3.4 Lecture Summary

3.4 Producer-Consumer Problem with Unbounded Buffer

Lecture Summary: In this lecture, we studied the *producer-consumer* pattern in concurrent programming which is used to solve the following classical problem: how can we safely coordinate accesses by multiple producer tasks, P_1, P_2, P_3, ... and multiple consumer tasks, C_1, C_2, C_3, ... to a shared buffer of *unbounded size* without giving up any concurrency? Part of the reason that this problem can be challenging is that we cannot assume any a priori knowledge about the rate at which different tasks produce and consume items in the buffer. While it is possible to solve this problem by using locks with wait-notify operations or by using object-based isolation, both approaches will require low-level concurrent programming techniques to ensure correctness and maximum performance. Instead, a more elegant solution can be achieved by using actors as follows.

The key idea behind any actor-based solution is to think of all objects involved in the concurrent program as actors, which in this case implies that producer tasks, consumer tasks, and the shared buffer should all be implemented as actors. The next step is to establish the communication protocols among the actors. A producer actor can simply send a message to the buffer actor whenever it has an item to produce. The protocol for consumer actors is a bit more complicated. Our solution requires a consumer actor to send a message to the buffer actor whenever it is ready to process an item. Thus, whenever the buffer actor receives a message from a producer, it knows which consumers are ready to process items and can forward the produced item to any one of them. Thus, with the actor model, all concurrent interactions involving the buffer can be encoded in messages, instead of using locks or isolated statements.

Mark as completed