



Assignment 2 - pthreads

Objectives

1. Introducing threads concepts and POSIX threads library.
2. Implementing popular algorithms as multi-threaded ones.

Problem Statement

It is required to implement two popular algorithms as multi-threaded ones. They are as follows:

Matrix Multiplication

It is required to implement two variations of this algorithm:

1. The computation of each element of the output matrix happens in a thread.
2. The computation of each row of the output matrix happens in a thread.

For both variations, you have to compute the elapsed time for each of them, compare them and justify your answer.

The program should read two input matrices from a file in the following format:

```
[number of rows of 1st matrix] [number of columns of 1st matrix]
1st matrix entries
[number of rows of 2nd matrix] [number of columns of 2nd matrix]
2nd matrix entries
```

Note: □ for clarity

example input:

```
3 5
1 -2 3 4 5
```

```
1 2 -3 4 5
-1 2 3 4 5
5 4
-1 2 3 4
1 -2 3 4
1 2 -3 4
1 2 3 -4
-1 -2 -3 -4
```

output format:

```
[result matrix entries]
END1 [elapsed time of procedure 1]
[result matrix entries]
END2 [elapsed time of procedure 2]
```

Once again please note that [] is for clarity you shouldn't print it

Merge Sort

Merge sort is an $O(n \log n)$ comparison-based sorting algorithm. It is a divide and conquers algorithm. Conceptually, a merge sort works as follows:

If the list is of length 0 or 1, then it is already sorted. Otherwise:

1. Divide the unsorted list into two sub-lists of about half the size.
2. Sort each sub-list recursively by re-applying the merge sort.
3. Merge the two sub-lists back into one sorted list

So you are required to implement it using pthreads. Each time the list is divided; two threads are created to do merge-sort on each half separately. This step is repeated recursively until each sub-list has only one element.

The program should read two input matrices from a file in the following format:

```
[number of elements]
Array elements which are space separated.
```

example input:

```
10
100 20 15 3 4 8 7 -1 0 33
```

When the program finishes, it should print out the sorted array.



Please note that in both cases your program should take the name of the input file as an argument

Deliverables

Complete source code in **TWO FILES** commented thoroughly and clearly.

- 1st filename is your ID-matrix (e.g., 5237-matrix.c, 5237-matrix.cpp, 5237-matrix.C, 5237-matrix.cc, ...)
- 2nd filename is your ID-sort (e.g., 5237-sort.c, 5237-sort.cpp, 5237-sort.C, 5237-sort.cc, ...)

Notes

- Languages used: C/C++.
- Students will work individually.
- You may talk together on the algorithms or functions being used but are **NOT** allowed to look at anybody's code.
- Revise the academic integrity note found on the class web page.