**Assignment 8**

Operating Systems

Submitted by:

Logo, company name

Description automatically generatedGodala Sukumar Reddy(2020BTechCSE032)

Institute of Engineering and Technology (IET)

JK Lakshmipat University, Jaipur

November 2022

**Implement Dining Philosopher’s’ Problem in Java and C language:**

**Java Programming:**

**public class DiningPhilosophers {**

**public static void main(String[] args) throws Exception {**

**Philosopher[] philosophers = new Philosopher[5];**

**Object[] forks = new Object[philosophers.length];**

**for (int i = 0; i < forks.length; i++) {**

**forks[i] = new Object();**

**}**

**for (int i = 0; i < philosophers.length; i++) {**

**Object leftFork = forks[i];**

**Object rightFork = forks[(i + 1) % forks.length];**

**if (i == philosophers.length - 1) {**

**philosophers[i] = new Philosopher(rightFork, leftFork); // The last philosopher picks up the right fork first**

**} else {**

**philosophers[i] = new Philosopher(leftFork, rightFork);**

**}**

**Thread t = new Thread(philosophers[i], "Philosopher " + (i + 1));**

**t.start();**

**}**

**}**

**}**

**C Programming:**

#include <pthread.h>

#include <semaphore.h>

#include <stdio.h>

#define N 5

#define THINKING 2

#define HUNGRY 1

#define EATING 0

#define LEFT (phnum + 4) % N

#define RIGHT (phnum + 1) % N

int state[N];

int phil[N] = { 0, 1, 2, 3, 4 };

sem\_t mutex;

sem\_t S[N];

void test(int phnum)

{

if (state[phnum] == HUNGRY

&& state[LEFT] != EATING

&& state[RIGHT] != EATING) {

state[phnum] = EATING;

sleep(2);

printf("Philosopher %d takes fork %d and %d\n", phnum + 1, LEFT + 1, phnum + 1);

printf("Philosopher %d is Eating\n", phnum + 1);

sem\_post(&S[phnum]);

}

}

void take\_fork(int phnum)

{

sem\_wait(&mutex);

state[phnum] = HUNGRY;

printf("Philosopher %d is Hungry\n", phnum + 1);

test(phnum);

sem\_post(&mutex);

sem\_wait(&S[phnum]);

sleep(1);

}

void put\_fork(int phnum)

{

sem\_wait(&mutex);

state[phnum] = THINKING;

printf("Philosopher %d putting fork %d and %d down\n",

phnum + 1, LEFT + 1, phnum + 1);

printf("Philosopher %d is thinking\n", phnum + 1);

test(LEFT);

test(RIGHT);

sem\_post(&mutex);

}

void\* philospher(void\* num)

{

while (1) {

int\* i = num;

sleep(1);

take\_fork(\*i);

sleep(0);

put\_fork(\*i);

}

}

int main()

{

int i;

pthread\_t thread\_id[N];

sem\_init(&mutex, 0, 1);

for (i = 0; i < N; i++)

sem\_init(&S[i], 0, 0);

for (i = 0; i < N; i++) {

pthread\_create(&thread\_id[i], NULL,

philospher, &phil[i]);

printf("Philosopher %d is thinking\n", i + 1);

}

for (i = 0; i < N; i++)

pthread\_join(thread\_id[i], NULL);

}