CSE 316

Operating System Assignment

Problem no: 2

Name: Rittike Ghosh

Student ID: 11810568

Section: K18SJ

Email: rittikghosh12rg@gmail.com

GitHub link: https://github.com/RittikeGhosh/multithreading_in_c_problem_2

Problem Statement Q2:

Write a multithreaded program that calculates various statistical values for a list of numbers. This program will be passed a series of numbers on the command line and will then create three separate worker threads. One thread will determine the average of the numbers, the second will determine the maximum value, and the third will determine the minimum value.

For example,

1. Suppose your program is passed the integers 90 81 78 95 79 72 85
2. The program will report:

   The average value is 82
   The minimum value is 72
   The maximum value is 95

3. The variables representing the average, minimum, and maximum values will be stored globally. The worker threads will set these values, and the parent thread will output the values once the workers have exited.

Problem Analysis:

In this problem we must take inputs as multiple numbers from the user. After we take the inputs, we need to find the *average, minimum and maximum* of these inputs. To do that we need to perform all the operations individually at the same time as three different threads. So, we must use the concept of multithreading to perform more than one task at the same time. To apply this concept and solve the problem we will use C language.

Algorithm:

1. Set int arg, char **args;
2. If *inputs* not given in the execution line
    1. arg = read (“Enter the number inputs”)
    2. args = read (“Enter the input”)
3. create 3 new threads: t1, t2, t3
4. calculate average with the thread t1
5. find maximum value with thread t2
6. find minimum value with thread t3
7. wait for all the threads to execute completely
8. exit ()

Complexity:

As there 3 three different thread created, each thread will require n comparisons to successfully complete the operation.

Therefore, the *Complexity:* ***O(n)***

Code:

Main function

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>
#define MAX_STRING_LEN 20
typedef long double l_double;
typedef unsigned long long int ull_int;

void* average(void* data);
void* max(void* data);
void* min(void* data);

struct Data{
    ull_int argn;
    l_double* arr_data;
};


int main(ull_int argn, char** args)
{
    // welcome message
    system("clear");
    printf("\n##############################################################################");
    printf("\n#                                                                            #");
    printf("\n#            Welcome: I will help you solving the problem                    #");
    printf("\n#       I will calculate the MIN, MAX and AVERAGE with threads               #");
    printf("\n#                                :-> :-) :->                                 #");
    printf("\n#                                                                            #");
    printf("\n##############################################################################");
    printf("\n\n");
    // check for user input in the execution line
    if (argn == 1)
    {
    printf("It seems that you did not pass any number.\n");
    printf("[You can pass the data(seperated with space) in the execution line of the application]\n");
    printf("Like this ==> ./a.out [[arg1] [arg2] [...]]\n");
    printf("But don't worry I can help you... You can give the data here too.\n\n");
    // ask user for continue to provide input now
    fflush(stdin);

    do{
       char choice = 'y';
       printf("Do you want to give inputs now ?(y/n)  ");
       scanf(" %c",&choice);
       if(choice == 'n')
        goto end;
       else if(choice == 'y')
        break;
    }while(1);

    printf("\n");
        printf("Enter the number of data elements to be entered : ");
    scanf("%llu", &argn);
    ++argn;

    // initializing args
    args = (void*)malloc(argn * sizeof(char*));
    for(ull_int i = 0; i < argn; i++)
        args[i] = (void*)malloc(MAX_STRING_LEN * sizeof(char));

    printf("\n[Enter the numbers adjacent to the staments](MAX DIGIT COUNT = 20)\n");
    for (ull_int i = 1; i<argn; i++)
    {
```

```
    for (ull_int i = 1; i<argn; i++)
    {
        printf("Enter the element %llu : ", i);
        scanf("%s", args[i]);
    }
    }
    // create instance of the struct Data
    struct Data object = {
        argn - 1,
        (void*)calloc(argn-1, sizeof(l_double))
    };

    for(ull_int i = 1; i < argn; i++)
        object.arr_data[i-1] = strtold(args[i], NULL);

    printf("\n*The provided values(rounded to 3 decimal values) are : \n=> [ ");
    for(ull_int i = 0; i < argn - 1; i++)
        printf("%0.3Lf ,", object.arr_data[i]);
    printf("\b]\n\n");

    pthread_t averageThread, minThread, maxThread;
    pthread_create(&averageThread, NULL, average, &object);
    pthread_create(&minThread, NULL, min, &object);
    pthread_create(&maxThread, NULL, max, &object);
    pthread_join(maxThread, NULL);
    pthread_join(minThread, NULL);
    pthread_join(averageThread, NULL);
    printf("\n(The values are rounded to 3 decimal values)");

    end:
    printf("\n\nThank You for your visit :-) :-> \n");
    printf("\n");
    getchar();
    return 0;
}
```

Average finding function

```
void* average(void* data)
{
    struct Data* object = (struct Data*)data;
    l_double sum = 0, average;

    for(ull_int i = 0; i < object->argn; i++)
        sum += object->arr_data[i];

    average = sum / object->argn;

    printf("\nTHE AVERAGE VALUE IS : %0.3Lf", average);
    return NULL;
}
```

Minimum finding Function

```
void* min(void* data)
{
    struct Data* object = (struct Data*)data;
    l_double min = object->arr_data[0];

    for(ull_int i = 0; i < object->argn; i++)
    {
        if(min > object->arr_data[i])
            min = object->arr_data[i];
    }

    printf("\nTHE MINIMUM VALUE IS : %0.3Lf", min);
    return NULL;
}
```

Maximum finding Function

```
131    void* max(void* data)
132    {
133        struct Data* object = (struct Data*)data;
134        l_double max = object->arr_data[0];
135
136        for(ull_int i = 0; i < object->argn; i++)
137        {
138            if(max < object->arr_data[i])
139                max = object->arr_data[i];
140        }
141
142        printf("\nTHE MAXIMUM VALUE IS : %0.3Lf", max);
143        return NULL;
144    }
145
```

Outputs:

1. *When inputs are passed in the execution line:*
   E.g. ***./a.out 1 2 3 4 5 6***

```
################################################################################
#                                                                              #
#            Welcome: I will help you solving the problem                      #
#       I will calculate the MIN, MAX and AVERAGE with threads                 #
#                              :-> :-) :->                                     #
#                                                                              #
################################################################################


*The provided values(rounded to 3 decimal values) are :
=> [ 1.000 ,2.000 ,3.000 ,4.000 ,5.000 ,6.000 ]


THE AVERAGE VALUE IS : 3.500
THE MINIMUM VALUE IS : 1.000
THE MAXIMUM VALUE IS : 6.000
(The values are rounded to 3 decimal values)

Thank You for your visit :-) :->
```

2. *When inputs are not passed in the execution line*

E.g. **./a.out**

```
##########################################################################
#                                                                        #
#            Welcome: I will help you solving the problem                #
#       I will calculate the MIN, MAX and AVERAGE with threads           #
#                              :-> :-) :->                               #
#                                                                        #
##########################################################################

It seems that you did not pass any number.
[You can pass the data(seperated with space) in the execution line of the application]
Like this ==> ./a.out [[arg1] [arg2] [...]]
But don't worry I can help you... You can give the data here too.

Do you want to give inputs now ?(y/n)  y

Enter the number of data elements to be entered : 6

[Enter the numbers adjacent to the staments](MAX DIGIT COUNT = 20)
Enter the element 1 : 1
Enter the element 2 : 2
Enter the element 3 : 3
Enter the element 4 : 4
Enter the element 5 : 5
Enter the element 6 : 6

*The provided values(rounded to 3 decimal values) are :
=> [ 1.000 ,2.000 ,3.000 ,4.000 ,5.000 ,6.000 ]


THE AVERAGE VALUE IS : 3.500
THE MINIMUM VALUE IS : 1.000
THE MAXIMUM VALUE IS : 6.000
(The values are rounded to 3 decimal values)

Thank You for your visit :-) :->
```