ሰኞ ፥ ሰኔ 30 2017
Mon, 7 July 2025

Dining Philosophers Project

1. Synchronization Strategy

The Dining Philosophers problem is implemented using Pthreads, mutexes, and condition variables. Each philosopher is represented by a thread that alternates between thinking and eating. A global mutex ensures mutual exclusion when modifying a shared state, and a separate condition variable is assigned to each philosopher.

When a philosopher becomes hungry, they invoke pickup_forks(int id) to request the two forks adjacent to them. This function checks whether both forks are available using the test() method. If available, the philosopher proceeds to eat; otherwise, the thread waits on its condition variable. After eating, the philosopher calls return_forks(int id) to mark forks as available and signal neighbors who may now be able to eat.

This strategy prevents deadlock by ensuring that a philosopher can only eat if both neighbors are not eating. It also prevents starvation by checking and signaling neighbors after releasing forks.

2. Deadlock and Starvation Prevention

Deadlock Prevention:

Deadlock is avoided by preventing a circular wait. A philosopher only proceeds if both adjacent forks are available, which ensures no thread holds one fork while waiting for the other indefinitely.

Starvation Prevention:

After finishing eating and releasing forks, each philosopher immediately tests whether either neighbor can proceed. This signaling mechanism gives every philosopher a fair opportunity to eat and prevents indefinite blocking.