**OPERATING SYSTEM ASSIGNMENT**

STUDENT NAME: Gatla Akanksh.

STUDENT ID: 11812121

SECTION: K18ZV

ROLLNO: 15

EMAIL ADDRESS: [gattla.akanksh@gmail.com](mailto:gattla.akanksh@gmail.com)

GITHUB LINK:

**Problem:**

**Question no: 2**

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, suppose your program is passed the integers 90 81 78 95 79 72 85.

The program will report

The average value is 82

The minimum value is 72

The maximum value is 95

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.

**Concepts used:**

It’s a multithreaded program where each thread is assigned a task, which is pre-defined

that are we need to take the elements from user and find average, minimum and maximum using pthread\_create and pthread\_join to call the process and give us the output. It’s has separate thread for each task.

Syantax:

1. pthread\_create: used to create a new thread

int pthread\_create(pthread\_t \* thread, const pthread\_attr\_t \* attr,

void \* (\*start\_routine)(void \*),void \*arg);

1. pthread\_join: used to wait for the termination of a thread.

int pthread\_join(pthread\_t th, void \*\*thread\_return);

**Code:**

#include<iostream>

#include<pthread.h>

#include<cstdlib>

using namespace std;

int arr[100],n,i;

void \*th(void \*array\_ptr)

{

float average;

cout<<"enter how many numbers to u want to add"<<endl;

cin>>n;

int i = 0,sum=0;

cout<<endl<<"enter values to be added"<<endl;

while(i < n)

{

cin>>arr[i];

(i++);

}

for(i=0;i<n;i++)

{

sum = sum + arr[i];

}

cout<<endl<<"Sum is "<<sum<<endl;

average = sum/n;

cout<<endl<<"Average value is :"<<average;

}

void \*th1(void \*array\_ptr)

{

int temp;

temp=arr[0];

for(i=1;i<n;i++)

{

if(temp>arr[i])

{

temp=arr[i];

}

}

cout<<"\nMinimum value is:"<<temp<<endl;

}

void \*th2(void \*array\_ptr)

{

int temp=arr[0];

for(i=1;i<n;i++)

{

if(temp<arr[i])

{

temp=arr[i];

}

}

cout<<"Maximium value:"<<temp;

}

void createThreads(int \*array\_ptr)

{

int i,n;

pthread\_t t1;

if(true)

{

n=pthread\_create(&t1,NULL,&th,(void \*)array\_ptr);

pthread\_join(t1,NULL);

}

pthread\_t t2;

if(true)

{

n=pthread\_create(&t2,NULL,&th1,(void \*)array\_ptr);

pthread\_join(t2,NULL);

}

pthread\_t t3;

if(true)

{

n=pthread\_create(&t3,NULL,&th2,(void \*)array\_ptr);

pthread\_join(t3,NULL);

}

}

int main()

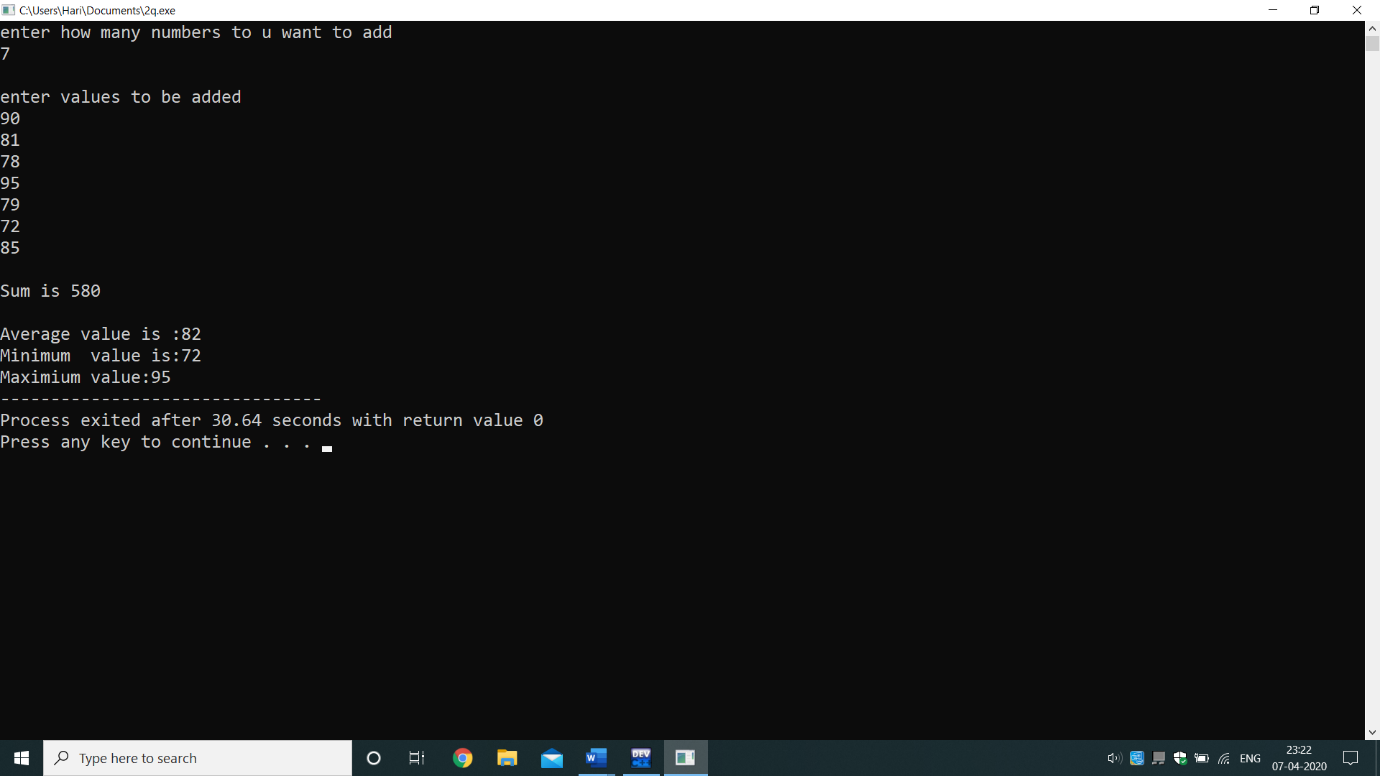
{ int n=1;

int \*array\_ptr =(int\*) malloc(n \* sizeof(int));

createThreads(array\_ptr);

}

**Example Output 1:**



**Algorithm:**

1. Declare constraints such as average, sum, i, n, and array arr[100] etc.
2. Create 4 functions for average, minimum, maximum and for threads.
3. In 1st function Take user input for number of elements in array (to store the values).
4. Enter the elements in the array.
5. The elements stored in array are added with sum=sumac[i].

Average is calculated with the help of sum. As average=sum/n;

1. Now with 2nd function minimum the elements stored will be traversed and will find the minimum.
2. Next with 3rd function maximum the elements stored will be traversed and will find the maximum.
3. In 4th function 3 pthread\_create for each task and pthread\_ join.
4. In main call the function.
5. All the values get printed.

**PURPOSE OF USE:**

This project helps us to understand multithreading concept, creating a thread and thread join also gives us brief idea of using thread and thread functions etc. its use by more than one user at a time and to even manage multiple requests by the same user without having to have multiple copies of the programming running in the computer. Each user request for a program or system service (and here a user can also be another program) is kept track of as a thread with a separate identity.

Threads are known as Lightweight processes. The CPU switches rapidly back and forth among the threads giving illusion that the threads are running in parallel.

**Test Cases:**

1. Taking input from user.
2. Adding up all the elements.
3. Generate average value from the values entered .
4. Generate minimum value from the values entered.
5. Generate maximum value from the values entered .

**Problem:**

Q25.

Design a program using concepts of inter-process communication ordinary pipes in

which one process sends a string message to a second process, and the second process

reverses the case of each character in the message and sends it back to the first process.

For example, if the first process sends the message Hi There, the second process will

return hI tHERE. This will require using two pipes, one for sending the original message

from the first to the second process and the other for sending the modified message from

the second to the first process. You can write this program using either UNIX or

Windows pipes.

**Concepts used:**

This program is based on inter process, where one process which already assigned a task which assigns another task to second process. Second process does the task and returns.

We write string to file and read it from file. getpid give us the parent id.

Syntax:

Write: write(int fd, const void \*buf, size\_t count);

Read: read(int fd, void \*buf, size\_t count);

**Code:**

#include<iostream>

#include<cstring>

#include<fstream>

#include<unistd.h>

#include<sys/types.h>

using namespace std;

void ChangeCase(char str[])

{

for(size\_t i=0;i<strlen(str);i++)

{

if(str[i]>=65 && str[i]<=90)

{

str[i] = str[i] + 32;

}

else if(str[i]>=97 && str[i]<=122)

{

str[i] = str[i] - 32;

}

}

}

int main()

{ char str[20];

cout<<"Enter string: ";

cin.getline(str,20);

int fd[2];

write(fd[1],str,str[20]);

cout<<endl<<endl<<"Pid id for writing pipe is :"<<getpid()<<endl<<endl;

cout<<"The Data written in pipe :"<<str<<endl;

ChangeCase(str);

close(fd[0]);

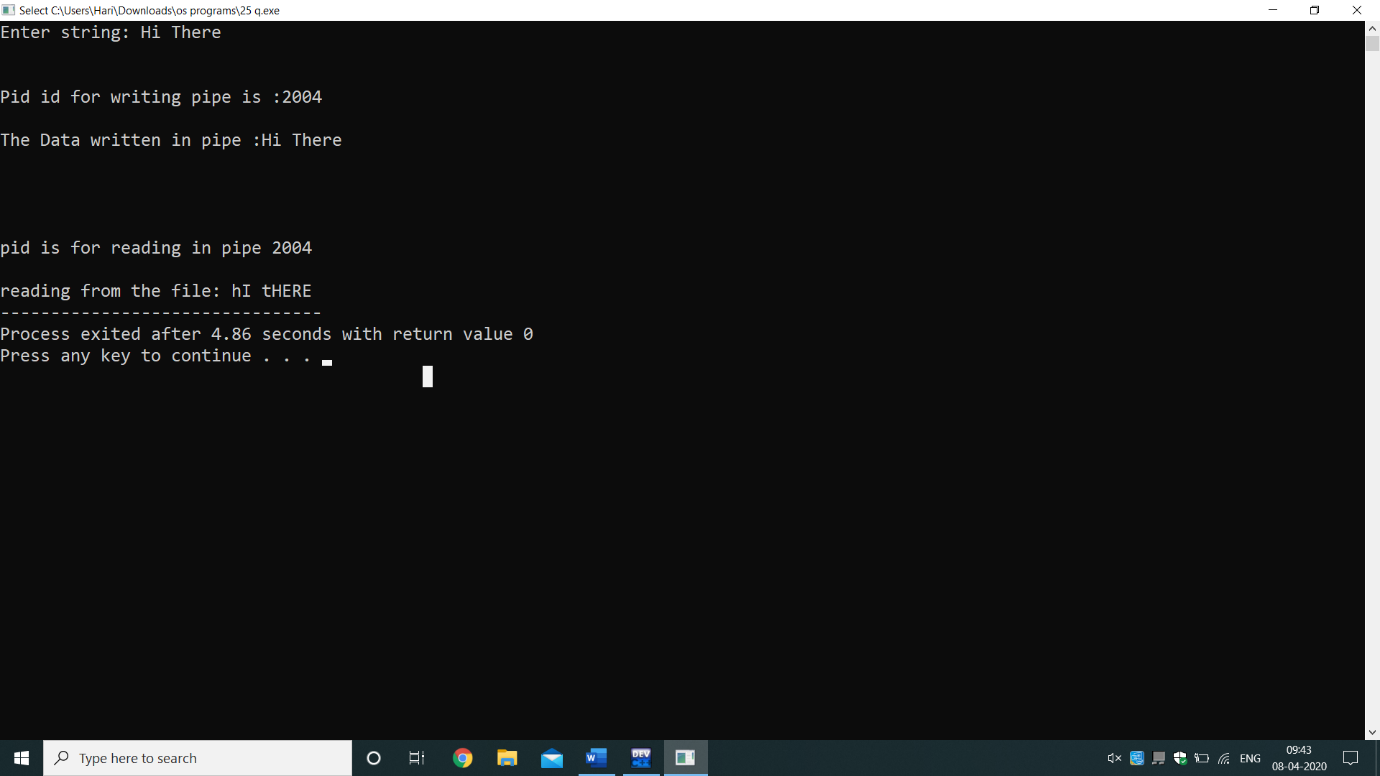
read(fd[0],str,strlen(str));

cout<<endl<<endl<<"\n\npid is for reading in pipe "<<getpid();

cout<<"\n\nreading from the file: "<<str;

}

**Example Output 1:**



**Algorithm:**

1. Declare constraints string array, i.
2. First create a function for change case.
3. 2 loops one for lower to upper and another for upper to lower.
4. In main, take the string input from user. Using getpid get the parent id printed on screen.
5. Write the input to the text file using write syntax. Given above
6. Print the text written.
7. Read the string from the file.
8. Enter the pid number.
9. At last, output of the string from the file which is reversed.

**Purpose of Use:**

This project helps us to understand how to write and read into file. we can even generate the parent id i.e. the current running process. a pipe is a technique for passing information from one program process to another. Unlike other forms of interprocess communication (IPC), a pipe is one-way communication only. Basically, a pipe passes a parameter such as the output of one process to another process which accepts it as input. The system temporarily holds the piped information until it is read by the receiving process. We take two-way communication between processes, two pipes can be set up, one for each task.

**Test Cases:**

1. Taking string input.
2. Writing that string into file and printing on terminal (pipe1).
3. Printing the parent id.
4. reading the string from the file and reverse the case of the string (pipe 2).

Thank you!