Course name: **Operating systems / Besturingssystemen**

# FIRST LAB ASSIGNEMENT (updated)

DEADLINE: 16.12.2009

Using "task" and "task_queue", develop a program where N producers create and enqueue tasks, and M consumers dequeue such tasks. Producers and consumers must be implemented using threads, and access to the shared task_queue must be "safe". Only one producer/consumer is allowed to access the task_queue at a given moment. Tasks can have different priorities, and for a new task the priority is randomly chosen between a range [1..P].
N, M, P can be specified by the user through the command line. For instance, by issuing ***"./assignment1 -p 3 -c 5 -P 5"***, 3 producer and 5 consumer threads will be created, and tasks will have at most 5 possible priority values. If a parameter is not specified, a standard value is chosen (2 producers, 3 consumers and 3 priority values). Maximal number of threads in total should not exceed 50. There should not be more than 100 levels of priorities. Tasks with higher priority should be executed before the ones with lower priority (the highest priority is priority P).
Once a task is created or consumed, the thread will sleep for a number of milliseconds. A producer sleeps for at most (((task->priority)/P) * 300) ms, but never less than 100 ms. A consumer sleeps for at most (((task->priority)/P)* 600) ms, but never less than 200ms.
To terminate the program, the user must issue a KILL signal to the process ("CTRL+C" inside the terminal, or "kill -s SIGINT pid").
However, the program is required to catch the signal in order to gracefully release resources (stop threads and free dynamic memory).

**Notes:**

- Assignments should be handed in on blackboard as archive packages containing program code + .pdf file with documentation.
- Name of the archive package should be e.g. *"Group013_LabAssignment_1.zip"*
- Documentation should contain important information describing your solution. As an example, it should contain: explanation of data structures used, assumptions taken during work, what were possible solutions in particular implementation and why was a specific way chosen, constraints of your solution, instructions for compiling/running the code etc…
- The style of the code written will influence your grade also.
- Commented code is more appreciated.
- Code **has** to be working when compiled with regular ***gcc***.
- Make sure to mention your student numbers in the documentation.
- Respect the deadline.
- Only one group member should submit solution package on the blackboard.