Lab

Exercise 06 „Platforms for Embedded Systems“ Prof. Cochlovius

- Multiple Threads and the Dining Philosophers -

Goal:
With this exercise you gain some insight into multi-thread programming on host and target and you also solve the problem of the dining philosophers.

Exercise 6.1: The first – and wrong - approach

Create a new C-template project named mutex in Eclipse as a Managed Make project. Import the source code **`mutex.c`**. Get familiar with the POSIX functions used from pthread.h and try to understand their arguments and usage. Now build the project for host and target. What happens during linking? How can we tell the linker to use additional libraries in Eclipse? Which libraries exactly do we need? Run the executable several times on the host and on the target. What do you see?

Exercise 6.2: Now we fix the biggest bugs

Why ist he program executing (sometimes) on the host, but (almost) never on the target? Try to analyze or try to debug. Now fix the issue an re-test the program on both host and target.

Note: you want to keep all variants of your program inside your project. You can copy your program inside the SRC directory, rename it and then remove all other *.c files from the compilation scope to avoid multiple main() functions. This can be done with right mouse button on the filename → resource configurations.

Exercise 6.3: But now we need a working solution

Pls. verify whether the result of exercise 6.2 is really(!) 100%(!) free of deadlocks. This means, there is no dependency whatsoever on a specific timing behavior of the scheduler. What is the underlying issue with 6.2? How can you augment the program (without changing its algorithm) to expose the deadlock for sure? Demonstrate a deadlock both on the host and on the target.

But how can we improve the algorithm of the philosophers accessing their forks? See the link http://web.eecs.utk.edu/~mbeck/classes/cs560/560/notes/Dphil/lecture.html , here you find some additional hints to various solutions.
Note: the „asymmetric solution“ is the easiest and most straightforward alternative suitable for our needs. Understand it and implement it in your program. Re-test again on host and target.