Solving the Dining Philosophers Problem Using Locked Queue Strategy

Umut Azazi 190315071

Onur Tiriş 190315085

Introduction

The Dining Philosophers Problem is a classic example in computer science, illustrating synchronization issues in concurrent algorithm design. It involves a scenario where five philosophers sit around a table and do two things: think and eat. However, they share a limited number of forks (five, one between each pair of adjacent philosophers) and must use two forks to eat. This report discusses an implementation of a solution using the Locked Queue method in Python, preventing deadlock and ensuring fair resource allocation among philosophers.

Problem Description

In the problem, five philosophers (who do nothing but think and eat) sit around a circular table. A fork is placed between each pair of adjacent philosophers, making up five forks in total. Each philosopher can either think or eat. However, to eat, they need to use the forks on their immediate left and right, which leads to a synchronization problem if more than one philosopher picks up a fork at the same time.

Solution Overview

The Locked Queue method provides a structured way to avoid deadlock in the dining philosophers problem. This approach uses a queue to control the order of fork acquisition by philosophers. Each philosopher is represented as a thread, and they request access to their left and right forks. If unable to acquire both, the philosopher waits, adding themselves to the queue. This queue ensures that resources (forks) are allocated in an orderly manner, preventing the deadlock scenario.

Implementation Details

The implementation in Python involves defining classes for Philosophers and Forks, along with a LockedQueue class. Each Philosopher attempts to pick up the forks using a locking mechanism provided by the Fork class. The LockedQueue class manages the order of fork requests, ensuring that a philosopher is only able to eat if there is no contention for the forks they require. This system thus makes sure that all philosophers get a fair chance to eat without leading to a deadlock.

Conclusion

The Locked Queue method effectively resolves the deadlock problem in the Dining Philosophers scenario. It demonstrates a practical application of resource allocation and synchronization in concurrent programming. By implementing this solution, each philosopher can eat and think without indefinitely waiting for resources, ensuring fairness and efficiency in the system.