Rene Sorger - se22m015 25.02.2023

Gregor Zeman - se22m048

# What are the necessary conditions for deadlocks (discussed in the lecture) [0.5 points]?

Mutual Exclusion: At least one resource must be held in a non-sharable mode. This means that only one process can use the resource at any given time.

Hold and Wait: A process must be holding at least one resource and waiting for another resource to be freed by another process.

No Preemption: A resource cannot be forcibly taken away from a process that is holding it. It can only be released voluntarily by the process holding it.

Circular Wait: A set of processes is waiting for each other in a circular chain. For example, process A is waiting for a resource held by process B, process B is waiting for a resource held by process C, and process C is waiting for a resource held by process A.

# Why does the initial solution lead to a deadlock

Because it takes the first fork and then waits for the second fork, it is possible that another philosopher takes the second fork and then waits for the first fork. This will result in a deadlock.

# Does this strategy resolve the deadlock and why?

No, this strategy, in fact, creates a deadlock by trying to make 2 philosophers pick up the same fork at once.

# Measure the total time spent in waiting for forks and compare it to the total runtime:

As expected, the waiting time was far higher than the actual runtime, since multiple threads need to wait for free resources to prevent a deadlock.

**Total elapsed time: 18,22 seconds**

**Total waited time: 26,16 seconds**

# Interpret the measurement - Was the result expected?

The more shared resources are used, the more likely it is that more time is spent by threads waiting for these shared resources to be freed at any given moment.

# Can you think of other techniques for deadlock prevention?

Make resources inaccessible if processes are trying to access them simultaneously through mutual exclusion.

Make a process access one resource while waiting for another resource to get freed in order to access the desired resource. (Hold and wait)

# Make sure to always shutdown the program cooperatively and to always cleanup all allocated resources.

C# Garbage Collector, my beloved 💔 💔