EECS 678 - Lab 07

Theodore Lindsey

April 24, 2015

`http://people.eecs.ku.edu/~vvivekan/lab/pthreads_dp/pthreads_dp.html`

1. *Describe the asymmetric solution. How does the asymmetric solution guarantee the philosophers never enter a deadlocked state?*

   The asymmetric solution forces any even numbered philosopher to try to grab a left-side chopstick first and any odd numbered philosopher to try to grad a right-side chopstick first. In other words, philosophers who sit next to each other will both be trying to grab the same chopstick first. When one gets the chopstick between them, the other waits for it to be free. This leaves half of the chopsticks unclaimed and avalible so that when the philosophers who got chopsticks look on the other side, they find one.

2. *Does the asymmetric solution prevent starvation? Explain.*

   No, it does not prevent starvation. It is possible that a philosopher never finds the initial chopstick free when he looks for it. As such, he won't ever be able to eat because the condition for picking up the next chopstick and then eating is never met.

3. *Describe the waiter's solution. How does the waiter's solution guarantee the philosophers never enter a deadlocked state?*

   In this case, the waiter is left to distribute chopsticks. The waiter ensures that both chopsticks are available before distributing them both at the same time to a single philosopher. Because the waiter checks for both chopsticks at once (and prevents others from looking while checking), he has exclusive control.

4. *Does the waiter's solution prevent starvation? Explain.*

   Yes, it does. Rather than blindly checking at times when the chopstick may not be available, it instead is signaled as soon as the chopstick it is waiting on becomes available so it can lay claim to it.

5. *Consider a scenario under a condition variable based solution where a philosopher determines at the time it frees its chopsticks that both chopsticks of another philosopher (Phil) it shares with are free, and so it sends the (possibly) waiting Phil a signal. Under what circumstances may Phil find that both of its chopsticks are NOT free when it checks?*

   If a while loop conditioned on the chopstick availability isn't used to continually wait for the signal, it is possible that between the time that Phil receives the signal and goes to lock both chopsticks, the chopsticks will be grabbed by another more quick philosopher (eg if the CPU scheduling does not favor Phil).