**OPERATING SYSTEMS PROJECT PROPOSAL**

**Parallel Programming - Comparison of sorting Algorithms using Pthreads vs. OpenMP**

# Group Member

Huzaifa Abid 17K-3807

Mustafa Manga 17K-3795

Danish Abdullah 17K-3720

# Introduction

Parallel programming is a type of computation in which many calculations or the execution of processes are carried out simultaneously. Large problems can often be divided into smaller ones, which can then be solved at the same time to obtain the result much quicker. There are many ways to achieve parallel programming but we will be discussing two of them right now i.e. Pthreads and OpenMP.

Sorting algorithms are of different types; some use iterations which are dependent on each other (bubble sort, insertion sort etc.) hence threading would not be of much use because threads will have to wait for other threads to provide their results to proceed, which will eventually result in a sequential programming result. However some algorithms (merge sort and quick sort) have iterations which are independent of each other, therefore; threading will be useful to increase their performance by reducing the processing time.

Pthreads and OpenMP both have their pros and cons, sometimes Pthreads are better with an algorithm and other times OpenMP is better. This is what we will be finding out that which threading method is more suitable with the sorting algorithms.

# Methodology

The sorting algorithms the program will test are as follows:

* Merge sort
* Quick sort
* Radix sort

First, all three algorithms will be implemented using both Pthreads and OpenMP; hence the iterations of two of them are independent of each other (Merge, Quick sort). For Radix sort, data parallelism approach was used to split the workload evenly between the threads, all threads work on the same data set, but each thread is assigned a different block of data to sort, threading will decrease their processing time immensely but the question is which threading method will be better? And that is what our project will answer.

The program will run multiple data sets on the algorithms using both threading methods and compare their time complexities. Data sets will be of different types, some of which are as follows:

* Small data
* Large data
* Already sorted data
* Reverse sorted data

Time complexity will be calculated by computing the time taken by the algorithm to completely sort the data and will be displayed in nanoseconds.

Results will be compared to check which threading method worked better with which algorithm.