**Assignment-3**

Name: Aastha Jain

Roll No: 39

PRN: 12211635

Class: CS-B

**Dining Philosopher Problem**

**Code:**

#!/bin/bash

# Number of philosophers

NUM\_PHILOSOPHERS=5

# Array to store the forks, initialized as available (1)

forks=()

# Initialize forks as available

for ((i=0; i<$NUM\_PHILOSOPHERS; i++)); do

forks[$i]=1

done

# Function to simulate a philosopher's behavior

philosopher() {

local id=$1

local left\_fork=$id

local right\_fork=$(( ($id + 1) % $NUM\_PHILOSOPHERS ))

echo "Philosopher $id is thinking."

while true; do

echo "Philosopher $id is hungry."

# Try to pick up forks

if [[ ${forks[$left\_fork]} -eq 1 && ${forks[$right\_fork]} -eq 1 ]]; then

forks[$left\_fork]=0

forks[$right\_fork]=0

echo "Philosopher $id picked up forks $left\_fork and $right\_fork and is eating."

sleep 2 # Simulate eating

echo "Philosopher $id finished eating and puts down forks $left\_fork and $right\_fork."

forks[$left\_fork]=1

forks[$right\_fork]=1

else

echo "Philosopher $id couldn't pick up forks, will retry later."

fi

echo "Philosopher $id is thinking."

sleep 1 # Simulate thinking

done

}

# Create processes for each philosopher

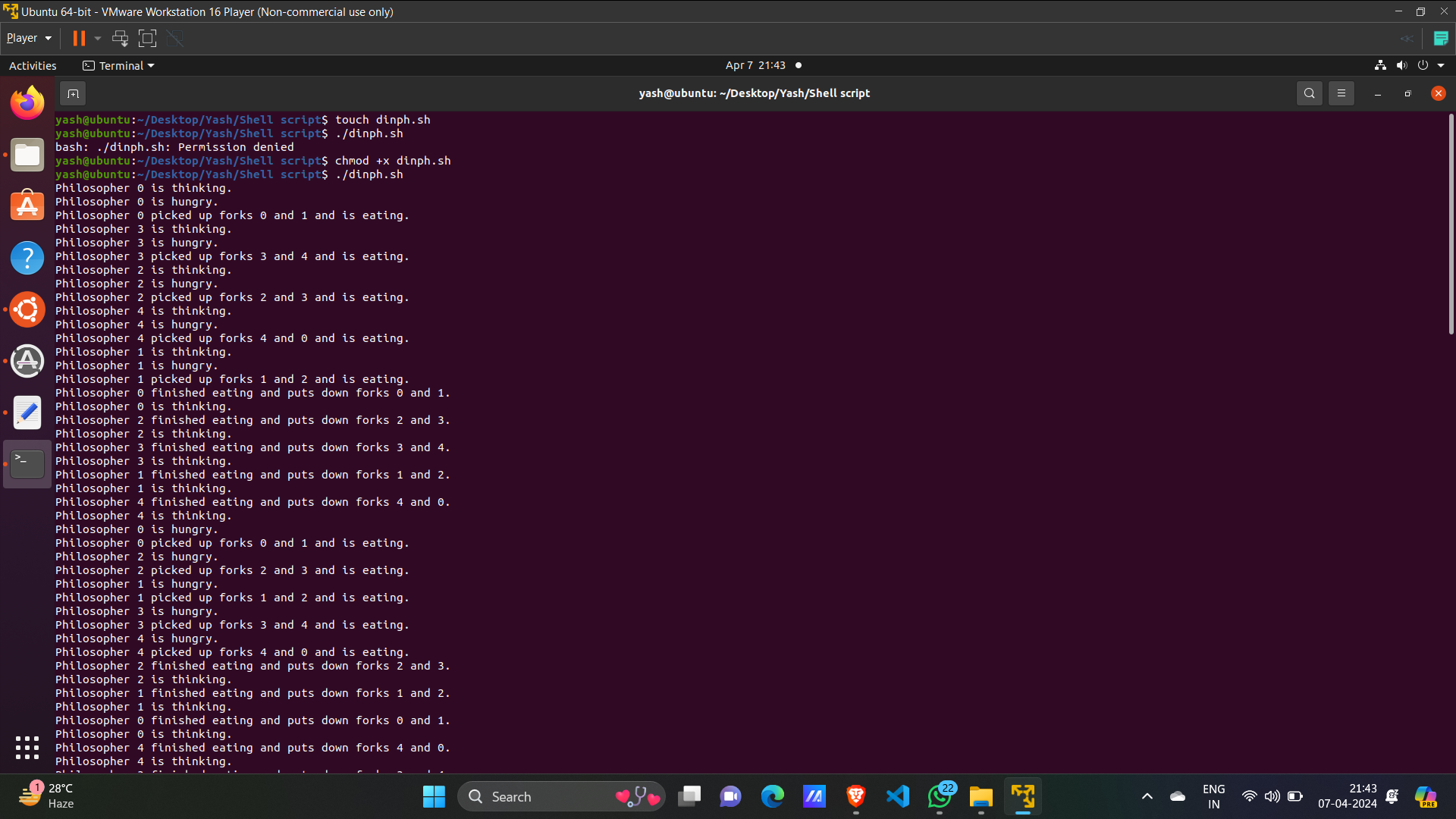
for ((i=0; i<$NUM\_PHILOSOPHERS; i++)); do

philosopher $i &

done

# Wait for all philosopher processes to finish

wait



#include <pthread.h>

#include <semaphore.h>

#include <stdio.h>

#include <unistd.h>

#define N 5a

#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 that 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\* philosopher(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,

philosopher, &phil[i]);

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

}

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

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

}