Muhammed Evgin 110510163

Dining Philosopher Problem

This project is designed to solve the dining philosopher's problem, which is a classic problem in concurrent programming that illustrates the challenges of managing shared resources. The problem describes a scenario in which a group of philosophers are sitting around a table, each with their own plate of food and a pair of chopsticks. The philosophers alternate between thinking and eating, and in order to eat, they must first pick up both chopsticks on their left and right side.

To solve the problem, the program uses Pthreads, mutex, and condition variables to synchronize the philosophers' access to the chopsticks. The program takes in command line arguments such as the number of philosophers, the minimum and maximum thinking and dining times, and the distribution type (uniform or exponential).

The program creates a Philosopher struct for each philosopher, which includes the philosopher's id, thinking and dining times, and a pointer to the chopsticks and self-condition variable. The program also includes a generateRandomTime function which generates a random time for the thinking and dining times based on the provided distribution type.

The philosopherThread function is run in each philosopher thread, and it implements the logic of the philosopher thinking and dining. The philosophers generate their thinking time, then think and sleeps for the generated time. Then philosopher try to acquire the chopsticks by locking them using mutex. After generating the dining time, philosopher dine and sleeps for the generated time. Then release the chopsticks by unlocking them using mutex.

The program also uses a deadlock-free solution such as using semaphores or monitors to ensure that the philosophers do not get stuck in a deadlock state, in which they are unable to acquire the resources they need to continue. The program terminates after all of the philosophers have finished dining.

The random values were generated in the function generateRandomTime(). The function takes in 3 parameters: min, max, and dst. min and max are the lower and upper bounds respectively for the random value. dst specifies the distribution of the random values. The function uses the rand() function to generate random values.

If dst is "uniform", the function generates a random value between min and max using the following formula: time = (rand() % (max - min + 1)) + min.

If dst is "exponential", the function generates a random value with an exponential distribution using the following steps:

Calculate the lambda value using the formula: lambda = 1.0 / ((min + max) / 2).

Generate a random value rand_val between 0 and 1 using the formula: rand_val = ((double) rand() / (double) RAND_MAX).

If rand_val is 0, repeat step 2.

Calculate the time value using the formula: time = -log(rand_val) / lambda.

The function then returns the generated time value.

I measured the duration of hungry state for each philosopher, find out the average hungry state and the standard deviation of hungry state. These are the result of ./ phsp 25 500 1000 50 100 exponential 500.

Philosophers total_thinking_time is 28852 ms
Philosophers average_thinking_time is 1154 ms
Philosophers standard deviation is 108 ms