Operating Systems
2019 - 2020 Fall
Project 2

In this project, I write a program for the dining philosopher problem, which will implement a deadlock-free solution with maximum concurrency. The program works for any number of philosophers (odd numbers, maximum 27). In the program, a thread will express a philosopher. These threads will be spawned by the main thread, which is not a philosopher. I used Pthreads, mutex and condition variables to synchronize. Also, I used semaphores that are deadlock free.

The input given:

phsp 5 500 1000 50 100 exponential 100

Sample Output:

Philosopher waited for 31.470039ms

Philosopher 3 is finish his job.

Philosopher waited for 41.583874ms

Philosopher 0 is finish his job.

Philosopher waited for 123.048535ms

Philosopher 1 is finish his job.

Philosopher waited for 137.891421ms

Philosopher 4 is finish his job.

Philosopher waited for 175.430389ms

Philosopher 2 is finish his job.

In the output, I measured the duration of hungry state for each philosopher. My codes are in the phsp.c file.