OS MID SEM

Mihir Shri
COE18B064

Question:
Develop a C program that generates the histogram (frequency count of unique characters in the input text) using multiple threads (thread numbers to match with unique character count) , each thread performing the counting for that respective character. Compare the efficiency of the threaded version over its equivalent serial version.

Code:

---------------------------------------For parallel execution-------------------------------------

```
#include<stdio.h>
#include<unistd.h>
#include<string.h>
#include<sys/wait.h>
#include<stdlib.h>
#include<pthread.h>
#include<time.h>
#define totchars 128
#define MAX 26

int freq[totchars] = {0};
char text[2000];
void *runner(void *param);

int main()
{
   printf("-----------------Parallel Execution-----------------\n");
   clock_t t;

         printf("Some random string generated...\n\n");
```

```
        //Data for generating random string
    char alphabet[MAX] = { 'a', 'b', 'c', 'd', 'e', 'f', 'g',
                          'h', 'i', 'j', 'k', 'l', 'm', 'n',
                          'o', 'p', 'q', 'r', 's', 't', 'u',
                          'v', 'w', 'x', 'y', 'z' };

     //generating random string
        for (int i = 0; i < 2000; i++)
        text[i] = alphabet[rand() % MAX];

        //clock started
        t = clock();

    pthread_t tid[2000];

    int j = 0;

    //looping through the entire string and creating a thread only if a character is not
visited i.e. unique
    for (int i = 0; i < strlen(text); i++)
    {
         if (freq[(int)text[i]] == 0)
         {
                freq[(int)text[i]] = 1;
                pthread_create(&tid[j++], NULL, runner, (void*)i);
         }
    }

    int num = j;
    j = 0;

    //Joining the threads
    while (j < num)
    {
         pthread_join(tid[j++], NULL);
    }

        t = clock() - t;
        double time_taken = ((double)t)/CLOCKS_PER_SEC;
        printf("Threaded program took %f seconds to execute \n\n", time_taken);
        printf("Total number of threads created = %d\n", num);

    return 0;
}

void *runner(void *param)
{
    int *n = (int *)param;
    int i = n;
    int j;

    //The runner function loops through the entire string and counts the occurance of the
character which is sent by the main function
    for (j = 0; j < strlen(text); j++)
    {
        if (text[j] == text[i])
        {
            freq[(int)text[i]]++;
        }
    }

    printf("%c (%d) ", text[i], freq[(int)text[i]]-1);
    for(j=0; j < freq[(int)text[i]]-1; j++)
        printf("*");

    printf("\n");


    pthread_exit(0);
}
```

---For serial execution---

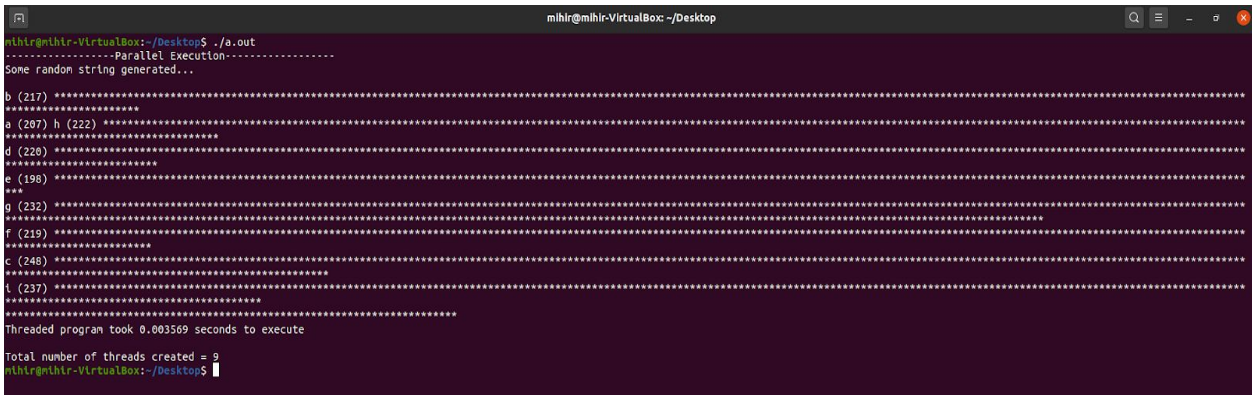
```
#include<stdio.h>
#include<unistd.h>
#include<string.h>
#include<sys/wait.h>
```

```
#include<stdlib.h>
#include<pthread.h>
#include<time.h>
#define totchars 128
#define MAX 26
int freq1[totchars] = {0};
char text1[3000];
void process(int i, char text1[]);

int main()
{
        clock_t t;
        printf("------------------Serial Execution------------------\n");

        //Data for generating random string
        char alphabet[MAX] = { 'a', 'b', 'c', 'd', 'e', 'f', 'g',
                            'h', 'i', 'j', 'k', 'l', 'm', 'n',
                            'o', 'p', 'q', 'r', 's', 't', 'u',
                            'v', 'w', 'x', 'y', 'z' };

        //generating random string
        for (int i = 0; i < 3000; i++)
        text1[i] = alphabet[rand() % MAX];

        //clock started
        t = clock();

        //looping through the entire string and calling the process function only if a
character is not visited i.e. unique
        for(int i = 0; i < strlen(text1); i++)
        {
        if (freq1[(int)text1[i]] == 0)
        {
        freq1[(int)text1[i]] = 1;
        process(i, text1);
        }
        }


        t = clock() - t;
        double time_taken1 = ((double)t)/CLOCKS_PER_SEC;

        printf("Serial execution took %f seconds to execute\n", time_taken1);

        return 0;
}

void process(int i, char text1[])
{
        //The process function loops through the entire string and counts the occurance
of the character which is sent by the main function
        for (int j = 0; j < strlen(text1); j++)
        {
        if (text1[j] == text1[i])
        {
        freq1[(int)text1[i]]++;
        }
        }

        printf("%c (%d) ", text1[i], freq1[(int)text1[i]]);
        for(int j = 0; j < freq1[(int)text1[i]]; j++)
        printf("*");


        printf("\n");
}
```

Output Screenshots:

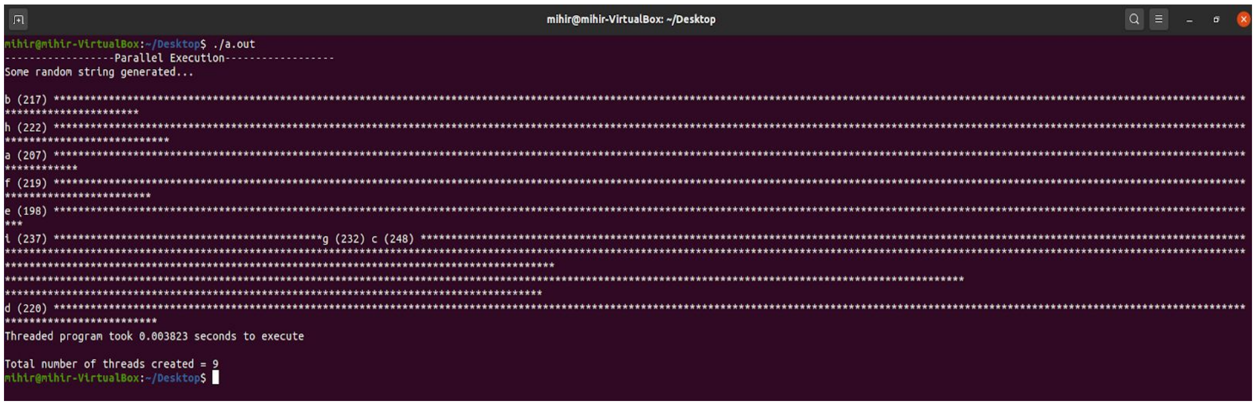
--For parallel execution--------------------------------------

1.



```
mihir@mihir-VirtualBox:~/Desktop$ ./a.out
-----------------Parallel Execution-----------------
Some random string generated...

b (217) ***************************************************************************************************************************************************************************************************
**********************
a (207) h (222) *******************************************************************************************************************************************************************************************
***********************************
d (220) ***************************************************************************************************************************************************************************************************
*************************
e (198) ***************************************************************************************************************************************************************************************************
***
g (232) ***************************************************************************************************************************************************************************************************
**********************************************************************************************************************************************************************************
f (219) ***************************************************************************************************************************************************************************************************
************************
c (248) ***************************************************************************************************************************************************************************************************
*****************************************************
i (237) ***************************************************************************************************************************************************************************************************
******************************************
**************************************************************************
Threaded program took 0.003569 seconds to execute

Total number of threads created = 9
mihir@mihir-VirtualBox:~/Desktop$ 
```

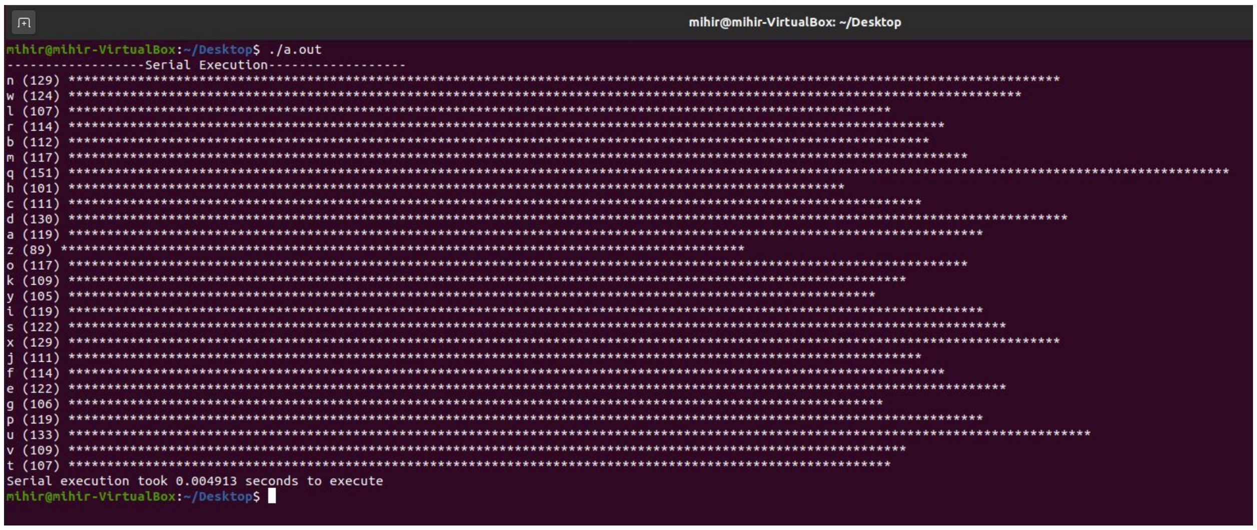
2.



```
mihir@mihir-VirtualBox:~/Desktop$ ./a.out
-----------------Parallel Execution-----------------
Some random string generated...

b (217) ***************************************************************************************************************************************************************************************************
**********************
h (222) ***************************************************************************************************************************************************************************************************
***************************
a (207) ***************************************************************************************************************************************************************************************************
************
f (219) ***************************************************************************************************************************************************************************************************
************************
e (198) ***************************************************************************************************************************************************************************************************
***
i (237) ********************************************g (232) c (248) ***************************************************************************************************************************************
***********************************************************************************************************************************************************************************************************
******************************************************************************************
*************************************************************************************************************************************************************
****************************************************************************************
d (220) ***************************************************************************************************************************************************************************************************
*************************
Threaded program took 0.003823 seconds to execute

Total number of threads created = 9
mihir@mihir-VirtualBox:~/Desktop$ 
```

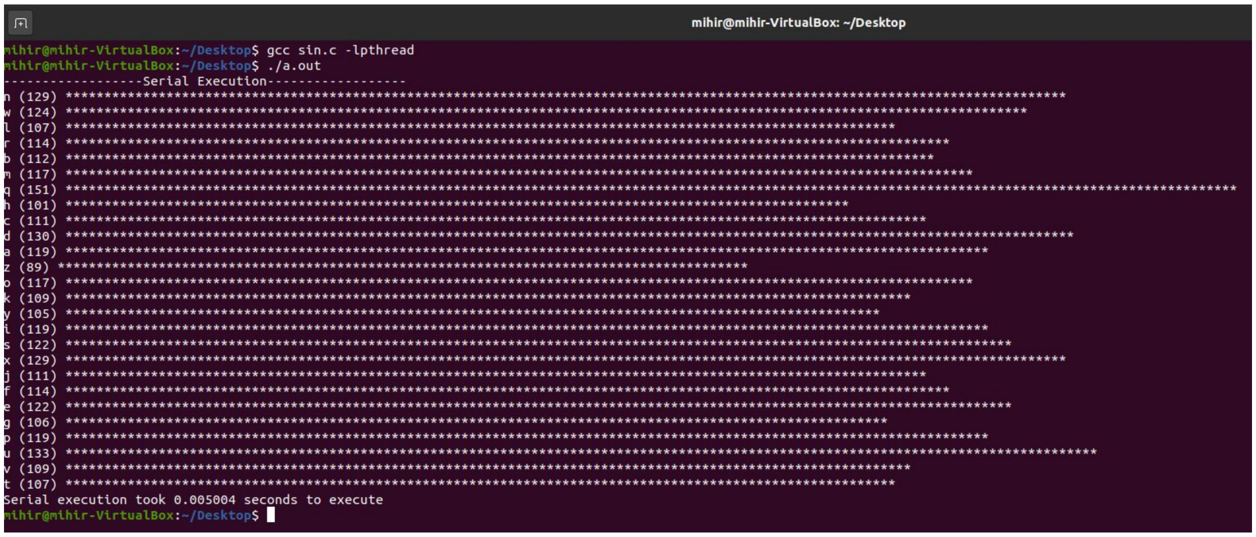
--For serial execution--

1.



```
mihir@mihir-VirtualBox:~/Desktop$ ./a.out
-----------------Serial Execution-----------------
n (129) *********************************************************************************************************************************
w (124) ****************************************************************************************************************************
l (107) ***********************************************************************************************************
r (114) ******************************************************************************************************************
b (112) ****************************************************************************************************************
m (117) *********************************************************************************************************************
q (151) *******************************************************************************************************************************************************
h (101) *****************************************************************************************************
c (111) ***************************************************************************************************************
d (130) **********************************************************************************************************************************
a (119) ***********************************************************************************************************************
z (89) *****************************************************************************************
o (117) *********************************************************************************************************************
k (109) *************************************************************************************************************
y (105) *********************************************************************************************************
i (119) ***********************************************************************************************************************
s (122) **************************************************************************************************************************
x (129) *********************************************************************************************************************************
j (111) ***************************************************************************************************************
f (114) ******************************************************************************************************************
e (122) **************************************************************************************************************************
g (106) **********************************************************************************************************
p (119) ***********************************************************************************************************************
u (133) *************************************************************************************************************************************
v (109) *************************************************************************************************************
t (107) ***********************************************************************************************************
Serial execution took 0.004913 seconds to execute
mihir@mihir-VirtualBox:~/Desktop$
```

2.



```
mihir@mihir-VirtualBox:~/Desktop$ gcc sin.c -lpthread
mihir@mihir-VirtualBox:~/Desktop$ ./a.out
-----------------Serial Execution-----------------
n (129) *********************************************************************************************************************************
w (124) ****************************************************************************************************************************
l (107) ***********************************************************************************************************
r (114) ******************************************************************************************************************
b (112) ****************************************************************************************************************
m (117) *********************************************************************************************************************
q (151) *******************************************************************************************************************************************************
h (101) *****************************************************************************************************
c (111) ***************************************************************************************************************
d (130) **********************************************************************************************************************************
a (119) ***********************************************************************************************************************
z (89) *****************************************************************************************
o (117) *********************************************************************************************************************
k (109) *************************************************************************************************************
y (105) *********************************************************************************************************
i (119) ***********************************************************************************************************************
s (122) **************************************************************************************************************************
x (129) *********************************************************************************************************************************
j (111) ***************************************************************************************************************
f (114) ******************************************************************************************************************
e (122) **************************************************************************************************************************
g (106) **********************************************************************************************************
p (119) ***********************************************************************************************************************
u (133) *************************************************************************************************************************************
v (109) *************************************************************************************************************
t (107) ***********************************************************************************************************
Serial execution took 0.005004 seconds to execute
mihir@mihir-VirtualBox:~/Desktop$
```

<u>Theory</u>

A frequency distribution shows how often each different value in a set of data occurs. A histogram is the most commonly used graph to show frequency distributions. It looks very much like a bar chart, but there are important differences between them. This helpful data collection and analysis tool is considered one of the best tools to assess the distribution of unordered data. A histogram generator can be used in this case to count and display the frequency of each character in a string using a multithreaded fashion.

Time Complexity

O(K * N)
Where,
K = number of unique characters
N = total number of characters

Code Explanation

---For parallel execution--

Firstly, instead of taking input from the user, a random string has been generated with the use of the **rand()** function from a corpus consisting of 9 alphabets (We can also add more characters if we want).

Now, we loop through the entire string and as and when a character is encountered we change its count to 1. We create a new thread only if a character is unvisited i.e. only if the character is unique. So, **number of unique characters = total number of threads**.

The thread calls the runner function and the looping variable (i) is passed as a parameter to the runner function. Inside the runner function, we again loop through the whole string and count the number of times the character text[i] (i.e. the character at i th position in our text) occurs. So, this way **each thread counts the number of times a particular character occurs**. The runner function also prints the character along with its count.

---For serial execution--

In serial execution, we take a corpus of 26 alphabets and instead of creating a thread for each unique character, we call the **process()** function. This function is the same as our runner function in parallel execution i.e. it counts the occurrence of the character passed by the main function.

Comparing Efficiency

If we see the execution times, **FOR 2000 INPUTS (CHARACTERS)**

For parallel execution,
In screenshot 1: 0.003569 sec
In screenshot 2: 0.003823 sec

For serial execution,
In screenshot 1: 0.004913 sec
In screenshot 2: 0.005004 sec

We notice that **the time taken by the parallel execution is less than the time taken by the serial execution given that the number of inputs is same**. The main reason for that is, for parallel execution, we are taking just 9 unique characters hence only 9 threads are created.

Had we have taken 26 characters complete, then 26 threads would've been needed at most and that would've taken more time than the serial execution. There are plenty of reasons for that, the major reason is explained below.

Reason

To ensure that a parallel algorithm fulfils its intent, the processors needs to be synchronized. If not, two separate concurrent threads might access the same data and try to process it, known as race condition. As a result, the data could end up corrupt or cause the application to either fail its purpose or provide false data. To avoid this, techniques like read- and write privileges, semaphores or mutual exclusion can be used.

As observed by the outputs the multi threaded aspect should take a lesser amount of time as compared to serial sorting algorithms, but a phenomenon called parallel slowdown occurs in this case.

In this, the parallelisation of an algorithm makes the execution of an application to run slower. This is most often related to the communication between processing threads.

After a certain number threads have been created, in relation to the purpose of the application, threads will spend most of its time communicating with each other rather than processing data. With the creation of each processing thread, comes a cost in terms of an overhead. When the total cost of the overhead becomes a greater factor than the extra resources it provides, the parallel slowdown occurs.

A key aspect of an efficient parallel implementation is to distribute the workload evenly over the processors available. While the processor with the smaller part will finish its execution fast in relation to the second one, the overall execution time for the application will be dependent on the processor with the greatest factor. Because of this, a parallel algorithm needs to ensure even balanced workload in order to maximize the performance. For larger size arrays that do not violate memory violation, multithreaded algorithms will perform better than the serial execution of sorting algorithms.

So, in real world applications where the size of input is as large as 100,000, the parallel execution will definitely work better.