

[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.

Code:-

```
GNU nano 8.7
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>

#define N 5

sem_t chopstick[N];

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

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

    if (id == N - 1) {
        sem_wait(&chopstick[(id + 1) % N]);
        sem_wait(&chopstick[id]);
    } else {
        sem_wait(&chopstick[id]);
        sem_wait(&chopstick[(id + 1) % N]);
    }

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

    sem_post(&chopstick[id]);
    sem_post(&chopstick[(id + 1) % N]);

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

int main() {
    pthread_t phil[N];
    int ids[N];

    for (int i = 0; i < N; i++)
        sem_init(&chopstick[i], 0, 1);

    for (int i = 0; i < N; i++) {
        ids[i] = i;
        pthread_create(&phil[i], NULL, philosopher, &ids[i]);
    }

    for (int i = 0; i < N; i++)
        pthread_join(phil[i], NULL);

    return 0;
}
```

⌘G Help ⌘O Write Out ⌘F Where Is ⌘K Cut ⌘T Execute
⌘X Exit ⌘R Read File ⌘\ Replace ⌘U Paste ⌘J Justify

Output:-

```
M ~  
ladde@Hashirama MSYS ~  
$ nano practical6.c  
  
ladde@Hashirama MSYS ~  
$ gcc practical6.c -o practical6  
  
ladde@Hashirama MSYS ~  
$ ./practical6  
Philosopher 0 is thinking  
Philosopher 1 is thinking  
Philosopher 2 is thinking  
Philosopher 3 is thinking  
Philosopher 4 is thinking  
Philosopher 0 is eating  
Philosopher 2 is eating  
Philosopher 0 finished eating  
Philosopher 4 is eating  
Philosopher 1 is eating  
Philosopher 2 finished eating  
Philosopher 3 is eating  
Philosopher 4 finished eating  
Philosopher 1 finished eating  
Philosopher 3 finished eating
```