6. [Process Synchronization]

Considered there are N philosophers seated around a circular table CO3 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

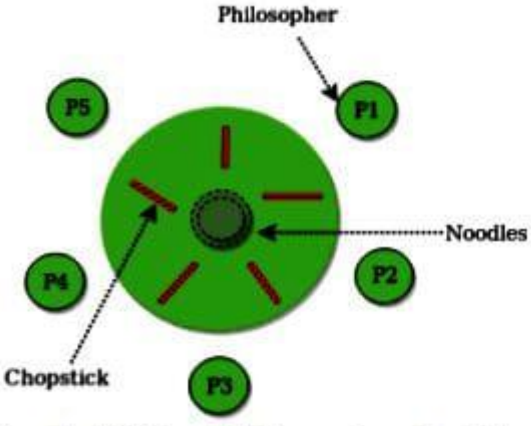
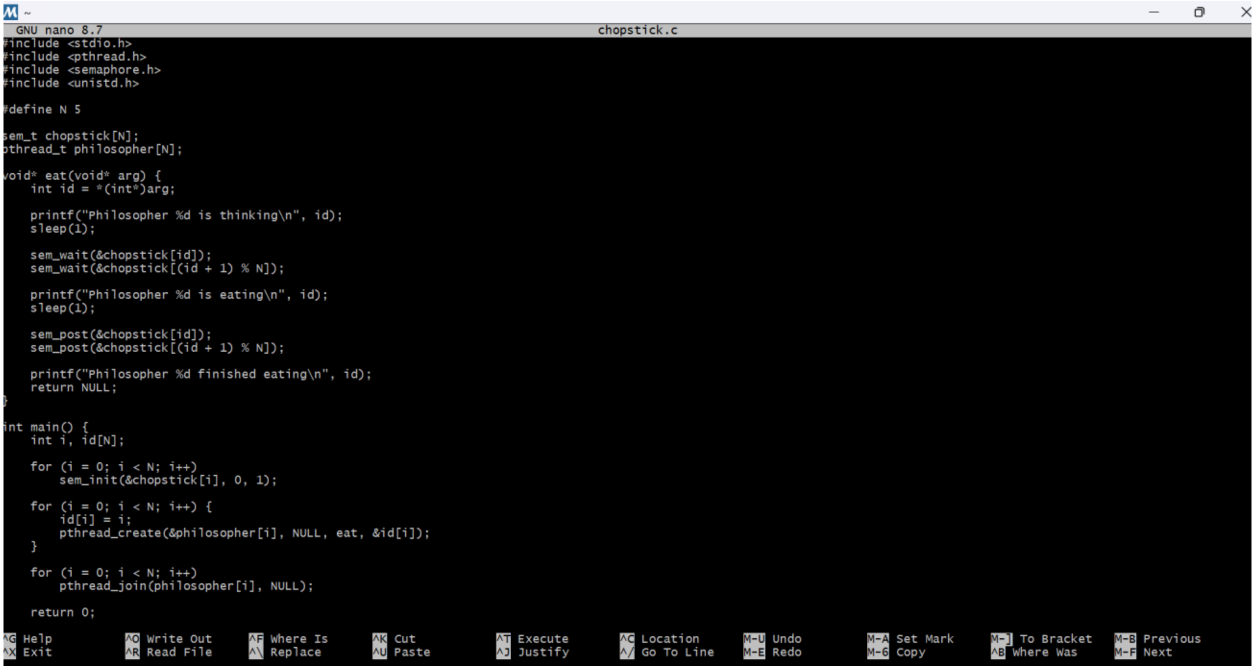
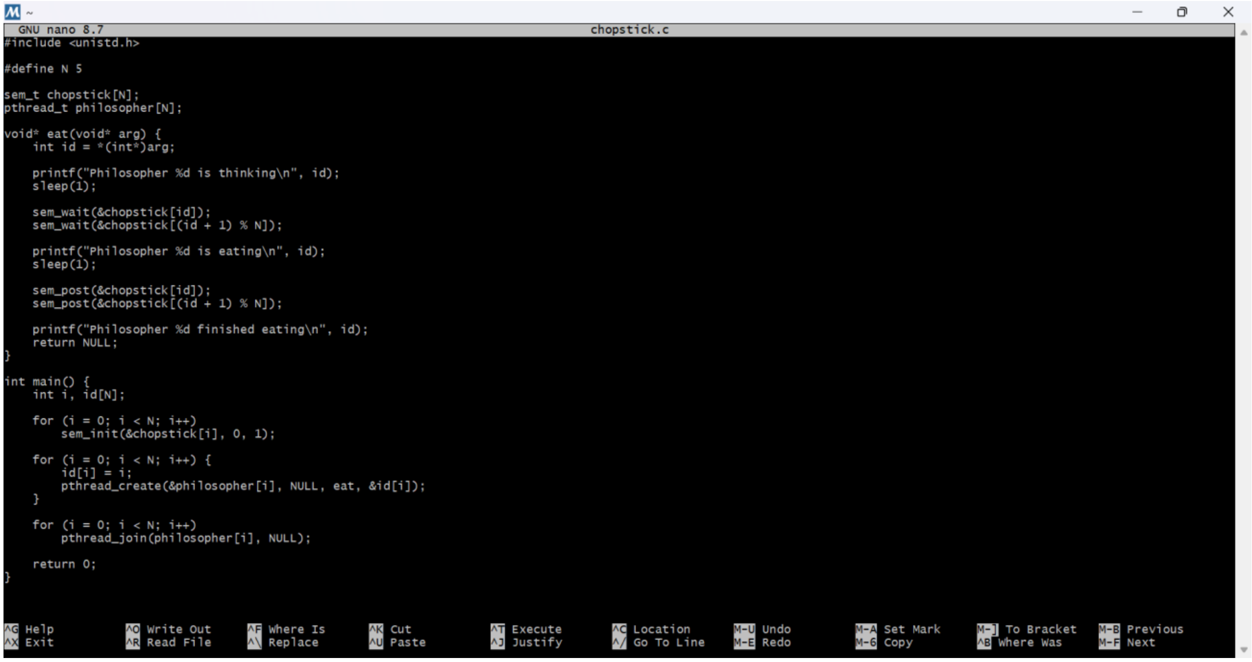


Fig: 1. Dining Philosopher Problem

CODE:

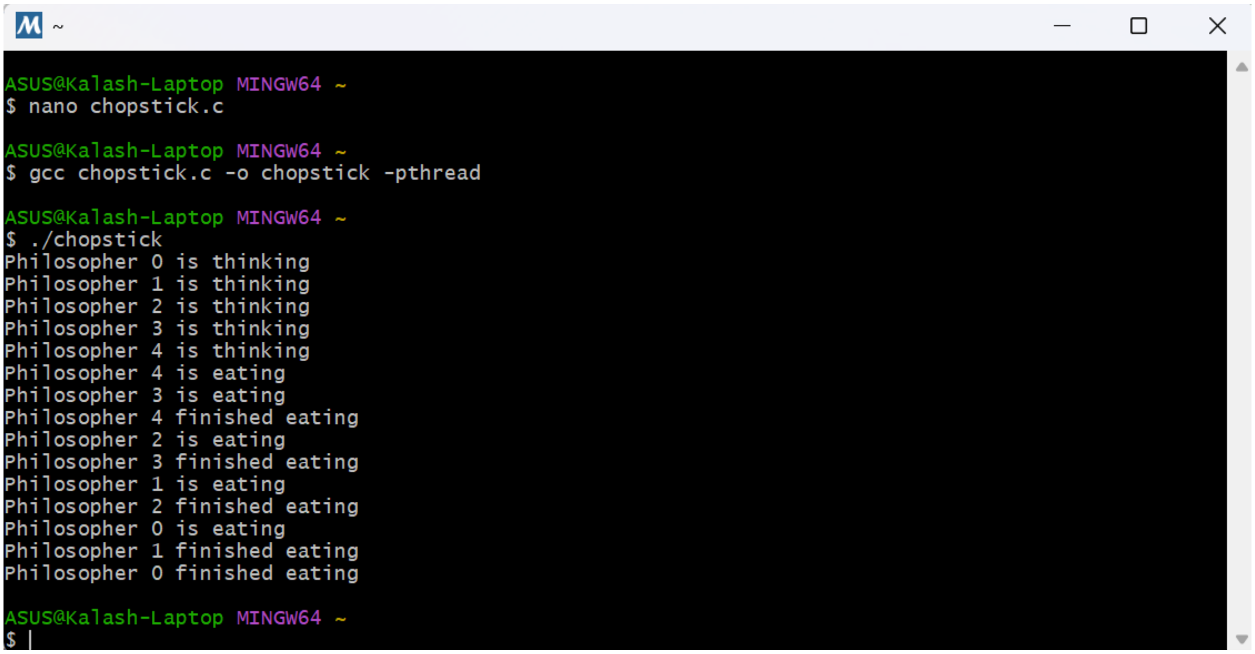


```
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>

#define N 5

sem_t chopstick[N];
pthread_t philosopher[N];

void* eat(void* arg) {
    int id = *(int*)arg;

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

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

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

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

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

int main() {
    int i, id[N];

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

    for (i = 0; i < N; i++) {
        id[i] = i;
        pthread_create(&philosopher[i], NULL, eat, &id[i]);
    }

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

    return 0;
```



```c
#include <unistd.h>

#define N 5

sem_t chopstick[N];
pthread_t philosopher[N];

void* eat(void* arg) {
    int id = *(int*)arg;

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

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

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

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

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

int main() {
    int i, id[N];

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

    for (i = 0; i < N; i++) {
        id[i] = i;
        pthread_create(&philosopher[i], NULL, eat, &id[i]);
    }

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

    return 0;
}
```

OUTPUT :



```
ASUS@Kalash-Laptop MINGW64 ~
$ nano chopstick.c

ASUS@Kalash-Laptop MINGW64 ~
$ gcc chopstick.c -o chopstick -pthread

ASUS@Kalash-Laptop MINGW64 ~
$ ./chopstick
Philosopher 0 is thinking
Philosopher 1 is thinking
Philosopher 2 is thinking
Philosopher 3 is thinking
Philosopher 4 is thinking
Philosopher 4 is eating
Philosopher 3 is eating
Philosopher 4 finished eating
Philosopher 2 is eating
Philosopher 3 finished eating
Philosopher 1 is eating
Philosopher 2 finished eating
Philosopher 0 is eating
Philosopher 1 finished eating
Philosopher 0 finished eating

ASUS@Kalash-Laptop MINGW64 ~
$ |
```