California State University Fullerton

Department of Computer Science

Conner Cook

connerccook@csu.fullerton.edu

Spencer DeMera

spencer.demera@csu.fullerton.edu

Diane Le

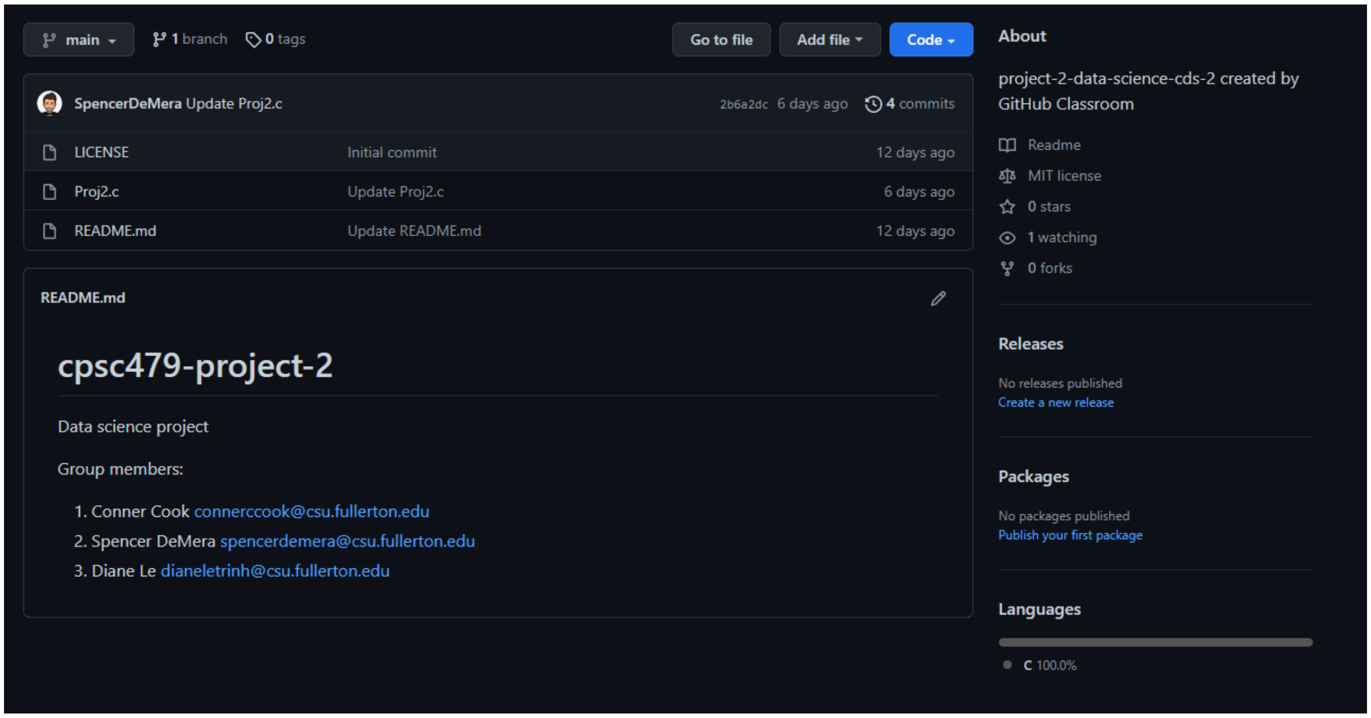
dianeletrinh@csu.fullerton.edu

CPSC 479 : Introduction to High Performance Computing

Project Two Report

13 May 2022

README Screenshot:



SpencerDeMera Update Proj2.c — 2b6a2dc 6 days ago — 4 commits

LICENSE	Initial commit	12 days ago
Proj2.c	Update Proj2.c	6 days ago
README.md	Update README.md	12 days ago

README.md

cpsc479-project-2

Data science project

Group members:

1. Conner Cook connerccook@csu.fullerton.edu
2. Spencer DeMera spencerdemera@csu.fullerton.edu
3. Diane Le dianeletrinh@csu.fullerton.edu

About

project-2-data-science-cds-2 created by GitHub Classroom

Readme
MIT license
0 stars
1 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

C 100.0%

Pseudo Code:

```
MERGE (int* arr, int l, int mid, int r)
    Initialize  i, j, k
    Initialize n1 = mid - l + 1
    Initialize n2 = r - mid

    Initialize *L = malloc(n1 * sizeof(int))
    initialize *R = malloc(n2 * sizeof(int))

    For,
    (i = 0; i < n1; i++)
        Then, L[i] = arr[l + i].

    For,
    (j = 0; j < n2; j++)
        Then, R[j] = arr[mid + 1 + j].

    // Merge the temp arrays bac
        [ i=0, j=0, k=l ]
```

```
          //Compare and reorganize Left and Right temp arrays.
          While loop (i < n1 && j < n2) {
                    IF left array is <= right array,
                              arr[k] = L[i]
                              i++
                     ELSE
                              arr[k] = R[j]
                              j++,  k++

          While (i < n1)
                arr[k] = L[i]
                i++, k++

          While (j < n2)
                arr[k] = R[j]
                j++, k++

VOID mergeSort(int* arr, int l, int r)
          IF (l < r),
                    int mid = l + (r - l) / 2

          #pragma omp task shared(arr)
                    mergeSort(arr, l, mid)

          #pragma omp task shared(arr)
                    mergeSort(arr, mid + 1, r)

          #pragma omp taskwait
                    merge(arr, l, mid, r)

MAIN():
          Initialize i  to zero (i=0)
          double start1, start2
          double end1, end2
          double time1, time2

          Print ("Desired Array Length: ")
          Scan (“%d”, size)
```

```
Initialize mainArr = malloc(SIZE * sizeof(int))
Initialize _2ThreadArr = malloc(SIZE * sizeof(int))
Initialize _4ThreadArr = malloc(SIZE * sizeof(int))

Initialize mainArr with random values of 0 - 2048

Copy the contents of mainArr to _2ThreadArr
Copy the contents of mainArr to _4ThreadArr

Set the number of threads to 2. (N=2)

    Get the start time: start1
    #pragma omp parallel
        #pragma omp simple
        mergeSort(_2ThreadArr, 0, SIZE - 1)
    Get the end time: end1

Set the number of threads to 4. (N=4)

    Get the start time: start2
    #pragma omp parallel
        #pragma omp simple
        mergeSort(_4ThreadArr, 0, SIZE - 1)
    Get the end time: end2

The first elapsed time = end1 - start1;
Print ("Sorted Array on 2 threads in %f seconds", elapsed time)

The second elapsed time = end2 - start2;
Print ("Sorted Array on 8 threads in %f seconds", elapsed time)

Print ("Program Finished...");
```

Code Compilation & Execution Description

Code is written and compiled in bash / Linux:

- Compile Command : `gcc Proj2.c -o exec -fopenmp`
- Run Command : `./exec`

== NOTES ==

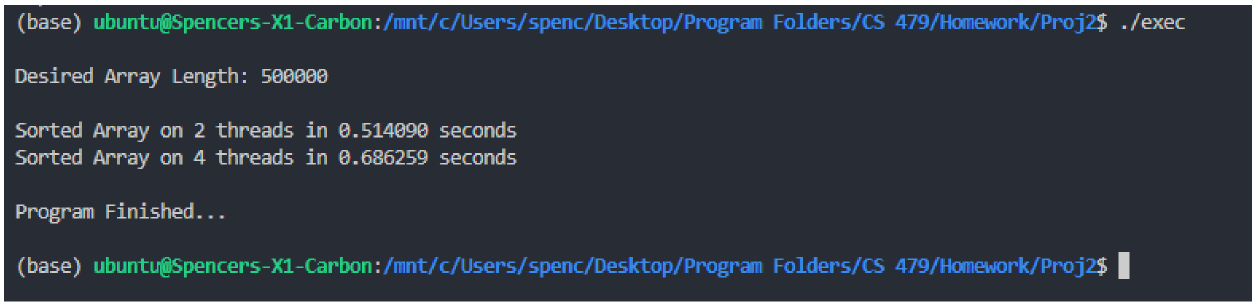
- N represents the intended number of threads to be used per run
- Elapsed times are output at the end of two runs before program termination
    - The value of SIZE is taken as user input, our size used in screenshot is 500,000
- Uncomment lines 156-164 to see the arrays printed (output is much too large to show in screenshots)
- Use of **`-fopenmp`** is necessary for compilation of OpenMP files

Code Output (in VS Code WSL Terminal)

NOTES:

- ***All outputs running on array of length SIZE when SIZE = 500,000***
- ***Only single screenshot since both values of N are ran in same file, rather than 2 separate ones, to compare execution times***

Output running on N number of threads where N = 2 and N = 4



```
(base) ubuntu@Spencers-X1-Carbon:/mnt/c/Users/spenc/Desktop/Program Folders/CS 479/Homework/Proj2$ ./exec

Desired Array Length: 500000

Sorted Array on 2 threads in 0.514090 seconds
Sorted Array on 4 threads in 0.686259 seconds

Program Finished...

(base) ubuntu@Spencers-X1-Carbon:/mnt/c/Users/spenc/Desktop/Program Folders/CS 479/Homework/Proj2$ 
```