

# Übung Echtzeitsysteme WS 2014 / 2015

## PThreads

Philipp Heise

### Exercise 0 PThreads

1. (*PThreads*) Read the documentation for *pthread\_create* and *pthread\_join*. Write a program that starts two threads and waits for each thread. Each thread should be passed an id as an argument that is used for printing a message on the command line. The third parameter of *pthread\_create* is the name of a function that returns a void pointer and also has one void pointer as parameter.

### Exercise 1 A bank account and pthread mutex

In the *pthreadmutex.c* file a simple multi-threaded program is given. The program creates two threads and each thread generates a random number that is added to a shared global variable.

- Why does this program not always work? What do you need to do to solve the problem?
- Read the documentation for *pthread\_mutex\_init*, *pthread\_mutex\_lock* and *pthread\_mutex\_unlock*.
- Fix the problem with one global pthread mutex.

### Exercise 2 Dining philosophers

In the lecture we already got to know the famous problem of the dining philosophers. You can download a code skeleton from our homepage. In the provided code the code to pickup the left and right fork is missing.



1. (*Semaphore functions*) Find out what the functions *sem\_init*, *sem\_destroy*, *sem\_post* and *sem\_wait* are for.
2. (*Solve the problem*) Use the semaphores to solve the problem. Explain the deadlock issue and find a way to avoid it.