A blue and white logo

Description automatically generated

A Study of the Synchronisation and Concurrency Issues in the Dining Philosophers’ Problem completed using the ThreadMentor Visualisation Tool

**Submitted by: Habiba Nour, B00151078**

**Steven Kelly, B00150588**

**Rochelle Mullen, B00156311**

**Piotr Momat, B00156112**

**Submission date:** *18/MARCH/2024*

Declaration Page:

*We am aware of the University policy on plagiarism in assignments and examinations (3AS08)*

*We understand that plagiarism, collusion, and copying are grave and serious offences in the Institute and We will accept the penalties that could be imposed if we engage in any such activity.*

*This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.*

*We declare that this material, which We now submit for assessment, is entirely of my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work.*

Author: Habiba Nour Dated: 18/MARCH/2024

Author: Steven Kelly Dated: 18/MARCH/2024

Author: Rochelle Mullen Dated: 18/MARCH/2024

Author: Piotr Momat Dated: 18/MARCH/2024

Abstract Page

<NEW PAGE>

Table of Contents

<NEW PAGE>

List of Figures

<NEW PAGE>

List of Tables (if you have them)

<NEW PAGE>

Note: Please make sure that you use proper citation within the body of the text of your report + include the reference cited in the References section at the end. It is **not** sufficient simply to list the references at the end. See:

<https://libguides.ucd.ie/harvardstyle/introduction>

# Introduction (Steven)

## Background

* Concurrency and Synchronisation Issues in OSes (Steven)
* Mutex Locks (Rochelle

According to Threadmentor ebook(2021), Mutex locks which is the short for mutually exclusive locks is one solution to the dining philosopher problem. These locks work as a pair in that if one is locked it cannot unlock until the other has locked too and are seeking to unlock. If only one lock is acquired the philosopher has to wait to acquire the second lock and cannot unlock unless it has both. With the locks the philosopher lock the chopstick that they pick up and will then unlock the chopsticks. Each philosopher will pick up their right chopstick and if they can will pick up the left chopstick. When they finish and put down their chopsticks other sitting nearby that are waiting to acquire chopstick will be able to lock those chopsticks. One way to ensure there is no deadlock with this method is to make one of the philosophers to firstly pick up the chopstick that is on their left while the rest will pick up the chopstick on the right. You can also make it so that all the philosopher pick up their left chopstick bar one. The issue with mutex locks is that it can lead to starvation as those waiting on the chopstick may be waiting for some time especially if the philosopher that picks up the chopstick that is opposite to the rest eats and thinks quickly. As they are quicker it means they will have more access to resources than the others as they will have to wait to acquire their second chopstick. Mutex locks work using a lock method and an unlock method. The handedness of the philosopher can be assigned too within the code.

* Semaphores (Habiba)
* ThreadMentor (Piotr)

## The Dining Philosophers Problem (Habiba)

## Outline/Layout of your Report

* In section 2, we describe “this”. In section 3, we describe “that”…
* Tie the sections together: *briefly* describe how they are related

# <Your Solution>

* Theory/How it works (Rochelle)
* According to the Threadmentor ebook the four chair solutions is a deadlock free solution. It works by only allowing 4 philosophers to sit at the table at one time. It does this using a private queue and counter as well as having a signal method. When a philosopher sits down to eat they must wait on the semaphore before picking up chopsticks and must signal to it when they are finished eating to release the chair for another to use.

Within the code the constructor takes in number and iter which are number assigned to the tread and the number of cycles respectively. Each chopstick is declared individually and stored in an array. There is also a semaphore called Fourchair which has a value of 4. This semaphore is what is used for the signal and waiting which are needed for the locking to occur. The semaphore and the locks are also static so they can only be used within the files it is declared.

As the locks are declared in philosopher file the main method does not have to initials the locks and contains the creation of the philosophers and number of cycles.

* ThreadMentor: (Piotr)
  + Makefile(Piotr)
  + Explanation of “Tags” and other ThreadMentor issues related to your solution (Piotr)

# Results and Analysis (Piotr & Steven)

Screenshots of a **single** program run, to include:

* Several History Graph screenshots + proper captions
* Screenshots of highlighted code corresponding to various ThreadMentor tags for Philosopher threads in the History graph above + proper captions
* Screenshots of main ThreadMentor window showing relevant and corresponding information relating to the above.
* Screenshots of Thread Status window showing relevant and corresponding information relating to the above.
* Detailed descriptions in the text of what is happening to each of the Philosopher threads making reference to the screenshots.

# Conclusions (Habiba & Rochelle)

Technical conclusions about your solution; for example:

* “our solution avoids a specified problem”, or
* “our does had a specified problem”
* Advantages/Disadvantages of our solution

The conclusions should **not**include things like: “I loved this project!”, “I hated this project!”, “I learned x, y and z on this project”. These things should go in the Personal Reflections section.

# References

The references listed here **must** be cited within the text of your report.

It is not good enough simply to list the references here and have no citations in the text – marks **will** be lost if you leave out the citations or if you use Wikipedia as a reference source.

# Appendix: Personal Reflections

* A brief summary of what you learned
* What you liked about the project
* What you didn’t like about the project
* What would you have done differently if you could do it again
* Any other recommendations/feedback for the Lecturer and/or future Students

# Appendix: Project Planning and Management

(Ask if we can put Diary here)

* Gantt chart
* Description of your Gantt chart
* How you managed the project on a week-to-week basis