Lab 4 Report

Name: Insiyah Ujjainwala
Email: ujjainwi@mcmaster.ca
Student ID: 400357483
Date: 19/11/2023

Dining Philsopher's Problem

- The dining-philosopher problem is a synchronisation challenge involving multiple philosophers sharing a table with limited resources (forks) to alternate between thinking and eating. In this scenario.
- The goal is to implement a solution using POSIX mutex locks and condition variables.
- Five philosophers, each represented by a number between 0 and 4, will operate as separate threads.
- They will oscillate between thinking and eating, simulating these actions by sleeping for random durations between one and three seconds.
- To manage the access to forks, the philosophers will use the functions *pickup_forks()* and *return_forks()*, which will require the implementation of POSIX mutex locks and condition variables to coordinate their actions effectively.

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

#define NUMPHILOSOPHERS 5

int forks[NUMPHILOSOPHERS] = {1, 1, 1, 1, 1};
pthread_mutex_t lock[NUMPHILOSOPHERS] = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t avail_fork[NUMPHILOSOPHERS];

// Philsopher tries to pick up forks
void pickup_forks(int philosopher_num) {
pthread_mutex_lock(&lock[philosopher_num]);

// Check ing for left fork
while (forks[philosopher_num] <= 0) {
pthread_cond_wait(&avail_fork[philosopher_num], &lock[philosopher_num]);
}

forks[philosopher_num]--;

// Checking for right fork
int f_right = (philosopher_num + 1) % NUMPHILOSOPHERS;
while (forks[f_right] <= 0) {
pthread_cond_wait(&avail_fork[f_right], &lock[f_right]);
}
```

```
forks[f_right]--;

pthread_mutex_unlock(&lock[philosopher_num]);
}

// Function to return forks by a philosopher
void return_forks(int philosopher_num) {
pthread_mutex_lock(&lock[philosopher_num]);

forks[philosopher_num]++;
pthread_cond_signal(&avail_fork[philosopher_num]);

int right_fork = (philosopher_num + 1) % NUMPHILOSOPHERS;
forks[right_fork]++;
pthread_cond_signal(&avail_fork[right_fork]);

pthread_mutex_unlock(&lock[philosopher_num]);
}

void *Phil(void *num) {
int philosopher_num = *((int *)num);
srand(time(NULL) + philosopher_num);

while (1) {

// Philsopher's thinking time
int thinkTime1 = rand() % 3 + 1;
// Thinking phase
printf("Philosopher %d is thinking for %d seconds.\n", philosopher_num,
thinkTime1);
// Philsopher sleeps for random period
sleep(thinkTime1);


// Pick up forks to start eating phase
pickup_forks(philosopher_num);
// Philsopher's thinking time
int thinkTime2 = rand() % 3 + 1;
// Eating phase
printf("Philosopher %d is eating for %d seconds.\n", philosopher_num, thinkTime2);
// Philsopher sleeps for a random period
sleep(thinkTime2);

// Philsopher returns forks after eating
return_forks(philosopher_num);
```

```
}
}

int main() {
pthread_t phil[NUMPHILOSOPHERS];
int i = 0;
int args[NUMPHILOSOPHERS];

for (i = 0; i < NUMPHILOSOPHERS; i++) {
args[i] = i;
pthread_cond_init(&avail_fork[i], NULL);
pthread_create(&phil[i], NULL, Phil, (void *)&args[i]);
}

// Wait for all philosopher threads to finish
for (i = 0; i < NUMPHILOSOPHERS; i++) {
pthread_join(phil[i], NULL);
}

return 0;
}
```

Output

```
insiyah@cs3sh3:~$ cd Desktop
insiyah@cs3sh3:~/Desktop$ gcc -o diningphil diningphil.c
insiyah@cs3sh3:~/Desktop$ ./diningphil
Philosopher 0 is thinking for 3 seconds.
Philosopher 1 is thinking for 3 seconds.
Philosopher 2 is thinking for 1 seconds.
Philosopher 3 is thinking for 3 seconds.
Philosopher 4 is thinking for 2 seconds.
Philosopher 2 is eating for 3 seconds.
Philosopher 4 is eating for 2 seconds.
Philosopher 2 is thinking for 3 seconds.
Philosopher 1 is eating for 2 seconds.
Philosopher 3 is eating for 1 seconds.
Philosopher 4 is thinking for 2 seconds.
Philosopher 3 is thinking for 3 seconds.
Philosopher 1 is thinking for 3 seconds.
Philosopher 0 is eating for 1 seconds.
Philosopher 0 is thinking for 2 seconds.
Philosopher 4 is eating for 1 seconds.
Philosopher 4 is thinking for 2 seconds.
Philosopher 3 is eating for 3 seconds.
Philosopher 3 is thinking for 1 seconds.
Philosopher 4 is eating for 2 seconds.
Philosopher 4 is thinking for 1 seconds.
Philosopher 3 is eating for 1 seconds.
Philosopher 3 is thinking for 3 seconds.
Philosopher 3 is eating for 1 seconds.
Philosopher 3 is thinking for 1 seconds.
Philosopher 3 is eating for 3 seconds.
Philosopher 3 is thinking for 2 seconds.
Philosopher 3 is eating for 2 seconds.
```