**DINING PHILOSOPHER PROBLEM**

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <semaphore.h>

#include <unistd.h>

#define N 5 // Number of philosophers

sem\_t forks[N]; // One semaphore per fork

sem\_t mutex; // For printing output without interleaving

void\* philosopher(void\* num) {

int id = \*(int\*)num;

while (1) {

// Thinking

sem\_wait(&mutex);

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

sem\_post(&mutex);

sleep(1);

// Pick up left fork

sem\_wait(&forks[id]);

// Pick up right fork

sem\_wait(&forks[(id + 1) % N]);

// Eating

sem\_wait(&mutex);

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

sem\_post(&mutex);

sleep(2);

// Put down right fork

sem\_post(&forks[(id + 1) % N]);

// Put down left fork

sem\_post(&forks[id]);

}

return NULL;

}

int main() {

pthread\_t thread\_id[N];

int philosopher\_ids[N];

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++) {

philosopher\_ids[i] = i;

pthread\_create(&thread\_id[i], NULL, philosopher, &philosopher\_ids[i]);

}

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

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

return 0;

}