Dining Philosophers Problem With Chandy/Misra Solution

Report

Batuhan YAVUZ – 190316023

Mert BOYAOĞLU – 190316056

Eren ERSOYLUOĞLU - 190316057

Abdullah Bora ÖZIŞIK - 190316018

Overview:

This comprehensive report combines insights from three distinct implementations of the Dining Philosophers Problem in Python. Each solution addresses the challenges of synchronization and resource sharing among philosophers in a multi-process environment. The implementations showcase different strategies, including traditional threading, visualizations using Matplotlib, and the Chandy/Misra token-based approach.

Common Components:

Fork Class:

- Represents a fork on the dining table.
- Methods include pick up, put down, clean, and dirty.
- Utilizes the context manager protocol for safe resource acquisition and release.
- Chandy/Misra solution introduces a cleanliness attribute.

Philosopher Class:

- Extends the threading.Thread class.
- Represents a philosopher engaging in thinking, eating, and managing forks.
- Utilizes locks for exclusive access to forks.
- In the Chandy/Misra solution, philosophers have a token (threading event) for controlled fork access.

Animated Visualization (animated_table Function):

- Uses Matplotlib to create a dynamic representation of the dining philosophers scenario.
- Visualizes philosophers, forks, and their states.
- Color-coded elements indicate the eating state and cleanliness of forks.
- Incorporates the token passing mechanism in the Chandy/Misra solution.

Textual Representation (table Function):

- Provides a real-time textual representation of the dining table.
- Displays the state of each philosopher, forks, and additional information (e.g., cleanliness in the Chandy/Misra solution).
- Updates continuously for monitoring the dining process.

Main Function:

- Initializes necessary components, including philosophers, forks, and tokens.
- Starts threads for each philosopher and visualization functions.
- Ensures proper joining of philosopher threads.

Execution Flow:

Traditional Threading Implementation:

- Philosophers cycle through thinking, requesting forks, eating, and putting down forks.
- Threads, locks, and a token event ensure mutual exclusion and fair access to resources.

Threading with Animated Visualization:

- Utilizes Matplotlib for dynamic visualization.
- Provides a graphical representation of philosopher interactions and system behavior.
- Threads synchronize through locks and a token event.

Chandy/Misra Token-Based Approach:

- Introduces a distributed and deadlock-free solution.
- Philosophers use a token for controlled fork access, passing it to the next philosopher after eating.
- Threads and locks ensure exclusive access to forks.

Conclusion:

The integrated report highlights the versatility of solutions to the Dining Philosophers Problem, combining traditional threading, visualizations, and a token-based approach. The visualizations aid in understanding the concurrent execution and interactions among philosophers, forks, and the overall system. The Chandy/Misra solution further enhances synchronization and coordination in a distributed system. Future enhancements could explore parameter variations and additional dynamic features. Overall, the code demonstrates well-structured solutions with informative visualizations to address the challenges posed by the Dining Philosophers Problem.