21AIE202 Operating Systems (2022 Odd)

Lab Exercise 6

Dining Philosopher's Problem
Duration : 1 hour

All the related files can be accessed from our Sharepoint course page.

Consider two processes P1 and P2 executing simultaneously, while trying to access the same resource R1, this raises the question who will get the resource and when? This problem is solved using process synchronisation.

The act of synchronising process execution such that no two processes have access to the same associated data and resources is referred as process synchronisation in operating systems.

It's particularly critical in a multi-process system where multiple processes are executing at the same time and trying to access the very same shared resource or data.

Dining Philosophers Problem is a typical example of limitations in process synchronisation in systems with multiple processes and limited resource. According to the Dining Philosopher Problem, assume there are K philosophers seated around a circular table, each with one chopstick between them. This means, that a philosopher can eat only if he/she can pick up both the chopsticks next to him/her. One of the adjacent followers may take up one of the chopsticks, but not both.

Open and understand dining_philosopher_1.c file, in which you create threads for each philosopher to pick up, eat, and then keep back the chopstick. Try to fill in the code – where you use mutex to lock access to the chopsticks. Run and get the outputs. Is there any deadlock occuring ?

Open and modify dining_philosopher_2.c – In this file, we have introduced a sleep(1) function right after locking a mutex. This will force the program to quickly show a deadlock situation. Run and understand the output. Are you able to figure out that there is a deadlock ?

Question: Modify dining_philosopher_2.c – so that you avoid deadlock from occuring ? (Of course, not by taking out the sleep(1) statement) Try to think about solutions to this problem, and implement it in the code to see if your solution works.

By doing this lab you are expected to have the following skillset.

Learning objectives:

1. Code the dining philosopher’s (DP) problem
2. Identify the problem of deadlocks in (DP) solution
3. Prevent deadlocks in DP code
4. Run test cases to simulate deadlock conditions and their solutions