A Comment on S. Kang’s and H. Lee’s paper on „Analysis and Solution of Non-Preemptive Policies for Scheduling Readers and Writers“ (OSR 32(2))

Winfried E. Kühnhauser
winfried.kuehnhauser@gmd.de
GMD National Research Center For Information Technology, Sankt Augustin, Germany

In their paper Analysis and Solution of Non-Preemptive Policies for Scheduling Readers and Writers, S. Kang and H. Lee presented several known and also new and interesting scheduling policies to solve the reader/writer problem. They also specified corresponding algorithms, and it is some of these implementations that my comments relate to.

Kang and Lee assume that their scheduling procedures (EnterReader, ExitReader, EnterWriter, and ExitWrite) are atomic. If we do not assume some invisible mechanism in the environment of these procedures (such as procedures within Hoare’s monitors posses), atomicity has to be implemented explicitly.

In their algorithms, Kang and Lee obviously dedicated semaphor Mutex to that task. Under this assumption, some of the algorithms fail to solve the traditional problem of combining the testing of a condition and a corresponding reaction into a single atomic action. The problem is as follows.

Let us, for example, look at the implementation of EnterReader on page 37. Assuming that a new reader arrives while some writer holds the buffers and no other reader is waiting, we proceed to the else-part of the condition. After executing V(Mutex), the condition that put us into the else-part may no longer hold, because V(Mutex) may immediately result in the current writer to become active. The current writer may then exit and test for waiting processes. It will find none, because the entering reader was suspended before executing P(RWSem,READER). After the new reader gains back control, it now will execute P(RWSem,READER) which will block the reader for ever.

A common solution to this problem are, as mentioned above, Hoare’s monitors which in turn are usually implemented on a lower abstraction level by globally disabling processor interrupts or by basic test&set machine instructions.