Practical No. 6

[Process Synchronization]

Considered there are N philosophers seated around a circular table with one chopstick between each pair of philosophers. There is one chopstick between each philosopher. A philosopher may eat if he can pick up the two chopsticks adjacent to him. One chopstick may be picked up by any one of its adjacent followers but not both.

Write a program to solve the problem using process synchronization technique.

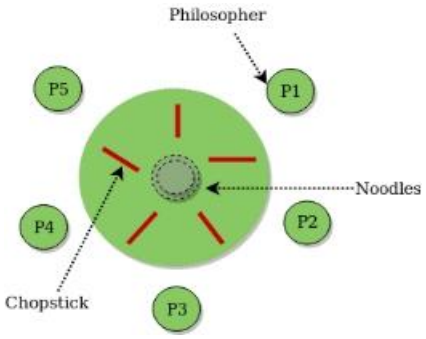

Fig: 1. Dining Philosopher Problem

INPUT :

MINGW64:/c/Users/HP

GNU nano 8.7 dining.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>

#define N 5   // Number of philosophers

pthread_t philosopher[N];
sem_t chopstick[N];
sem_t room;   // To prevent deadlock

void* dine(void* num)
{
    int id = *(int*)num;

    while(1)
    {
        printf("Philosopher %d is thinking\n", id);
        sleep(1);

        // Allow only N-1 philosophers inside room
        sem_wait(&room);

        // Pick left chopstick
        sem_wait(&chopstick[id]);
        printf("Philosopher %d picked LEFT chopstick\n", id);

        // Pick right chopstick
        sem_wait(&chopstick[(id+1)%N]);
        printf("Philosopher %d picked RIGHT chopstick\n", id);

        printf("Philosopher %d is EATING\n", id);
        sleep(2);

        // Put down chopsticks
        sem_post(&chopstick[id]);
        sem_post(&chopstick[(id+1)%N]);

        printf("Philosopher %d finished eating\n", id);

        // Leave room
        sem_post(&room);
    }
}

int main()
{
    int i;
    int phil_num[N];

    // Initialize room semaphore to N-1
    sem_init(&room, 0, N-1);

    // Initialize chopsticks as 1 (available)
    for(i = 0; i < N; i++)
        sem_init(&chopstick[i], 0, 1);

    // Create philosopher threads
    for(i = 0; i < N; i++)
    {
        phil_num[i] = i;
        pthread_create(&philosopher[i], NULL, dine, &phil_num[i]);
    }

    // Join threads
    for(i = 0; i < N; i++)
        pthread_join(philosopher[i], NULL);

    return 0;
}
```

OUTPUT :

```
~

HP@Hiteshri MSYS ~
$ nano dining.c

HP@Hiteshri MSYS ~
$ gcc dining.c -o dining -lpthread

HP@Hiteshri MSYS ~
$ ./dining
Philosopher 0 is thinking
Philosopher 1 is thinking
Philosopher 2 is thinking
Philosopher 3 is thinking
Philosopher 4 is thinking
Philosopher 3 picked LEFT chopstick
Philosopher 4 picked LEFT chopstick
Philosopher 2 picked LEFT chopstick
Philosopher 1 picked LEFT chopstick
Philosopher 4 picked RIGHT chopstick
Philosopher 4 is EATING
Philosopher 4 finished eating
Philosopher 3 picked RIGHT chopstick
Philosopher 4 is thinking
Philosopher 3 is EATING
Philosopher 0 picked LEFT chopstick
Philosopher 3 finished eating
Philosopher 2 picked RIGHT chopstick
Philosopher 3 is thinking
Philosopher 4 picked LEFT chopstick
Philosopher 2 is EATING
Philosopher 2 finished eating
Philosopher 2 is thinking
Philosopher 1 picked RIGHT chopstick
Philosopher 3 picked LEFT chopstick
Philosopher 1 is EATING
Philosopher 1 finished eating
Philosopher 1 is thinking
Philosopher 0 picked RIGHT chopstick
Philosopher 2 picked LEFT chopstick
Philosopher 0 is EATING
Philosopher 4 picked RIGHT chopstick
Philosopher 0 finished eating
Philosopher 4 is EATING
Philosopher 0 is thinking
Philosopher 1 picked LEFT chopstick
Philosopher 4 finished eating
Philosopher 3 picked RIGHT chopstick
Philosopher 4 is thinking
Philosopher 0 picked LEFT chopstick
Philosopher 3 is EATING
Philosopher 3 finished eating
Philosopher 3 is thinking
```