RTOS Lab 2014 #2

M.Platzner, L.Thiele

Assignment 4 (More on Task Synchronization) Implement an application that simulates the famous "Dining Philosophers" problem shown in Figure 1: In this example, five philosophers sit around a table. In front of each philosopher is a plate with spaghetti. To the left and right of the plate is one fork. In order to eat, a philosopher needs to use two forks (the one to the left and the one to the right). Philosophers mostly think but at certain times they get hungry and want to eat.

Figure 1: Dining Philosophers

In your application, each philosopher should be modeled by a task, and each fork should correspond to a semaphore. Structure the code for a philosopher as follows:

```
        ...
(1)     think for 50 time ticks
(2)     grab the first fork
(3)     wait for some time X
(4)     grab the second fork
(5)     eat for 10 time ticks
(6)     put back the forks
        ....
```

Write your application such that it works with a variable number of philosophers. When the application starts, the following parameters should be entered:

- overall simulation time in seconds
- the number of philosophers
- the waiting time X in time ticks

All tasks should have equal priority and time slicing should be activated with a time slice of 6 time ticks. After the simulation stops, display the number of times each philosopher has actually eaten.

The output of the application should look similar to:

```
Enter number of philosophers [3-20]: 5
Enter waiting time [1-100 ticks]: 20
Enter overall simulation time [1-100 s]: 3

Simulating 5 philosophers with waiting time 20 ticks for 3 seconds...

Philosopher 0 - start thinking.
Philosopher 1 - start thinking.
Philosopher 2 - start thinking.
Philosopher 3 - start thinking.
Philosopher 4 - start thinking.
Philosopher 3 - start eating.
Philosopher 3 - start thinking.
Philosopher 4 - start eating.
Philosopher 1 - start eating.
Philosopher 4 - start thinking.
Philosopher 1 - start thinking.
Philosopher 2 - start eating.
Philosopher 2 - start thinking.
Philosopher 0 - start eating.
Philosopher 0 - start thinking.
Philosopher 3 - start eating.
Philosopher 3 - start thinking.
Philosopher 1 - start eating.
Philosopher 1 - start thinking.
Philosopher 4 - start eating.
Philosopher 4 - start thinking.
Philosopher 2 - start eating.
Philosopher 0 - start eating.
Philosopher 2 - start thinking.
Philosopher 0 - start thinking.
Philosopher 3 - start eating.
Philosopher 3 - start thinking.

All philosophers stopped.
Eat counters: 2 2 2 3 2
```

Implement the application such that the "fork access protocol" is *deadlock-free* and, additionally, try to achieve *fairness*.

■