

### 3. Übung IBN

Maurice Donner      Ise Glade

May 16, 2022

---

## Aufgabe 1

Using spinlocks on a single core unit wouldn't make much sense, as a critical section cannot be accessed by two processes at the same time. Therefore there are no race conditions. Locking the only available processor on a single core (remember we are talking about spinlocks) might not be the best idea either.

## Aufgabe 2

- a) `wait()` unblocks a process, if `S` is larger than 0.  
`signal()` increases `S` by 1  
If those two instructions were to be exchanged, the critical section would be executed not regarding if there is a queue, since the instruction to wait has not been given yet.
- b) If  $S \geq 2$ , Two processes would be allowed to be executed at the same time, or, if  $S < 2$ , just one process would be started, never giving another process the signal to start.
- c) If `wait()` is missing, the critical section will just be executed without checking if there is a queue.  
If `signal()` is missing, only `S` processes can be executed, as there is no way to increase `S` after a process has finished.

## Aufgabe 3

- a) The Assertion can be entered by executing the code of Thread A twice, then executing the code of Thread B, before executing Thread A again. This way, the condition `i == 5` is fulfilled, and the error statement can be executed. Assuming `Debug_Assert(false)` is some piece of code that shall not be executed, this is unwanted, and will only happen if the threads run in a specific order.

## Aufgabe 4

The pseudocode can be found in `pseudo_4.cpp`

- b) The `ManualResetEventSlim` is an object with a single boolean flag and 3 methods: `Wait()`, `Set()` and `Reset()`. When a thread calls `Wait()`, it can only continue, once the reset event is signaled. This way one can implement the dependency of one thread to another, by only giving the first `Set()` signal when a certain part of the program has been run.

To access the `Debug_Assert()` statement, we can use the fact, that thread B doesn't have a `Wait()` condition. After setting the reset event for the first time (`i=2`), the critical region can be entered by thread A, and then immediately stopped before the `if`-condition is reached. Thread B can then increment the counter once again, so that the condition `i%2==1` is fulfilled. If we switch now back to thread A, the `Debug_Assert` can be executed.

---

## Aufgabe 5

## Aufgabe 6

- a) A Mutex has separate lock and condition variables. Therefore, condition variables have no history. The condition has to be tested separately, instead of relying on a signal. Semaphores will remember the Signals given through the Semaphore counter `S`. If a Thread broadcasts a signal, the next time another Thread calls `wait()`, it will start running immediately, regardless of when the signal was given.
- b) The Mutex has to hold the lock, in order for processes not to get stuck waiting. This can happen, when a wait function is called first, then a signal runs between the time, where the wait checked for a signal, and the condition. The Thread will not see that a signal has been called, and wait forever.

## Aufgabe 7

Memory Task	Long term memory Stores fundamental concepts	Working memory Stores currently needed things
Comparison	Random Access Memory (RAM)	Cache
Speed	Slow	Fast
Volatility	Long storage duration	short storage duration
Unit	Junks / Items	Junks / Items
Size	$10^9 \text{ Bytes}$	$10^2 \text{ Bytes}$

- 
- Memory can be moved from the working memory into the long term memory by repeating the information several times. This works especially well for memorizing vocabulary. Studying everything on one day is generally less efficient than repeating the vocabulary several times during the week.