

## **EXP 12: Design a C program to simulate the concept of Dining-Philosophers problem**

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

#define NUM_PHILOSOPHERS 5

pthread_mutex_t forks[NUM_PHILOSOPHERS];

void* philosopher(void* num) {
    int id = *(int*)num;
    int left = id;
    int right = (id + 1) % NUM_PHILOSOPHERS;

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

    pthread_mutex_lock(&forks[left]);
    printf("Philosopher %d picked up left fork %d.\n", id, left);

    pthread_mutex_lock(&forks[right]);
    printf("Philosopher %d picked up right fork %d.\n", id, right);

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

    pthread_mutex_unlock(&forks[right]);
    pthread_mutex_unlock(&forks[left]);
}
```

```

printf("Philosopher %d put down forks and finished eating.\n", id);

return NULL;
}

int main() {
    pthread_t philosophers[NUM_PHILOSOPHERS];
    int ids[NUM_PHILOSOPHERS];
    int i;

    for (i = 0; i < NUM_PHILOSOPHERS; i++) {
        pthread_mutex_init(&forks[i], NULL);
    }

    for (i = 0; i < NUM_PHILOSOPHERS; i++) {
        ids[i] = i;
        pthread_create(&philosophers[i], NULL, philosopher, &ids[i]);
    }

    for (i = 0; i < NUM_PHILOSOPHERS; i++) {
        pthread_join(philosophers[i], NULL);
    }

    for (i = 0; i < NUM_PHILOSOPHERS; i++) {
        pthread_mutex_destroy(&forks[i]);
    }

    return 0;
}

```

## Sample Output

```
Philosopher 2 is thinking.  
Philosopher 0 is thinking.  
Philosopher 3 is thinking.  
Philosopher 1 is thinking.  
Philosopher 4 is thinking.  
Philosopher 4 picked up left fork 4.  
Philosopher 0 picked up left fork 0.  
Philosopher 1 picked up left fork 1.  
Philosopher 2 picked up left fork 2.  
Philosopher 3 picked up left fork 3.  
|
```