CSE231 – Operating Systems
Assignment 4
Writeup

Name : Ansh Arora
Roll No.: 2019022
Task : Create your own semaphore using mutex locks and conditional variables, and solve the modified diner philosopher problem.

Functionality
The semaphore that I have made, my_semaphore has 3 main attributes, the count which maintains the integer value of the semaphore, the flaglock, which is a mutex lock that locks the semaphore when the value goes below 1, and the conditional variable that is used to put threads to sleep as they queue for any particular shared variable.
The attributes made are –
1) my_wait(*my_semaphore) – Subtracts the count of the seamphore by 1. If the count reaches 0, it locks the incoming threads until the count goes above 0 again.
2) my_signal(*my_semaphore) – Increases the count of the semaphore by 1. If there are any threads in sleep, they are also signalled using the pthread_cond_signal method.
3) my_signal_printvalue(*my_semaphore) – Made for debugging, alongside increasing the count also displays the count of the semaphore at that particular point of time.
4) sem_init – Used to initialize the declared semaphore.

Implementation
For the dining philosopher's problem, a total of 8 semaphores have been declared – 1 counting semaphore for the current philosophers allowed on the dining table(to avoid deadlocks, because if there are 4 philosophers and 5 forks, deadlock will not occur), 5 binary semaphores - 1 for each fork, and 2 binary semaphores one for each sauce bowl. In the philosopher thread, the critical section is first marked by the roomlock semaphore, which only allows at max 4 philosophers at any point of time. The philosopher then tries to acquire the fork on his left/right using their own specific semaphores wait command. If the philosopher is able to acquire both forks, they are then sent to the eat state, where they try to acquire both the saucebowls in a similar fashion. One of the two saucebowls has been nested in the other's critical section to avoid any deadlocks. Once the philosopher is done eating, they signal to release the left and right forks, and leave the room using the roomlock semaphore signal. This process continues till the noOfEats allowed using the given variable.
Non-Blocking – The locks are replaced with trylocks and cond_waits are removed.