Dining Philosophers

Name - Kunal Sharma
Roll Number - 2021331
Branch - CSD

Code Description

A.) 1. Starting from the main() function I have created 5 threads which represent 5 philosophers. The primitive used is mutex lock. I am initializing the mutex using pthread_mutex_init(). Then I am creating 5 threads and then using pthread_join().The resources used are 5 forks.In this i have strict ordering that all the philosophers will pick their right fork first then left. I am breaking the deadlock by doing that the last philosopher will pick his left fork first than right fork.

2. I've generated 5 threads starting with the main() method to represent 5 philosophers. Semaphore is the primitive employed. Using sem_init(), I am initializing the Semaphore using sem_init(). I then use pthread_create() to create 5 new threads. Five Semaphore forks are the resources used. I have a rigorous rule in this that all philosophers must choose their right fork before choosing their left. By forcing the final philosopher to choose his left fork before choosing his right fork, I am breaking the deadlock.

B.) 1. To symbolize 5 philosophers, I generated 5 threads, each of which began with the main() method. It uses a primitive called mutex lock. Using sem init, I am initializing the Semaphore (). I then initialize the sauce bowls to 2 and use pthread_create() to create 5 new threads. The tools used are five Semaphore forks and sauce dishes. All philosophers must pick their right fork before picking their left, according to my strict rule in this regard. I solve the impasse by making the final philosopher choose his left fork before his right fork.

2. I've generated 5 threads starting with the main() method to represent 5 philosophers. Semaphore is the primitive employed. The resources used are Semaphore forks and semaphore sauce bowls. By forcing the final philosopher to choose his left fork before choosing his right fork, I am breaking the deadlock.