Ganpat University | Institute of Computer Technology
॥ विद्यया समाजोत्कर्षः ॥

Name: Tushar Panchal

En.No: 21162101014

Sub: OS(Operating Systems)

Branch: CBA

Batch:41

----------PRACTICAL 7----------

❖ *Experiment-7 :*

To demonstrate thread synchronization using Semaphores and Mutex.

Q1.Job Master Problem:

✓ Source Code :

```c
// JOB MASTER PROBLEM
#include <stdio.h>
#include <pthread.h>
int x = 0;
pthread_mutex_t mutex;
void *routine(void *arg)
{
    pthread_mutex_lock(&mutex);
    x++;

    printf("Job %d has Started \n", x);
    printf("//////////////////////////\n");
    printf("Job %d has finished \n", x);
    printf("========================\n");
    pthread_mutex_unlock(&mutex);
}

int main()
{
    printf("\n JOB MASTER PROBLEM \n");
    printf("========================\n");
    pthread_t t1, t2;
    pthread_mutex_init(&mutex, NULL);
    pthread_create(&t1, NULL, routine, NULL);
    pthread_create(&t2, NULL, routine, NULL);
    pthread_join(t1, NULL);
```

```c
    pthread_join(t2, NULL);
    return 0;
}
```

✓ **Output :**

```
╭─tushar@tushar in ~/Documents/OS/7 via C v12.2.1-gcc took 1ms
╰─λ ./1
JOB MASTER PROBLEM
========================
Job 1 has Started
////////////////////////
Job 1 has finished
========================
Job 2 has Started
////////////////////////
Job 2 has finished
========================
```

Q2.Dining Philosopher Problem:

[A] Using Semaphore

✓ **Source Code :**

```c
/* Q2: Dining Philosophers Problem
[A] Using Semaphores */
#include<stdio.h>
#include<stdlib.h>
#include<pthread.h>
#include<semaphore.h>

#define N 4
sem_t forks[N];
sem_t mutex;

void *routine(void *arg)
{
    int id=*(int *)arg;
    int left=id %N;
    int right=(id+1)%N;
    printf("Philosopher %d is thinking\n",id);
    sem_wait(&mutex);
    sem_wait(&forks[left]);  // L fork Picked
    sem_wait(&forks[right]);  // R fork Picked
    sem_post(&mutex);
    printf("Philosopher %d is eating\n",id);
    sem_post(&forks[right]); // R fork down
    sem_post(&forks[left]); // L fork down
    printf("========================\n");
    return NULL;
}

int main()
{
    printf("⋘⋘⋘ Using Semaphores ⋙⋙⋙\n");
    printf("////////////////////////\n");
```

```
    pthread_t p[N];
    int pid[N]={0,1,2,3};
    sem_init(&mutex,0,1);
    for(int i=0;i<N;i++)
    {
        sem_init(&forks[i],0,1);
    }
    for(int i=0;i<N;i++)
    {
        pthread_create(&p[i],NULL,routine,&pid[i]);
    }
    for(int i=0;i<N;i++)
    {
        pthread_join(p[i],NULL);
    }
    return 0;
}
```

✓ **Output :**

```
┌──tushar@tushar in ~/Documents/OS/7 via C v12.2.1-gcc took 3ms
└─λ ./2_first
<<<<<<< Using Semaphores >>>>>>>
//////////////////////////
Philosopher 0 is thinking
Philosopher 0 is eating
==========================
Philosopher 1 is thinking
Philosopher 1 is eating
==========================
Philosopher 2 is thinking
Philosopher 2 is eating
==========================
Philosopher 3 is thinking
Philosopher 3 is eating
==========================
```

[B] Using Mutex

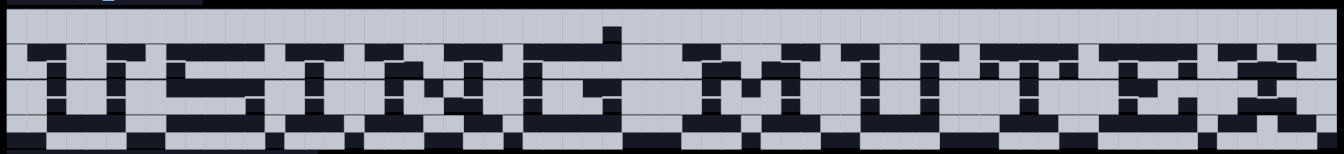
✓ **Source Code :**

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>
#define NUM_PHILOSOPHERS 5
pthread_mutex_t chopsticks[NUM_PHILOSOPHERS];
void *philosopher(void *arg)
{
    int id = *(int *)arg;
    int left_chopstick = id;
    int right_chopstick = (id + 1) % NUM_PHILOSOPHERS;
    if (id == 4)
    {
```

```
        int temp = left_chopstick;
        left_chopstick = right_chopstick;
        right_chopstick = temp;
    }
    printf("Philosopher %d is 🤔thinking.\n", id);
    pthread_mutex_lock(&chopsticks[left_chopstick]);
    printf("%cPhilosopher %d picked up left 🥢chopstick.\n", id);
    pthread_mutex_lock(&chopsticks[right_chopstick]);
    printf("%cPhilosopher %d picked up right 🥢chopstick.\n", id);
    printf("Philosopher %d is 😋eating.\n", id);
    sleep(2);
    pthread_mutex_unlock(&chopsticks[left_chopstick]);
    printf("%cPhilosopher %d put down left 🥢chopstick.\n", id);
    pthread_mutex_unlock(&chopsticks[right_chopstick]);
    printf("%cPhilosopher %d put down right 🥢chopstick.\n", id);
    printf("Philosopher %d Finished 😋eating.\n", id);
    return NULL;
}
int main()
{
    printf("                                                                                \n");
    printf("                                                                                \n");
    printf("                                                                                \n");
    printf("                                                                                \n");
    pthread_t philosophers[NUM_PHILOSOPHERS];
    int ids[NUM_PHILOSOPHERS];
    for (int i = 0; i < NUM_PHILOSOPHERS; i++)
    {
        ids[i] = i;
        pthread_mutex_init(&chopsticks[i], NULL);
    }
    for (int i = 0; i < NUM_PHILOSOPHERS; i++)
    {
        pthread_create(&philosophers[i], NULL, philosopher, &ids[i]);
    }
    for (int i = 0; i < NUM_PHILOSOPHERS; i++)
    {
        pthread_join(philosophers[i], NULL);
    }
    for (int i = 0; i < NUM_PHILOSOPHERS; i++)
    {
        pthread_mutex_destroy(&chopsticks[i]);
    }
    return 0;
}
```

✓ **Output :**

```
┌──tushar@tushar in ~/Documents/OS/7 via C v12.2.1-gcc took 2ms
└─λ ./2_second
```



```
Philosopher 0 is thinking.
Philosopher 1 picked up left chopstick.
Philosopher 1 picked up right chopstick.
Philosopher 0 is  ·eating.
Philosopher 1 is thinking.
Philosopher 2 is thinking.
```

```
Philosopher 1 picked up left chopstick.
Philosopher 1 picked up right chopstick.
Philosopher 2 is eating.
Philosopher 3 is thinking.
Philosopher 4 is thinking.
Philosopher 0 put down left chopstick.
Philosopher 1 put down right chopstick.
Philosopher 2 Finished  ·eating.
Philosopher 2 picked up left chopstick.
Philosopher 1 picked up right chopstick.
Philosopher 3 is  ·eating.
Philosopher 2 picked up left chopstick.
Philosopher 1 put down left chopstick.
Philosopher 1 put down right chopstick.
Philosopher 0 Finished  ·eating.
Philosopher 2 picked up left chopstick.
Philosopher 1 picked up right chopstick.
Philosopher 1 is  ·eating.
Philosopher 1 put down left chopstick.
Philosopher 1 put down right chopstick.
Philosopher 3 Finished  ·eating.
Philosopher 2 picked up right chopstick.
Philosopher 4 is  ·eating.
Philosopher 1 put down left chopstick.
Philosopher 0 put down right chopstick.
Philosopher 1 Finished  ·eating.
Philosopher 1 put down left chopstick.
Philosopher 1 put down right chopstick.
Philosopher 4 Finished  ·eating.
```