System and device programming
14 September 2011

(C program: textbooks and/or course material allowed)

(18 marks) The final mark is the sum of the 1st > 8 and the 2nd part > 10

The final mark cannot be refused, it will be registered (no retry for marks >= 18)

Write a concurrent C program in the **UNIX** environment which creates **`K`** threads (**`K`** given as the first command line argument).

Each thread instance belongs to a randomly selected class among *C1, C2,* and *C3*. The index of the class *i* defines the priority class of the thread, 3 being the maximum priority.

Priority means that when a thread finishes using its critical region, and there are one or more threads waiting to enter their critical regions, **the waiting thread with highest priority has to be unblocked**. The threads must implement this rule using normal semaphores and global variables.

A thread **cyclically** makes a request to access its critical region, and waits a random time (between 2 and 10 seconds) before issuing the next access request.

The thread sends on a pipe, shared with the main thread, its relevant status information, i.e. "I'm requesting access to my critical region, or I'm inside my critical region". Thus the status is defined as the triple

`<thread_identifier, priority_class, N>`

where **`N`** is an integer (**`N=0`** when the thread is requesting access to its critical region, **`N=1`** the thread is inside its critical region).

The main thread waits on the pipe the sent information sent by the created threads and **prints the received information and the current time in a log file** (name given as the second argument of the command line), so that one can assess the correct behaviour of the concurrent threads.