lea.schoenberger [☺] tu-dortmund.de
benjamin.glaeser [☺] tu-dortmund.de
niklas.ueter [☺] tu-dortmund.de
mikail.yayla [☺] tu-dortmund.de

Exercise for the lecture
Embedded Systems
Wintersemester 17/18

Exercise Sheet 10

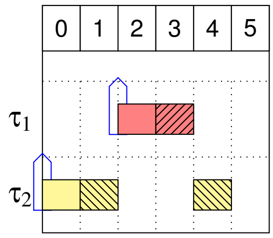
(10 Points)

Lab exercises for the period from Wednesday, 10th January 2018

For this exercise sheet, we use the virtual machine **RTEMS-de**.

Background

The Priority Inversion Protocol discussed in the previous exercise sheet is not a universal solution to deadlock prevention – if multiple semaphores are used in order to protect multiple resources, the occurrence of a deadlock is still possible. An example situation is given in the following:



τ_1 reserves resource A and blocks access to resource B

τ_2 reserves resource B and blocks access to resource A

Two tasks τ_1 and τ_2 as well as two resources A and B shall be given. At $t = 3$, task τ_1 holds resource A and tries to access resource B at $t = 4$. This attempt is blocked since τ_2 holds resource B since $t = 1$ and, in addition, tries to access resource A at $t = 5$. Hence, τ_2 waits for τ_1 to free resource A and τ_1 waits for τ_2 to free resource B – a deadlock. This could be prevented by establishing the rule that resources must be accessed in the same order by each task (e.g., A first, B second) and freed in reverse order. However, such a restriction is not always easy to realize and must be implemented carefully.

The Priority Ceiling Protocol (PCP) offers a solution for this problem by, similarly to the Priority Inheritance Protocol, modifying the priority of a task during the usage of semaphores. For each semaphore must be known which task holds the highest priority. This value is denoted as *Priority Ceiling* and must be determined manually in RTEMS. If a task tries to access an already allocated resource, it is blocked but, in contrast to PIP, no priority modification occurs. Instead, PCP increases the priority of a task to the semaphore's priority ceiling as soon as it successfully holds the respective semaphore. If the task holds a higher priority than the priority ceiling of the semaphore, its priority remains unchanged.

In the above example, the priority ceiling of both semaphores equals the priority of task τ_1. Thus, at $t = 1$, the priority of τ_2 is increased to the priority τ_1, such that τ_2 can finish its execution without any interruption.

Preparation

The hardware is connected as described in exercise sheet 8. To update your copy of RTEMS to the version required for this exercise sheet, please execute the command `rtems-setup` and subsequently `make -j4 install` in the directory `$HOME/rtems/build`.

The program required for this exercise sheet is located in the subdirectory `rtems-gpio/testsuites/samples/blatt10` (subsequently short *blatt10-source*) while the related directory containing the compiled files is `build/`

`arm-rtems4.11/c/raspberrypi/testsuites/samples/blatt10` (short *blatt10-build*). To transfer the compiled program, execute the command `raspbootcom /dev/ttyUSB0 blatt10.ralf` in the *blatt10-build* directory. In case you only modified the program, it is sufficient to execute `make` in *blatt10-build*. If you modified RTEMS, it is necessary to execute `make install` in the directory `$HOME/rtems/build`. Due to technical reasons, it is recommended to additionally modify the file `init.c` of the *blatt10* program and to recompile it as described before.

10.1 Deadlock with Inheritance (4 Points)

The given program uses three tasks with the following parameters:

task	period	arrival time	execution time (total)	semaphore A		semaphore B	
				begin	duration	begin	duration
τ_1	20	2	8	1	6	3	2
τ_2	30	5	3	-	-	-	-
τ_3	40	0	8	3	2	1	6

Please make sure that the *blatt10* program is configured for the Priority Inheritance Protocol, then start it and inspect its behavior by drawing a scheduling diagram.

10.2 Implementation of Priority Ceiling (6 Points)

In general, RTEMS supports the Priority Ceiling Protocol but, unfortunately, the given version lacks of a certain amount of program code such that PCP does not work.

Assignment: Complete the implementation of PCP in RTEMS. Thereon, make sure that the *blatt10* program is configured for PCP, compile RTEMS and *blatt10*, and examine if PCP solves the problem discovered in the previous assignment. How does the scheduling diagram change?

Hints:

- Code lacks only in two files:
  - `rtems-gpio/cpukit/score/src/coremutexsurrender.c`
  - `rtems-gpio/cpukit/score/include/rtems/score/coremuteximpl.h`
- The respective locations are indicated with “fulfil”.
- `_Thread_Raise_priority(A, B);` increases the priority of object A to value B.
- `_Thread_queue_Release( queue, lock );` removes a lock from the queue of a mutex.