703013-4 PS Operating Systems
Exam – June 27th 2018, 16:30-18:15

General information: Talking, chatting, texting or any other form of live human-human communication is not allowed and may lead to a negative grade. However, you are allowed to use all other offline and online resources such as previous homework solutions, manpages, Stackoverflow, etc.

Note: Please make sure that

- the first line of your source code explains how to compile and run your application (alternatively you can also create a Makefile or shell script),
- your code compiles properly without any errors or warnings,
- your code is well-commented (explaining why e.g. a particular function is called),
- your program performs proper cleanups of any resources it acquires,
- you submit your solution before 18:15 in OLAT (submission will be automatically closed!)

The task below has a total of 10 Points, deviating from the requirements above may reduce the number of points you get. A minimum of 5 points is required for a positive exam grade.

Task

In an automotive system, synchronization between tasks (threads) is extremely critical since dead-locks and live-locks (starvation) could lead to a catastrophic consequence. You as a programmer are asked to write a code for Electronic Stability Program (ESP) as one of the essential components of modern cars.

The ESP system includes three different resources listed in the table below and *N* threads where half of the threads (from 0 to $\frac{N}{2} - 1$) belong to the Braking Control (BC) group, and the other half of the threads (N/2 to N-1) belong to the Steering Control (SC) group.

The threads belonging to the BC group need both resource A and B at the same time to do their functionality, while the threads belonging to the SC group need to get both resource B and C at the same time to do their functionality. Let's assume that the order of requests for acquiring the resources by the threads are randomly. For example, if a thread needs resource A and B, it can either ask for resource A and then B or first ask for resource B and then A.

Resource_id	No. of instances
A	1
B	2
C	1

Let's also assume that all the threads are periodic with a known period. The threads of the BC group have a period = 150 msec, in the sense that each thread after performing its functionality and releasing the resources, sleeps 150 msec and executes again. The threads of the second group have the period 100 msec.

Write a program for this system with no deadlock and starvation.

Note:

- *N* is an integer values is given by the user.
- To simulate a functionality of a thread, we only print on the screen that “the thread *thread_id* is performing its functionality” and wait 20 msec.
- If a thread gets the first resource but the second resource is not available it should print out on the screen that “the thread *thread_id* acquires the resource *resource_id* while the resource *resource_id* is not available”.
- To generate the random order of acquiring the resources in a thread, you can easily generate a random binary value, if it is zero, ask for the first resource and then the second resource; otherwise (i.e., if the random value is one) firstly ask for second resource and then the first resource.
- We would like to see the output of the system in one hyper-period. The hyper-period is the Least Common Multiple (LCM) of the periods of the threads which in our system is 300 msec, in other words each thread of the first group is run 2 times while the threads of the second group is executed 3 times.