1.5 Lecture Summary

1.5 Dining Philosophers Problem

Lecture Summary: In this lecture, we studied a classical concurrent programming example that is referred to as the *Dining Philosophers Problem*. In this problem, there are five threads, each of which models a "philosopher" that repeatedly performs a sequence of actions which include *think, pick up chopsticks, eat*, and *put down chopsticks*.

First, we examined a solution to this problem using structured locks, and demonstrated how this solution could lead to a deadlock scenario (but not livelock). Second, we examined a solution using unstructured locks with `tryLock()` and `unlock()` operations that never block, and demonstrated how this solution could lead to a livelock scenario (but not deadlock). Finally, we observed how a simple modification to the first solution with structured locks, in which one philosopher picks up their right chopstick and their left, while the others pick up their left chopstick first and then their right, can guarantee an absence of deadlock.

Optional Reading:

1. Wikipedia article on the Dining Philosophers Problem

Mark as completed