Bonus Assignment

Dining Philosophers

Logic and Working of the Program(s)

In the program, there are 5 philosophers in a round table and 5 forks. Each of them has been numbered from 0 to 4 respectively. A fork on the philosophers left would have the number 'i' and a fork on his right would have the number (i+1)%5. To solve the dining philosopher's problem, I have made use of semaphores as the synchronization primitive.

Part 1: To solve the problem with 5 forks and 5 philosophers, where each philosopher needs 2 fork in order to eat, I have applied the following logic. Each philosopher would first wait for the left fork to be available, then pick up his left fork. After that, similarly, the philosophers would wait for the right fork to be available and pick it whenever it is available. This logic has been implemented with the help of a binary semaphore, that is defined in the code for each fork. Hence, the sem_wait() decrements the count of the semaphore and it becomes 0, which enables the fork to not be used by anyone else (since they would have to wait for the semaphore count to again become 1). Similarly, sem_post() would increase the semaphore count, enabling others to use the fork.

After both forks have been picked up, the philosopher can start and eat. After he finishes eating, he drops both forks that he had picked up and releases it.

There is a deadlock in the above solution, however. Consider the case when all philosophers simultaneously may pick their left fork. Now, there is no philosopher that has a fork towards his right, and hence no philosopher can finish his meal. This leads to a deadlock due to an infinite waiting stage for all philosophers. **To solve this deadlock**, I have used the condition that ensures at least one of the philosophers should pick up his right fork first, and only then his left fork. This ensures that the circle is broken and hence avoids deadlock situation to ever arise, as atleast one philosopher would be guaranteed to finish his meal and drop the forks later.

Part 2: There is no deadlock possible in this case, where there is just one fork needed and one sauce bowl needed to eat a meal. This is because, the philosophers would always have a fork available at all times (say to their left if everyone only picks their left forks to eat). The sauce bowls would also not cause any deadlock, since philosophers with the sauce bowls are guaranteed to finish their meal and hence put the sauce bowls back, leading to no chance of a deadlock case. Hence, no code has been submitted.

Part 3: In this case, the philosophers need both forks to eat, along with a sauce bowl. The solution to this is very similar to part 1, which avoids deadlocks as well. The sauce bowls are also made as a counting semaphore with count 4. After a philosopher picks up his left and right forks, then he is allowed to pick up the sauce bowl (sem_wait for saucebowl semaphore). And after finishing his meal, he puts back the sauce bowl along with the forks (sem_post for saucebowl semaphore). Since the philosopher picks up the sauce bowl only after he can get both the forks, the 4-sauce bowls would also never run out, since at a time only 2 philosophers can have both forks in his hands (in the worst case). Hence this solution is guaranteed deadlock free, along with the first solution.