National University of Computer & Emerging Sciences (FAST-NU)

Operating Systems Project Report

Dining Philosophers Problem

Instructor
Miss Naz Memon

Section: BSE-4B

Team Members:
21K-3922 Mirza Uzair Baig
21K-3895 Sharib Khan
21K-3890 Ahmed Salar

April 2023

Dining Philosophers Problem

1.Introduction

In computer science, the dining philosopher’s problem is an example problem often used in concurrent algorithm design to illustrate synchronization issues and techniques for resolving them. The dining-philosophers problem is considered a classic synchronization problem neither because of its practical importance nor because computer scientists dislike philosophers but because it is an example of a large class of concurrency-control problems. It was originally formulated in 1965 by Edsger Dijkstra as a student exam exercise, presented in terms of computers competing for access to tape drive peripherals. Soon after, Tony Hoare gave the problem its present formulation.

2.Problem Statement

Five silent philosophers sit at a round table around a bowl of spaghetti. chopsticks are placed between each pair of adjacent philosophers. Each philosopher must alternately think and eat. However, a philosopher can only eat spaghetti when they have both left and right chopsticks . Each chopstick can be held by only one philosopher and so a philosopher can use the chopstick only if it is not being used by another philosopher. After an individual philosopher finishes eating , they need to put down both forks so that the forks become available to others. A philosopher can take the fork on their right or the one on their left as they become available, but cannot start eating before getting both chopsticks . Eating is not limited by the remaining amounts of spaghetti or stomach space; an infinite supply and an infinite demand are assumed. The problem is how to design a discipline of behaviour (a concurrent algorithm) such that no philosopher will starve; i.e., each can forever continue to alternate between eating and thinking, assuming that no philosopher can know when others may want to eat or think .

3.Features And Solution

We used Semaphore and threads to demonstrate and solve the problem One simple solution is to represent each chopstick with a semaphore. A philosopher tries to grab a chopstick by executing a wait() operation on that semaphore. She releases her chopsticks by executing the signal() operation on the appropriate semaphores.

- ALGORITHM FOR OUR CODE

1. Define the number of philosophers
2. Declare one thread per philosopher
3. Declare one mutex(represent chopsticks) per philosopher
4. When a philosopher is hungry
5. See if chopsticks on both sides are free
6. acquire chopsticks
7. eat
8. restore the chopsticks
9. If chopsticks aren’t free
10. wait till they’re available
11. repeat endlessly

Technology Used:

Programming Language: C language
Platform: Ubuntu 22.04

4.Results

Getting rid of deadlock by properly synchronizing . Meeting need to allocate several resources among several processes in a deadlock-free and starvation-free manner. The main Objective of this project was to learn the use of semaphore and to understand the concept of deadlock and way to resolve this problem.

5.References

Abraham silberschatz operating system concepts Wikipedia.

6.Acknowledgement

We would like to thank Ma'am Naz Memon faculty of FAST NUCES karachi (https://pk.linkedin.com/in/nazmemon), for assigning this project to us. It was because of her lectures and guidance that We are able to complete this project.