y	Hangup detected on controlling terminal or death of controlling process

More details: http://man7.org/linux/man-pages/man7/signal.7.html

Signals: some important signals

Nr	Signal	Key	Blockable	Description
1	SIGHUP		Y	Hangup detected on controlling terminal or death of controlling process
2	SIGINT	CTRL+C	Y	Interrupt from keyboard

More details: http://man7.org/linux/man-pages/man7/signal.7.html

Signals: some important signals

Nr	Signal	Key	Blockable	Description
1	SIGHUP		Y	Hangup detected on controlling terminal or death of controlling process
2	SIGINT	CTRL+C	Y	Interrupt from keyboard
3	SIGQUIT	CTRL+\	Y	Quit from keyboard

More details: http://man7.org/linux/man-pages/man7/signal.7.html

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Signals: some important signals

Nr	Signal	Key	Blockable	Description
1	SIGHUP		Y	Hangup detected on controlling terminal or death of controlling process
2	SIGINT	CTRL+C	Y	Interrupt from keyboard
3	SIGQUIT	CTRL+\	Y	Quit from keyboard
4	SIGILL		Y	Illegal Instruction

More details: http://man7.org/linux/man-pages/man7/signal.7.html

Signals: some important signals

Nr	Signal	Key	Blockable	Description
1	SIGHUP		Y	Hangup detected on controlling terminal or death of controlling process
2	SIGINT	CTRL+C	Y	Interrupt from keyboard
3	SIGQUIT	CTRL+\	Y	Quit from keyboard
4	SIGILL		Y	Illegal Instruction
6	SIGABRT		Y	Abort signal from abort()

More details: http://man7.org/linux/man-pages/man7/signal.7.html

Signals: some important signals

Nr	Signal	Key	Blockable	Description
1	SIGHUP		Y	Hangup detected on controlling terminal or death of controlling process
2	SIGINT	CTRL+C	Y	Interrupt from keyboard
3	SIGQUIT	CTRL+\	Y	Quit from keyboard
4	SIGILL		Y	Illegal Instruction
6	SIGABRT		Y	Abort signal from abort()
8	SIGFPE		Y	Floating-point exception

More details: http://man7.org/linux/man-pages/man7/signal.7.html

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Signals: some important signals

Nr	Signal	Key	Blockable	Description
1	`SIGHUP`		Y	Hangup detected on controlling terminal or death of controlling process
2	`SIGINT`	`CTRL+C`	Y	Interrupt from keyboard
3	`SIGQUIT`	`CTRL+\`	Y	Quit from keyboard
4	`SIGILL`		Y	Illegal Instruction
6	`SIGABRT`		Y	Abort signal from `abort()`
8	`SIGFPE`		Y	Floating-point exception
9	`SIGKILL`		N	Kill signal

More details: `http://man7.org/linux/man-pages/man7/signal.7.html`

Signals: some important signals

Nr	Signal	Key	Blockable	Description
1	SIGHUP		Y	Hangup detected on controlling terminal or death of controlling process
2	SIGINT	CTRL+C	Y	Interrupt from keyboard
3	SIGQUIT	CTRL+\	Y	Quit from keyboard
4	SIGILL		Y	Illegal Instruction
6	SIGABRT		Y	Abort signal from abort()
8	SIGFPE		Y	Floating-point exception
9	SIGKILL		N	Kill signal
14	SIGALRM		Y	Timer signal from alarm()

More details: http://man7.org/linux/man-pages/man7/signal.7.html

Signals: some important signals

Nr	Signal	Key	Blockable	Description
1	SIGHUP		Y	Hangup detected on controlling terminal or death of controlling process
2	SIGINT	CTRL+C	Y	Interrupt from keyboard
3	SIGQUIT	CTRL+\	Y	Quit from keyboard
4	SIGILL		Y	Illegal Instruction
6	SIGABRT		Y	Abort signal from abort()
8	SIGFPE		Y	Floating-point exception
9	SIGKILL		N	Kill signal
14	SIGALRM		Y	Timer signal from alarm()
15	SIGTERM		Y	Termination signal

More details: http://man7.org/linux/man-pages/man7/signal.7.html

Signals: some important signals

Nr	Signal	Key	Blockable	Description
1	SIGHUP		Y	Hangup detected on controlling terminal or death of controlling process
2	SIGINT	CTRL+C	Y	Interrupt from keyboard
3	SIGQUIT	CTRL+\	Y	Quit from keyboard
4	SIGILL		Y	Illegal Instruction
6	SIGABRT		Y	Abort signal from abort()
8	SIGFPE		Y	Floating-point exception
9	SIGKILL		N	Kill signal
14	SIGALRM		Y	Timer signal from alarm()
15	SIGTERM		Y	Termination signal
10	SIGUSR1		Y	User-defined signal 1

More details: http://man7.org/linux/man-pages/man7/signal.7.html

Signals: some important signals

Nr	Signal	Key	Blockable	Description
1	`SIGHUP`		Y	Hangup detected on controlling terminal or death of controlling process
2	`SIGINT`	`CTRL+C`	Y	Interrupt from keyboard
3	`SIGQUIT`	`CTRL+\`	Y	Quit from keyboard
4	`SIGILL`		Y	Illegal Instruction
6	`SIGABRT`		Y	Abort signal from `abort()`
8	`SIGFPE`		Y	Floating-point exception
9	`SIGKILL`		N	Kill signal
14	`SIGALRM`		Y	Timer signal from `alarm()`
15	`SIGTERM`		Y	Termination signal
10	`SIGUSR1`		Y	User-defined signal 1
12	`SIGUSR2`		Y	User-defined signal 2

More details: `http://man7.org/linux/man-pages/man7/signal.7.html`

Signals: some important signals

Nr	Signal	Key	Blockable	Description
1	SIGHUP		Y	Hangup detected on controlling terminal or death of controlling process
2	SIGINT	CTRL+C	Y	Interrupt from keyboard
3	SIGQUIT	CTRL+\	Y	Quit from keyboard
4	SIGILL		Y	Illegal Instruction
6	SIGABRT		Y	Abort signal from abort()
8	SIGFPE		Y	Floating-point exception
9	SIGKILL		N	Kill signal
14	SIGALRM		Y	Timer signal from alarm()
15	SIGTERM		Y	Termination signal
10	SIGUSR1		Y	User-defined signal 1
12	SIGUSR2		Y	User-defined signal 2
18	SIGCONT		Y	Continue if stopped

More details: http://man7.org/linux/man-pages/man7/signal.7.html

Signals: some important signals

Nr	Signal	Key	Blockable	Description
1	SIGHUP		Y	Hangup detected on controlling terminal or death of controlling process
2	SIGINT	CTRL+C	Y	Interrupt from keyboard
3	SIGQUIT	CTRL+\	Y	Quit from keyboard
4	SIGILL		Y	Illegal Instruction
6	SIGABRT		Y	Abort signal from abort()
8	SIGFPE		Y	Floating-point exception
9	SIGKILL		N	Kill signal
14	SIGALRM		Y	Timer signal from alarm()
15	SIGTERM		Y	Termination signal
10	SIGUSR1		Y	User-defined signal 1
12	SIGUSR2		Y	User-defined signal 2
18	SIGCONT		Y	Continue if stopped
19	SIGSTOP		N	Stop process

More details: http://man7.org/linux/man-pages/man7/signal.7.html

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Signals: some important signals

Nr	Signal	Key	Blockable	Description
1	SIGHUP		Y	Hangup detected on controlling terminal or death of controlling process
2	SIGINT	CTRL+C	Y	Interrupt from keyboard
3	SIGQUIT	CTRL+\	Y	Quit from keyboard
4	SIGILL		Y	Illegal Instruction
6	SIGABRT		Y	Abort signal from `abort()`
8	SIGFPE		Y	Floating-point exception
9	SIGKILL		N	Kill signal
14	SIGALRM		Y	Timer signal from `alarm()`
15	SIGTERM		Y	Termination signal
10	SIGUSR1		Y	User-defined signal 1
12	SIGUSR2		Y	User-defined signal 2
18	SIGCONT		Y	Continue if stopped
19	SIGSTOP		N	Stop process
20	SIGTSTP	CTRL+Z	Y	Stop typed at terminal

More details: `http://man7.org/linux/man-pages/man7/signal.7.html`

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Signals: some important signals

Nr	Signal	Key	Blockable	Description
1	`SIGHUP`		Y	Hangup detected on controlling terminal or death of controlling process
2	`SIGINT`	`CTRL+C`	Y	Interrupt from keyboard
3	`SIGQUIT`	`CTRL+\`	Y	Quit from keyboard
4	`SIGILL`		Y	Illegal Instruction
6	`SIGABRT`		Y	Abort signal from `abort()`
8	`SIGFPE`		Y	Floating-point exception
9	`SIGKILL`		N	Kill signal
14	`SIGALRM`		Y	Timer signal from `alarm()`
15	`SIGTERM`		Y	Termination signal
10	`SIGUSR1`		Y	User-defined signal 1
12	`SIGUSR2`		Y	User-defined signal 2
18	`SIGCONT`		Y	Continue if stopped
19	`SIGSTOP`		N	Stop process
20	`SIGTSTP`	`CTRL+Z`	Y	Stop typed at terminal

More details: `http://man7.org/linux/man-pages/man7/signal.7.html`

Signals: handling

- If a process receives a signal: the **signal** is **saved** in the **PCB**.
- If the process state changes to “**running**” the **process** will be **interrupted**.
- The operating system looks if there is a registered handler for the received signal
  - If there is a **registered handler**, then **this function** will be called.
  - If there **no handler** registered, the **default handler** will be called.
- If the handler hasn’t exited the process, the **process proceeds** exactly at the **position before** it was **interrupted**.

Signals: handling

- If a process receives a signal: the **signal** is **saved** in the **PCB**.
- If the process state changes to “**running**” the **process** will be **interrupted**.
- The operating system looks if there is a registered handler for the received signal
  - If there is a **registered handler**, then **this function** will be called.
  - If there **no handler** registered, the **default handler** will be called.
- If the handler hasn’t exited the process, the **process proceeds** exactly at the **position before** it was **interrupted**.

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Signals: handling

- If a process receives a signal: the **signal** is **saved** in the **PCB**.
- If the process state changes to "**running**" the **process** will be **interrupted**.
- The operating system looks if there is a registered handler for the received signal
  - If there is a **registered handler**, then **this function** will be called.
  - If there **no handler** registered, the **default handler** will be called.
- If the handler hasn't exited the process, the **process proceeds** exactly at the **position before** it was **interrupted**.

Signals: handling

- If a process receives a signal: the **signal** is **saved** in the **PCB**.
- If the process state changes to "**running**" the **process** will be **interrupted**.
- The operating system looks if there is a registered handler for the received signal
  - If there is a **registered handler**, then **this function** will be called.
  - If there **no handler** registered, the **default handler** will be called.
- If the handler hasn't exited the process, the **process proceeds** exactly at the **position before** it was **interrupted**.

Signals: handling

- If a process receives a signal: the **signal** is **saved** in the **PCB**.
- If the process state changes to "**running**" the **process** will be **interrupted**.
- The operating system looks if there is a registered handler for the received signal
  - If there is a **registered handler**, then **this function** will be called.
  - If there **no handler** registered, the **default handler** will be called.
- If the handler hasn't exited the process, the **process proceeds** exactly at the **position before** it was **interrupted**.

Signals: handling

- If a process receives a signal: the **signal** is **saved** in the **PCB**.
- If the process state changes to "**running**" the **process** will be **interrupted**.
- The operating system looks if there is a registered handler for the received signal
  - If there is a **registered handler**, then **this function** will be called.
  - If there **no handler** registered, the **default handler** will be called.
- If the handler hasn't exited the process, the **process proceeds** exactly at the **position before** it was **interrupted**.

Signals: handling

- If a process receives a signal: the **signal** is **saved** in the **PCB**.
- If the process state changes to “**running**” the **process** will be **interrupted**.
- The operating system looks if there is a registered handler for the received signal
  - If there is a **registered handler**, then **this function** will be called.
  - If there **no handler** registered, the **default handler** will be called.
- If the handler hasn’t exited the process, the **process proceeds** exactly at the **position before** it was **interrupted**.

CAMPUS Rosenheim
Computer Science

Signals: shell

Commands

Command	Description

CAMPUS Rosenheim
Computer Science

Signals: shell

Commands

Command	Description
`kill PID`	Sends the signal **15 (SIGTERM)** to the process.

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Signals: shell

Commands

Command	Description
`kill PID`	Sends the signal **15 (SIGTERM)** to the process.
`kill -1 PID`	Sends the signal **1 (SIGHUP)** to the process.

CAMPUS Rosenheim
Computer Science

Signals: shell

Commands

Command	Description
`kill PID`	Sends the signal **15 (SIGTERM)** to the process.
`kill -1 PID`	Sends the signal **1 (SIGHUP)** to the process.

CAMPUS Rosenheim
Computer Science

Signals: shell

Commands

Command	Description
`kill PID`	Sends the signal **15 (SIGTERM)** to the process.
`kill -1 PID`	Sends the signal **1 (SIGHUP)** to the process.
`kill -SIGHUP PID`	Sends the signal **1 (SIGHUP)** to the process.

CAMPUS Rosenheim
Computer Science

Signals: shell

Commands

Command	Description
`kill PID`	Sends the signal **15 (SIGTERM)** to the process.
`kill -1 PID`	Sends the signal **1 (SIGHUP)** to the process.
`kill -SIGHUP PID`	Sends the signal **1 (SIGHUP)** to the process.
`killall process_name`	Sends the signal **15 (SIGTERM)** to the process.

Signals: shell

Commands

Command	Description
`kill PID`	Sends the signal **15 (SIGTERM)** to the process.
`kill -1 PID`	Sends the signal **1 (SIGHUP)** to the process.
`kill -SIGHUP PID`	Sends the signal **1 (SIGHUP)** to the process.
`killall process_name`	Sends the signal **15 (SIGTERM)** to the process.
`killall -s HUP process_name`	Sends the signal **1~~5~~ (~~SIGTERM~~)** to the process. 1 SIGHUP

Signals: signal handling C example

```c
#include <stdio.h>  //printf
#include <stdlib.h> //EXIT_SUCCESS
#include <signal.h> //signal
#include <unistd.h> //sleep

void signal_handler(int signal) {
    printf("No, I don't want to terminate right now!\n");
}

int main(int argc, char** argv) {
    //register the signal handler
    signal(SIGTERM, signal_handler);

    for(long long int i = 0; i < __LONG_LONG_MAX__; ++i) { //do something usefull...
        printf("sleeping!!\n");
        sleep(5);
    }

    printf("%s exits main() now!\n", argv[0]);
    return EXIT_SUCCESS;
}
```

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Signals: C function overview

Function*	Description

*return types not shown here!

CAMPUS Rosenheim
Computer Science

Signals: C function overview

Function*	Description
`raise(int sig);`	**Sends** a **signal** to the calling process or thread.

*return types not shown here!

CAMPUS Rosenheim
Computer Science

Signals: C function overview

Function*	Description
`raise(int sig);`	**Sends** a **signal** to the calling process or thread.
`kill(pid_t pid, int sig);`	**Sends** a **signal** to the process with the specified pid.

*return types not shown here!

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Signals: C function overview

Function*	Description
`raise(int sig);`	**Sends** a **signal** to the calling process or thread.
`kill(pid_t pid, int sig);`	**Sends** a **signal** to the process with the specified pid.
`pause(void);`	Causes the calling process or thread to **sleep until a signal is delivered**.

*return types not shown here!

CAMPUS Rosenheim
Computer Science

Signals: C function overview

Function*	Description
`raise(int sig);`	**Sends** a **signal** to the calling process or thread.
`kill(pid_t pid, int sig);`	**Sends** a **signal** to the process with the specified pid.
`pause(void);`	Causes the calling process or thread to **sleep until a signal is delivered**.
`sleep(unsigned int seconds);`	**Sleeps** for the **specified seconds** or **until a signal delivered**.

*return types not shown here!

Signals: C function overview

Function*	Description
`raise(int sig);`	**Sends** a **signal** to the calling process or thread.
`kill(pid_t pid, int sig);`	**Sends** a **signal** to the process with the specified pid.
`pause(void);`	Causes the calling process or thread to **sleep until a signal is delivered**.
`sleep(unsigned int seconds);`	**Sleeps** for the **specified seconds** or **until a signal delivered**.
`alarm(unsigned int seconds);`	**Sends an alarm** to the calling process or thread in the specified seconds.

*return types not shown here!

CAMPUS Rosenheim
Computer Science

Signals: C function overview

Function*	Description
`raise(int sig);`	**Sends** a **signal** to the calling process or thread.
`kill(pid_t pid, int sig);`	**Sends** a **signal** to the process with the specified pid.
`pause(void);`	Causes the calling process or thread to **sleep until a signal is delivered**.
`sleep(unsigned int seconds);`	**Sleeps** for the **specified seconds** or **until a signal delivered**.
`alarm(unsigned int seconds);`	**Sends an alarm** to the calling process or thread in the specified seconds.
`signal(int signum, sighandler_t handler);`	**Registers** a **signal handler** for signum.

*return types not shown here!

CAMPUS Rosenheim
Computer Science

Signals: C function overview

Function*	Description
`raise(int sig);`	**Sends** a **signal** to the calling process or thread.
`kill(pid_t pid, int sig);`	**Sends** a **signal** to the process with the specified pid.
`pause(void);`	Causes the calling process or thread to **sleep until a signal is delivered**.
`sleep(unsigned int seconds);`	**Sleeps** for the **specified seconds** or **until a signal delivered**.
`alarm(unsigned int seconds);`	**Sends an alarm** to the calling process or thread in the specified seconds.
`signal(int signum, sighandler_t handler);`	**Registers** a **signal handler** for signum.
`signal(int signum, SIG_IGN);`	**Ignores** signals for signum, by setting a `SIG_IGN` handler, which doesn't exits the process.

*return types not shown here!

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Signals: C function overview

Function*	Description
`raise(int sig);`	**Sends** a **signal** to the calling process or thread.
`kill(pid_t pid, int sig);`	**Sends** a **signal** to the process with the specified pid.
`pause(void);`	Causes the calling process or thread to **sleep until a signal is delivered**.
`sleep(unsigned int seconds);`	**Sleeps** for the **specified seconds** or **until a signal delivered**.
`alarm(unsigned int seconds);`	**Sends an alarm** to the calling process or thread in the specified seconds.
`signal(int signum, sighandler_t handler);`	**Registers** a **signal handler** for signum.
`signal(int signum, SIG_IGN);`	**Ignores** signals for signum, by setting a SIG_IGN handler, which doesn't exits the process.
`signal(int signum, SIG_DFL);`	**Sets** the **default handler** for signum.

*return types not shown here!

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Questions?

All right? ⇒ ✓

Question? ⇒ ✋ and use **chat**

or

speak *after* I ask you to

CAMPUS Rosenheim
Computer Science

Sockets

Socket concept

- Endpoint for sending or receiving data
- Inter-process communication (IPC)
- Byte oriented data transfer
- Full-duplex -> `send()`/`recv()` over the same socket

CAMPUS Rosenheim
Computer Science

Sockets

Socket concept

- Endpoint for sending or receiving data
- Inter-process communication (IPC)
- Byte oriented data transfer
- Full-duplex -> `send()`/`recv()` over the same socket

CAMPUS Rosenheim
Computer Science

Sockets

Socket concept

- Endpoint for sending or receiving data
- Inter-process communication (IPC)
- Byte oriented data transfer
- Full-duplex -> `send()`/`recv()` over the same socket

CAMPUS Rosenheim
Computer Science

Sockets

Socket concept

- Endpoint for sending or receiving data
- Inter-process communication (IPC)
- Byte oriented data transfer
- Full-duplex -> `send()/recv()` over the same socket

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Sockets

Socket concept

- Endpoint for sending or receiving data
- Inter-process communication (IPC)
- Byte oriented data transfer
- Full-duplex -> `send()`/`recv()` over the same socket

CAMPUS Rosenheim
Computer Science

Technische
Hochschule
Rosenheim
Technical University of Applied Sciences

Sockets

Connection oriented vs connectionless.

↑ TCP

↑ UDP

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Socket: connection oriented

Pseudo C code

```
void server() {
  socket(...); //create comm. interface
  bind(...);   //connect address with socket
  listen(...); //create a queue
  accept(...); //wait until client connects


  //unblock the server


  //receive data: wait for data
  recv(...)
  //...



  //close socket and connection
  close(...);
}
```

```
void client() {




  socket(...);  //create comm. interface
  connect(...); //connect to server

  //send data: wait until data are sent
  send(...)


  //...

  //close socket and connection
  close(...);


}
```

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Sockets

Unix vs network sockets.

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Questions?

All right? ⇒ ✓

Question? ⇒ ✋ and use **chat**

or

speak *after* I ask you to

CAMPUS Rosenheim
Computer Science

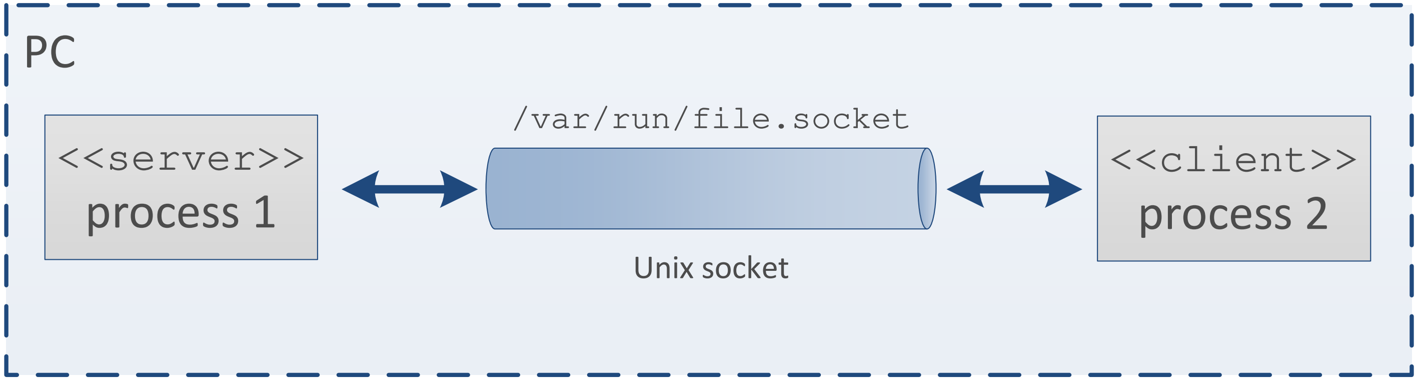
Technische Hochschule Rosenheim
Technical University of Applied Sciences

Unix sockets

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Unix sockets



CAMPUS Rosenheim
Computer Science

Unix sockets

Unix socket concept

- Unix domain
- Communication only on same PC
- Is faster than network (TCP/IP or UDP/IP) socket
- Use file system as address name space
- User ID can be determined
- Access control via file system

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Unix sockets

Unix socket concept

- Unix domain
- Communication only on same PC
- Is faster than network (TCP/IP or UDP/IP) socket
- Use file system as address name space
- User ID can be determined
- Access control via file system

CAMPUS Rosenheim
Computer Science

Unix sockets

Unix socket concept

- Unix domain
- Communication only on same PC
- Is faster than network (TCP/IP or UDP/IP) socket
- Use file system as address name space
- User ID can be determined
- Access control via file system

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Unix sockets

Unix socket concept

- Unix domain
- Communication only on same PC
- Is faster than network (TCP/IP or UDP/IP) socket
- Use file system as address name space
- User ID can be determined
- Access control via file system

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Unix sockets

Unix socket concept

- Unix domain
- Communication only on same PC
- Is faster than network (TCP/IP or UDP/IP) socket
- Use file system as address name space
- User ID can be determined
- Access control via file system

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Unix sockets

Unix socket concept

- Unix domain
- Communication only on same PC
- Is faster than network (TCP/IP or UDP/IP) socket
- Use file system as address name space
- User ID can be determined
- Access control via file system

CAMPUS Rosenheim

Computer Science

Technische Hochschule Rosenheim

Technical University of Applied Sciences

Unix sockets

Unix socket concept

- Unix domain
- Communication only on same PC
- Is faster than network (TCP/IP or UDP/IP) socket
- Use file system as address name space
- User ID can be determined
- Access control via file system

CAMPUS Rosenheim
Computer Science

Unix sockets

Example

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Questions?

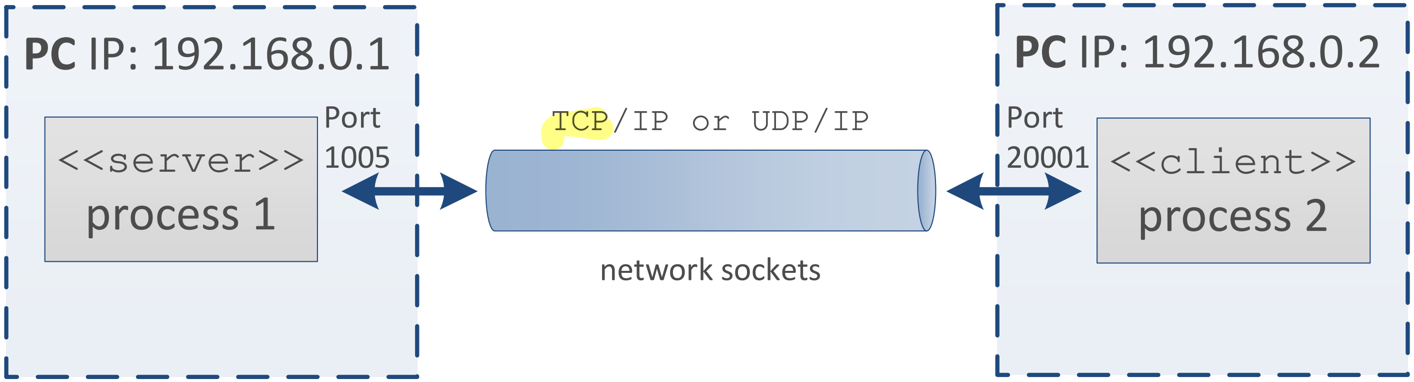
All right? ⇒ ✓

Question? ⇒ ✋ and use **chat**

or

speak *after* I ask you to

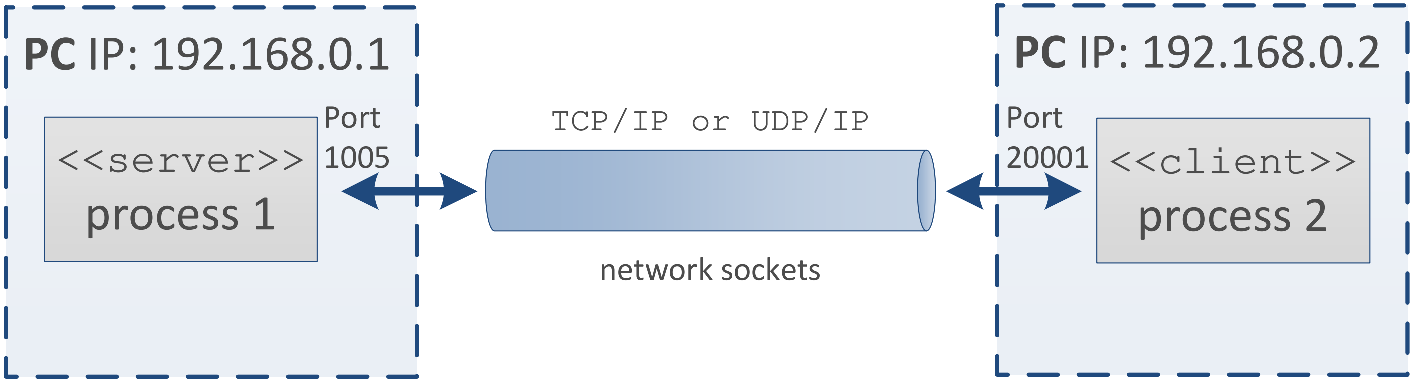
Network sockets: remote



CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

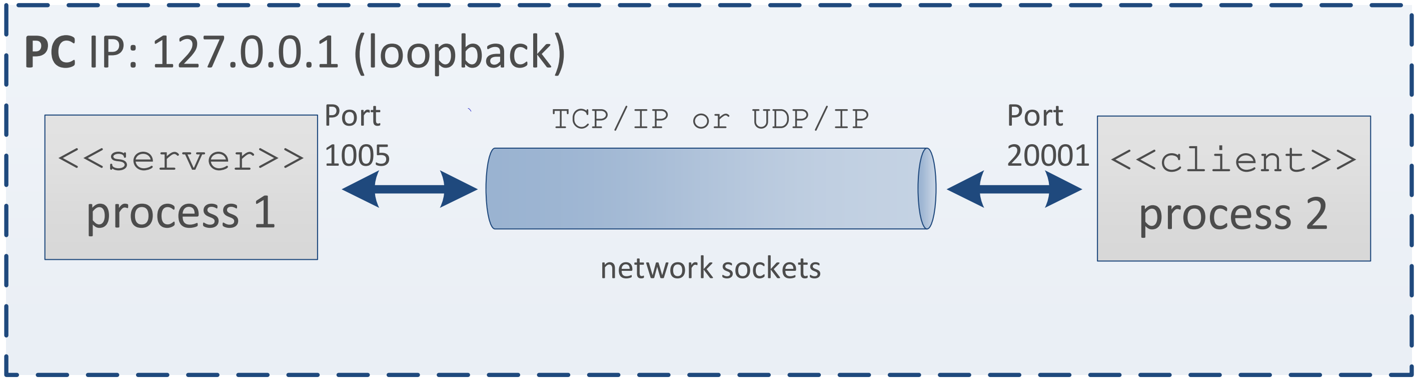
Network sockets: remote



CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Network sockets: local



CAMPUS Rosenheim
Computer Science

Network sockets

Network socket concept

- Internet/network domain
- Communication over the network
- Communication on same PC over loopback
- TCP/IP: connection oriented
- UDP/IP: simple connectionless communication
- Access control on package filter level

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Network sockets

Network socket concept

- Internet/network domain
- Communication over the network
- Communication on same PC over loopback
- TCP/IP: connection oriented
- UDP/IP: simple connectionless communication
- Access control on package filter level

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Network sockets

Network socket concept

- Internet/network domain
- Communication over the network
- Communication on same PC over loopback
- TCP/IP: connection oriented
- UDP/IP: simple connectionless communication
- Access control on package filter level

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Network sockets

Network socket concept

- Internet/network domain
- Communication over the network
- Communication on same PC over loopback
- TCP/IP: connection oriented
- UDP/IP: simple connectionless communication
- Access control on package filter level

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Network sockets

Network socket concept

- Internet/network domain
- Communication over the network
- Communication on same PC over loopback
- TCP/IP: connection oriented
- UDP/IP: simple connectionless communication
- Access control on package filter level

Network sockets

Network socket concept

- Internet/network domain
- Communication over the network
- Communication on same PC over loopback
- TCP/IP: connection oriented
- UDP/IP: simple connectionless communication
- Access control on package filter level

CAMPUS Rosenheim
Computer Science

Network sockets

Network socket concept

- Internet/network domain
- Communication over the network
- Communication on same PC over loopback
- TCP/IP: connection oriented
- UDP/IP: simple connectionless communication
- Access control on package filter level

CAMPUS Rosenheim
Computer Science

Network sockets

C example

Questions?

All right? ⇒ ✓

Question? ⇒ ✋ and use **chat**

or

speak *after* I ask you to

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Summary and outlook

Summary

- Process communication concept
- Signals
- Sockets (Unix, network)

Outlook

- Message queues
- Shared memory

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Summary and outlook

Summary

- Process communication concept
- Signals
- Sockets (Unix, network)

Outlook

- Message queues
- Shared memory