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 9

(10 Points)

Lab exercises for the period from Wednesday, 20th December 2017

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

Background

The concept of semaphors should be already known from the operating systems lecture. However, in the context of real-time operating systems they may cause additional problems: Although no deadlock occurs, the usage of semaphors can lead to a task's deadline miss. Obviously, this is not acceptable in a real-time system and may even have hazardous consequences. Hence, special protocols are applied to avoid such incidents. In this exercise we consider an example to demonstrate this particular problem.

Preparation

The hardware is connected as described on exercise sheet 8. To update your copy of RTEMS, execute the command `rtems-setup` and thereon (in the directory `$HOME/rtems/build`) the command `make -j4 install`.

The program necessary for this exercise is located in the subdirectory `rtems-gpio/testsuites/samples/blatt9` (subsequently short *blatt9-source*), while the complied data can be found in `build/arm-rtems4.11/c/raspberrypi/testsuites/samples/blatt9` (short *blatt9-build*). To transmit the compiled program, execute the command `raspbootcom /dev/ttyUSB0 blatt9.ralf` in directory *blatt9-build*. In case you modified the program only, it is sufficient to execute `make` in *blatt9-build*. Otherwise, if you modified RTEMS, execute `make install` in `$HOME/rtems/build`. In the course of this, possible modifications of this exercise's program are compiled as well.

9.1 Reindeer Feeding (2 Points)

Its Christmas time and in Santa Claus' house everyone is busy with preparations. All gifts are wrapped, carefully listed and stored in the festively decorated sleigh. After storing some Christmas cookies as well (Santa should not be hungry during his travel), there is only one task left before the journey begins: The reindeers must be feeded so that they have sufficient energy to gallop all night in order to distribute Santa's gifts all over the world.

Santa Claus owns nine reindeers of which five (Dasher, Dancer, Prancer, Vixen, Comet) have already been feeded. Now the remaining four reindeers, Cupid, Dunder, Blixem, and Rudolph, must be feeded. These are abstractly represented as four tasks (`task1.c` to `task4.c`). While for Cupid and Blixem standard fodder is sufficient, Dunder and Rudolph need special nutrient-rich feed, which is served in a separate manger, which can be accessed by only one reindeer at each point in time. For the purpose of preventing the two reindeers from an exhausting fight for food, the access to the special feed is controlled by means of semaphores. The manger as a shared resource is symbolized by the white LED H downright, next to the red button on the Pibrella board. This glows whenever a task holds the semaphore, i.e., while either Dunder or Rudolph satisfy their hunger.

The reindeers' eating behavior can be expressed as follows:

task	period	arrival time	execution time	critical section	
				begin	duration
τ_1	8	4	1	-	-
τ_2	16	2	4	1	1
τ_3	40	5	3	-	-
τ_4	50	0	12	1	10

Assignment: Execute the program and sketch the resulting scheduling diagram.

9.2 Program Modification (3 Points)

Modify the program such that Priority Inheritance Protocol is used for the semaphore. Test the modified program and sketch the resulting scheduling program.

9.3 Operating System Modification (5 Points)

One could have the idea to implement a simplified version of priority inheritance: If a task attempts to occupy an already occupied semaphore, the priority of the first task could be increased to the possible maximum rather than to the second task's priority level. In the context of our example program, task 4 (Rudolph) would not receive the priority of task 2 (Dunder) but a fixed, predefined priority. However, this simplified protocol should not be used, as we will see in the following.

Assignment: Modify the RTEMS operating system such that the task priority is not increased to the second task's priority level but to the maximum possible priority. Examine the behavior of the program resulting from the modification and draw the respective scheduling diagram. Where does a problem occur? Does the system detect this problem?

Hints: The modification of the task priority under the priority inheritance protocol can be found in the file `rtems-gpio/cpukit/score/src/coremutexseize.c`. We recommend a fixed maximum priority value of "1". Since you modify the operating system itself, you have to compile not only the example program but also the operating system (which will not take much time). For this purpose, execute the command `make install` in the directory `$HOME/rtems/build`. This will also automatically compile the example program.

The Embedded Systems lecture team wishes you
a merry Christmas and a happy new year!

General information: An overview about the exercise sessions as well as further information can be found on `https://ls12-www.cs.tu-dortmund.de/daes/de/lehre/lehrveranstaltungen/wintersemester-20172018/es-1718.html`. The exercise sheets will usually be published on the course website on Mondays and will be solved during the respective exercise sessions. The exercises are divided into two parts, in each of which at least 50% of the points must be achieved in order to receive the exam admission.