西北工業大學
NORTHWESTERN POLYTECHNICAL UNIVERSITY

Lab | Report

Report Subject: OS Experiment - Lab 4

Student ID : 2019380141

Student Name : ABID ALI

Experiment Name : Threads

Objective:

- Practice working with multi-threaded C programs. This lab will give you a good introduction on how to work with the Pthread library. You will be focusing on applying some of the most important concepts discussed in lecture. (Later labs will delve deeper into the topic.)
- Practice working with C function pointers. The C language has a very interesting feature that allows programs to use functions as parameters to other functions. The concept and the syntax for function pointers can be a little intimidating, but this is something you will have to master in order to use Pthreads.

Equipment:

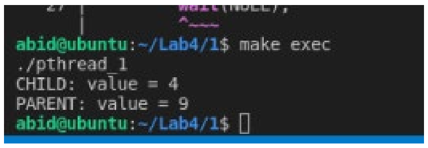
VMware with Ubuntu Linux

Methodology:

Experiment 1: data sharing

1) What would be the output from the program at LINE A and LINE B? please use the data sharing mechanism to explain it.

Solution:



```
abid@ubuntu:~/Lab4/1$ make exec
./pthread_1
CHILD: value = 4
PARENT: value = 9
abid@ubuntu:~/Lab4/1$
```

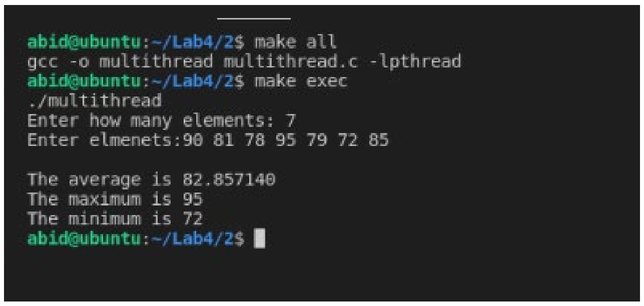
The Line A is the value of CHILD which is 4.
The Line B is the value of PARENT which is 9.

• Threads provide support for expressing concurrency and synchronization.
• Can be used for hiding memory latency
• A thread is a light weight process (has its own stack and execution state, but shares the address space with its parent).
• Hence, threads have local data but also can share global data.

Threads are very useful in modern programming whenever a process has multiple tasks to perform independently of the others.

We can see that Parent and Child has their own value inside their respective functions. We can't see a pointer pointing to those local variable values. So, they are sending different value to the parameter 10 and 5. After subtracting by 1 then 9 and 5 are send back.

Experiment 2: calculates various statistical values using pthread



```
abid@ubuntu:~/Lab4/2$ make all
gcc -o multithread multithread.c -lpthread
abid@ubuntu:~/Lab4/2$ make exec
./multithread
Enter how many elements: 7
Enter elmenets:90 81 78 95 79 72 85

The average is 82.857140
The maximum is 95
The minimum is 72
abid@ubuntu:~/Lab4/2$
```

At the beginning the program will ask us ,how many element we want to use for the calculation.

After pressing the number ,we press the given number(90 81 78 95 79 72 85) in the question and we got the expected value.

The program has reported

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

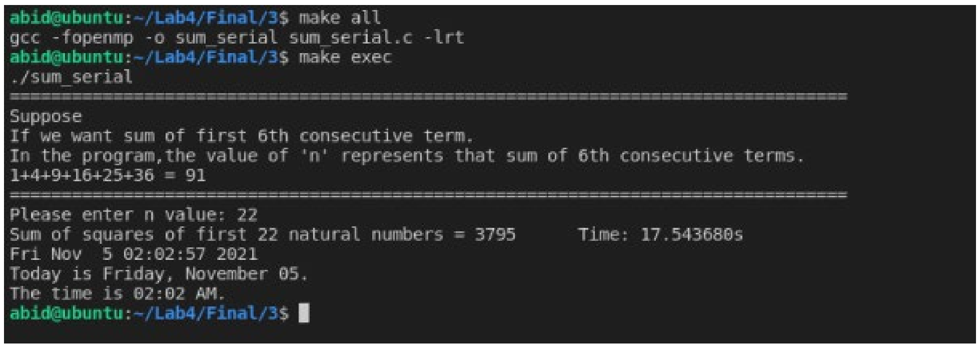
Experiment 3: calculate the sum of number of squares

The formula given to us is 1^2 + (1+1)^2 + (1+2)^2 + … + (a-2)^2 + (a-1)^2

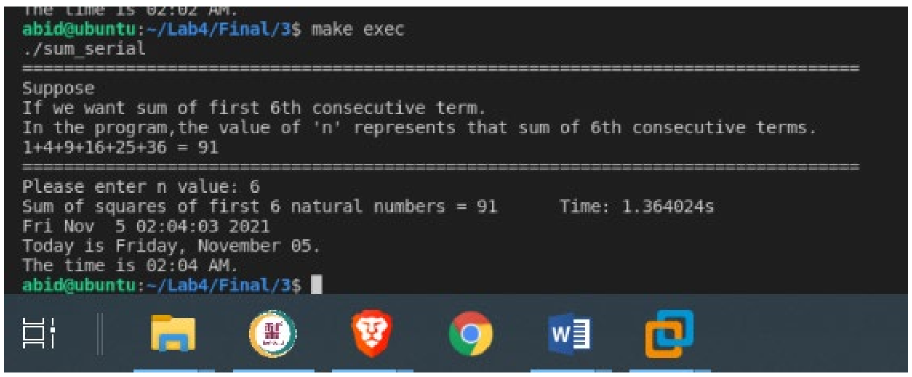
N-th term	: 1 ,2, 3 4 5 6
Square	: 1+4+9+16+25+36
Total	: 1 5 14 30 55 91

Rough Calculation for the program.

Now, implementing in the program



```
abid@ubuntu:~/Lab4/Final/3$ make all
gcc -fopenmp -o sum_serial sum_serial.c -lrt
abid@ubuntu:~/Lab4/Final/3$ make exec
./sum_serial
================================================================================
Suppose
If we want sum of first 6th consecutive term.
In the program,the value of 'n' represents that sum of 6th consecutive terms.
1+4+9+16+25+36 = 91
================================================================================
Please enter n value: 22
Sum of squares of first 22 natural numbers = 3795      Time: 17.543680s
Fri Nov  5 02:02:57 2021
Today is Friday, November 05.
The time is 02:02 AM.
abid@ubuntu:~/Lab4/Final/3$
```

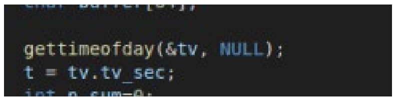


```
The time is 02:02 AM.
abid@ubuntu:~/Lab4/Final/3$ make exec
./sum_serial
================================================================================
Suppose
If we want sum of first 6th consecutive term.
In the program,the value of 'n' represents that sum of 6th consecutive terms.
1+4+9+16+25+36 = 91
================================================================================
Please enter n value: 6
Sum of squares of first 6 natural numbers = 91      Time: 1.364024s
Fri Nov  5 02:04:03 2021
Today is Friday, November 05.
The time is 02:04 AM.
abid@ubuntu:~/Lab4/Final/3$
```

Two requirement of the Question was fulfilled

1) We write a program based on the formula

2) **gettimeofday()** function was used in the the program.



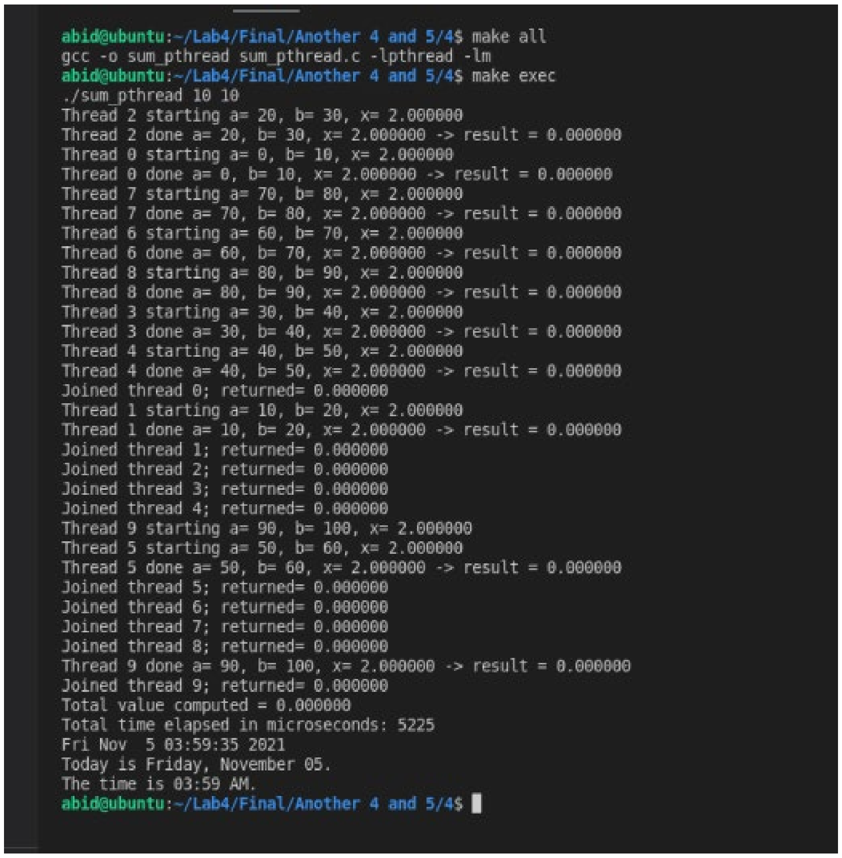
```
gettimeofday(&tv, NULL);
t = tv.tv_sec;
```

Experiment 4: calculate the sum of number of squares using pthread

Questions:

1) What is the type of parallelism that you use? Is the data parallelism or task parallelism? Please explain it.

Data Parallelism



```
abid@ubuntu:~/Lab4/Final/Another 4 and 5/4$ make all
gcc -o sum_pthread sum_pthread.c -lpthread -lm
abid@ubuntu:~/Lab4/Final/Another 4 and 5/4$ make exec
./sum_pthread 10 10
Thread 2 starting a= 20, b= 30, x= 2.000000
Thread 2 done a= 20, b= 30, x= 2.000000 -> result = 0.000000
Thread 0 starting a= 0, b= 10, x= 2.000000
Thread 0 done a= 0, b= 10, x= 2.000000 -> result = 0.000000
Thread 7 starting a= 70, b= 80, x= 2.000000
Thread 7 done a= 70, b= 80, x= 2.000000 -> result = 0.000000
Thread 6 starting a= 60, b= 70, x= 2.000000
Thread 6 done a= 60, b= 70, x= 2.000000 -> result = 0.000000
Thread 8 starting a= 80, b= 90, x= 2.000000
Thread 8 done a= 80, b= 90, x= 2.000000 -> result = 0.000000
Thread 3 starting a= 30, b= 40, x= 2.000000
Thread 3 done a= 30, b= 40, x= 2.000000 -> result = 0.000000
Thread 4 starting a= 40, b= 50, x= 2.000000
Thread 4 done a= 40, b= 50, x= 2.000000 -> result = 0.000000
Joined thread 0; returned= 0.000000
Thread 1 starting a= 10, b= 20, x= 2.000000
Thread 1 done a= 10, b= 20, x= 2.000000 -> result = 0.000000
Joined thread 1; returned= 0.000000
Joined thread 2; returned= 0.000000
Joined thread 3; returned= 0.000000
Joined thread 4; returned= 0.000000
Thread 9 starting a= 90, b= 100, x= 2.000000
Thread 5 starting a= 50, b= 60, x= 2.000000
Thread 5 done a= 50, b= 60, x= 2.000000 -> result = 0.000000
Joined thread 5; returned= 0.000000
Joined thread 6; returned= 0.000000
Joined thread 7; returned= 0.000000
Joined thread 8; returned= 0.000000
Thread 9 done a= 90, b= 100, x= 2.000000 -> result = 0.000000
Joined thread 9; returned= 0.000000
Total value computed = 0.000000
Total time elapsed in microseconds: 5225
Fri Nov  5 03:59:35 2021
Today is Friday, November 05.
The time is 03:59 AM.
abid@ubuntu:~/Lab4/Final/Another 4 and 5/4$ 
```

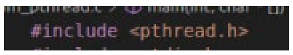
Data parallelism is parallelization across multiple processors in parallel computing environments. It focuses on distributing the data across different nodes, which operate on the data in parallel. It can be applied on regular data structures like arrays and matrices by working on each element in parallel. It contrasts to task parallelism as another form of parallelism.

A data parallel job on an array of *n* elements can be divided equally among all the processors.

We can see in the program, it's following Data Parallelism principle. The big data are distributed different nodes which operate in the data in the parallel manner.

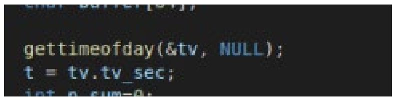
Two requirements of the question was fulfilled

1)We used pthread to write sum_pthread.c then got the desired result.



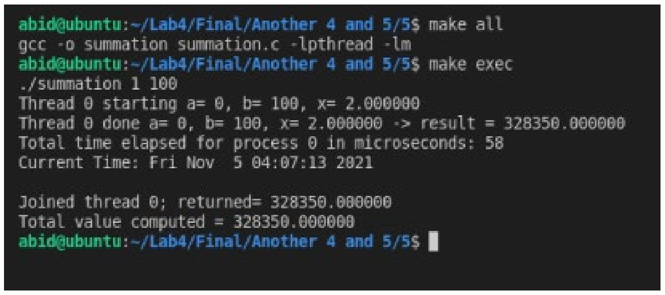
```
#include <pthread.h>
```

2) **gettimeofday()** function was used in the the program



```
gettimeofday(&tv, NULL);
t = tv.tv_sec;
```

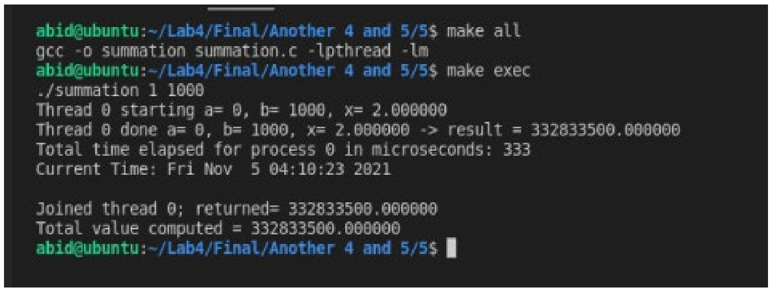
Experiment 5: analysis the performance:



```
abid@ubuntu:~/Lab4/Final/Another 4 and 5/5$ make all
gcc -o summation summation.c -lpthread -lm
abid@ubuntu:~/Lab4/Final/Another 4 and 5/5$ make exec
./summation 1 100
Thread 0 starting a= 0, b= 100, x= 2.000000
Thread 0 done a= 0, b= 100, x= 2.000000 -> result = 328350.000000
Total time elapsed for process 0 in microseconds: 58
Current Time: Fri Nov  5 04:07:13 2021

Joined thread 0; returned= 328350.000000
Total value computed = 328350.000000
abid@ubuntu:~/Lab4/Final/Another 4 and 5/5$ 
```

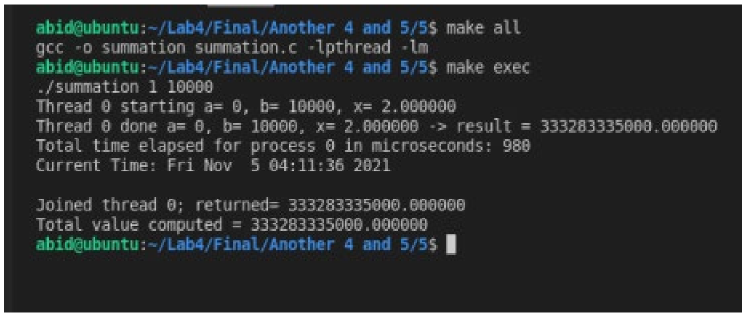
Thread is 1 and the data is **100**. Time is **58 microseconds**



```
abid@ubuntu:~/Lab4/Final/Another 4 and 5/5$ make all
gcc -o summation summation.c -lpthread -lm
abid@ubuntu:~/Lab4/Final/Another 4 and 5/5$ make exec
./summation 1 1000
Thread 0 starting a= 0, b= 1000, x= 2.000000
Thread 0 done a= 0, b= 1000, x= 2.000000 -> result = 332833500.000000
Total time elapsed for process 0 in microseconds: 333
Current Time: Fri Nov  5 04:10:23 2021

Joined thread 0; returned= 332833500.000000
Total value computed = 332833500.000000
abid@ubuntu:~/Lab4/Final/Another 4 and 5/5$ 
```

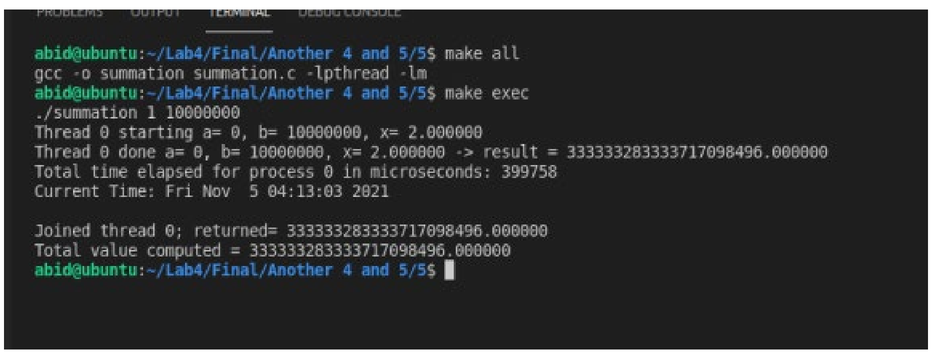
Thread is 1 and the data is **1000**. Time is **333 microseconds**



```
abid@ubuntu:~/Lab4/Final/Another 4 and 5/5$ make all
gcc -o summation summation.c -lpthread -lm
abid@ubuntu:~/Lab4/Final/Another 4 and 5/5$ make exec
./summation 1 10000
Thread 0 starting a= 0, b= 10000, x= 2.000000
Thread 0 done a= 0, b= 10000, x= 2.000000 -> result = 333283335000.000000
Total time elapsed for process 0 in microseconds: 980
Current Time: Fri Nov  5 04:11:36 2021

Joined thread 0; returned= 333283335000.000000
Total value computed = 333283335000.000000
abid@ubuntu:~/Lab4/Final/Another 4 and 5/5$ 
```

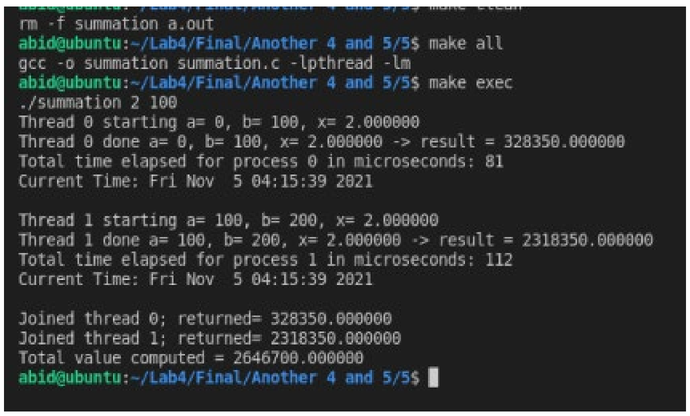
Thread is 1 and the data is **10000**. Time is **980 microseconds**



```
abid@ubuntu:~/Lab4/Final/Another 4 and 5/5$ make all
gcc -o summation summation.c -lpthread -lm
abid@ubuntu:~/Lab4/Final/Another 4 and 5/5$ make exec
./summation 1 10000000
Thread 0 starting a= 0, b= 10000000, x= 2.000000
Thread 0 done a= 0, b= 10000000, x= 2.000000 -> result = 333333283333717098496.000000
Total time elapsed for process 0 in microseconds: 399758
Current Time: Fri Nov  5 04:13:03 2021

Joined thread 0; returned= 333333283333717098496.000000
Total value computed = 333333283333717098496.000000
abid@ubuntu:~/Lab4/Final/Another 4 and 5/5$ 
```

Thread is 1 and the data is **10000000**. Time is **399758 microseconds**

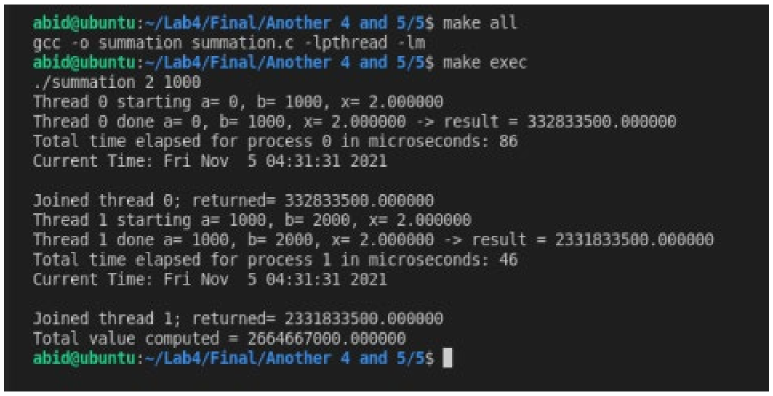


```
rm -f summation a.out
abid@ubuntu:~/Lab4/Final/Another 4 and 5/5$ make all
gcc -o summation summation.c -lpthread -lm
abid@ubuntu:~/Lab4/Final/Another 4 and 5/5$ make exec
./summation 2 100
Thread 0 starting a= 0, b= 100, x= 2.000000
Thread 0 done a= 0, b= 100, x= 2.000000 -> result = 328350.000000
Total time elapsed for process 0 in microseconds: 81
Current Time: Fri Nov  5 04:15:39 2021

Thread 1 starting a= 100, b= 200, x= 2.000000
Thread 1 done a= 100, b= 200, x= 2.000000 -> result = 2318350.000000
Total time elapsed for process 1 in microseconds: 112
Current Time: Fri Nov  5 04:15:39 2021

Joined thread 0; returned= 328350.000000
Joined thread 1; returned= 2318350.000000
Total value computed = 2646700.000000
abid@ubuntu:~/Lab4/Final/Another 4 and 5/5$ 
```

Thread is 2 and the data is **100**. Time is Thread 0 **(81 microseconds) and** Thread 1**(112 microseconds)**

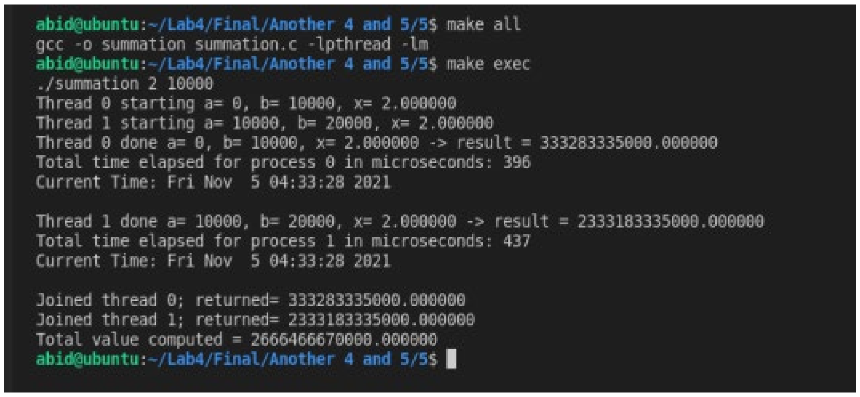


```
abid@ubuntu:~/Lab4/Final/Another 4 and 5/5$ make all
gcc -o summation summation.c -lpthread -lm
abid@ubuntu:~/Lab4/Final/Another 4 and 5/5$ make exec
./summation 2 1000
Thread 0 starting a= 0, b= 1000, x= 2.000000
Thread 0 done a= 0, b= 1000, x= 2.000000 -> result = 332833500.000000
Total time elapsed for process 0 in microseconds: 86
Current Time: Fri Nov  5 04:31:31 2021

Joined thread 0; returned= 332833500.000000
Thread 1 starting a= 1000, b= 2000, x= 2.000000
Thread 1 done a= 1000, b= 2000, x= 2.000000 -> result = 2331833500.000000
Total time elapsed for process 1 in microseconds: 46
Current Time: Fri Nov  5 04:31:31 2021

Joined thread 1; returned= 2331833500.000000
Total value computed = 2664667000.000000
abid@ubuntu:~/Lab4/Final/Another 4 and 5/5$ 
```

Thread is 2 and the data is **1000.** Time is Thread 0 **(86 microseconds)** **and** Thread 1**(46 microseconds)**

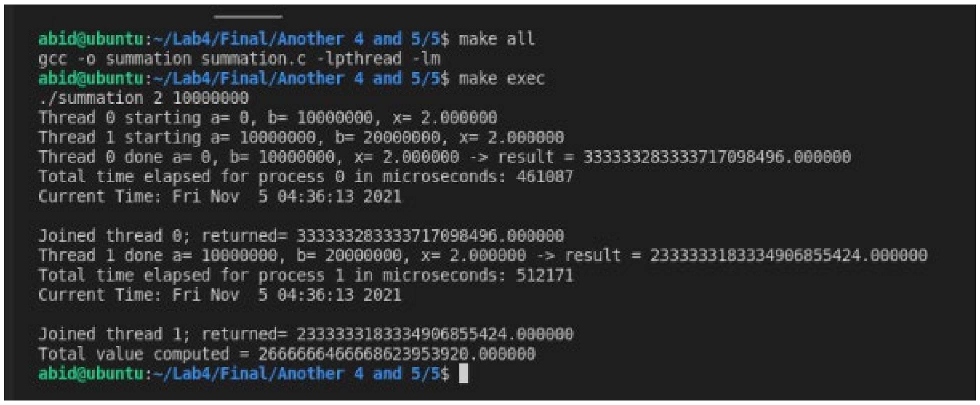


```
abid@ubuntu:~/Lab4/Final/Another 4 and 5/5$ make all
gcc -o summation summation.c -lpthread -lm
abid@ubuntu:~/Lab4/Final/Another 4 and 5/5$ make exec
./summation 2 10000
Thread 0 starting a= 0, b= 10000, x= 2.000000
Thread 1 starting a= 10000, b= 20000, x= 2.000000
Thread 0 done a= 0, b= 10000, x= 2.000000 -> result = 333283335000.000000
Total time elapsed for process 0 in microseconds: 396
Current Time: Fri Nov  5 04:33:28 2021

Thread 1 done a= 10000, b= 20000, x= 2.000000 -> result = 2333183335000.000000
Total time elapsed for process 1 in microseconds: 437
Current Time: Fri Nov  5 04:33:28 2021

Joined thread 0; returned= 333283335000.000000
Joined thread 1; returned= 2333183335000.000000
Total value computed = 2666466670000.000000
abid@ubuntu:~/Lab4/Final/Another 4 and 5/5$ 
```

Thread is 2 and the data is **10000.** Time is Thread 0 **(396 microseconds)** **and** Thread 1**(437 microseconds)**

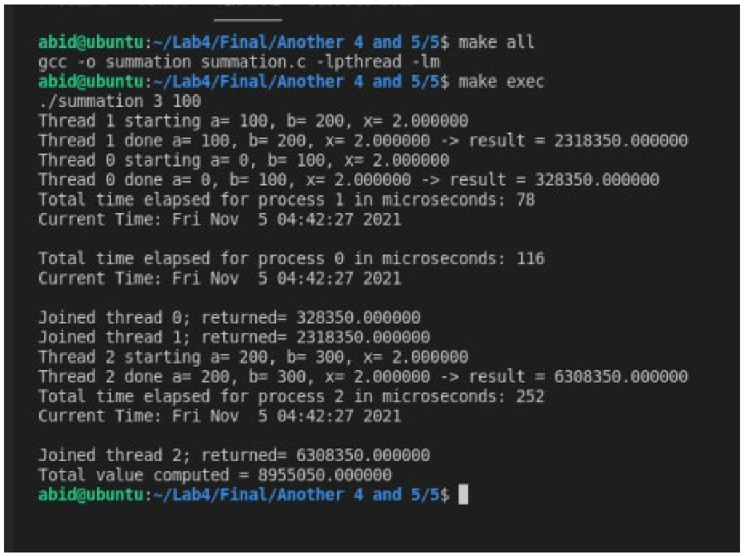


```
abid@ubuntu:~/Lab4/Final/Another 4 and 5/5$ make all
gcc -o summation summation.c -lpthread -lm
abid@ubuntu:~/Lab4/Final/Another 4 and 5/5$ make exec
./summation 2 10000000
Thread 0 starting a= 0, b= 10000000, x= 2.000000
Thread 1 starting a= 10000000, b= 20000000, x= 2.000000
Thread 0 done a= 0, b= 10000000, x= 2.000000 -> result = 333333283333717098496.000000
Total time elapsed for process 0 in microseconds: 461087
Current Time: Fri Nov  5 04:36:13 2021

Joined thread 0; returned= 333333283333717098496.000000
Thread 1 done a= 10000000, b= 20000000, x= 2.000000 -> result = 2333333183334906855424.000000
Total time elapsed for process 1 in microseconds: 512171
Current Time: Fri Nov  5 04:36:13 2021

Joined thread 1; returned= 2333333183334906855424.000000
Total value computed = 2666666466668623953920.000000
abid@ubuntu:~/Lab4/Final/Another 4 and 5/5$ 
```

Thread is 2 and the data is **10000000.** Time is Thread 0 **(461087 microseconds)** **and** Thread 1**(512171 microseconds)**



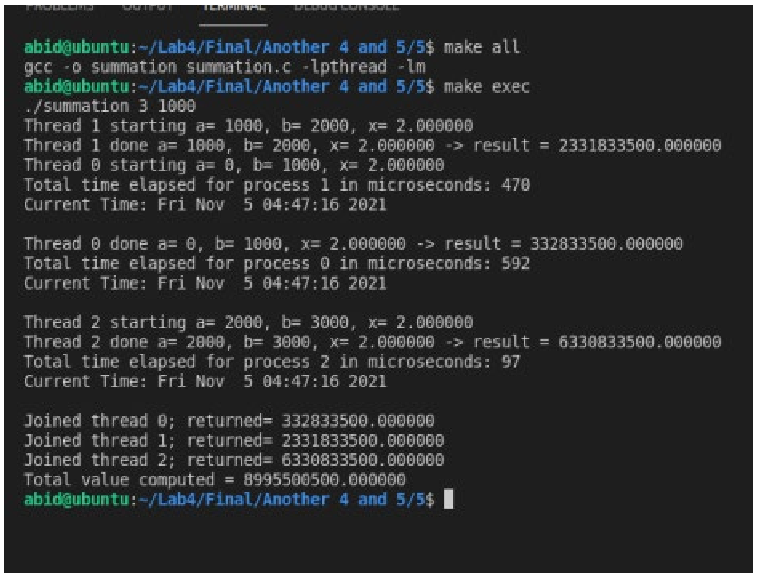
```
abid@ubuntu:~/Lab4/Final/Another 4 and 5/5$ make all
gcc -o summation summation.c -lpthread -lm
abid@ubuntu:~/Lab4/Final/Another 4 and 5/5$ make exec
./summation 3 100
Thread 1 starting a= 100, b= 200, x= 2.000000
Thread 1 done a= 100, b= 200, x= 2.000000 -> result = 2318350.000000
Thread 0 starting a= 0, b= 100, x= 2.000000
Thread 0 done a= 0, b= 100, x= 2.000000 -> result = 328350.000000
Total time elapsed for process 1 in microseconds: 78
Current Time: Fri Nov  5 04:42:27 2021

Total time elapsed for process 0 in microseconds: 116
Current Time: Fri Nov  5 04:42:27 2021

Joined thread 0; returned= 328350.000000
Joined thread 1; returned= 2318350.000000
Thread 2 starting a= 200, b= 300, x= 2.000000
Thread 2 done a= 200, b= 300, x= 2.000000 -> result = 6308350.000000
Total time elapsed for process 2 in microseconds: 252
Current Time: Fri Nov  5 04:42:27 2021

Joined thread 2; returned= 6308350.000000
Total value computed = 8955050.000000
abid@ubuntu:~/Lab4/Final/Another 4 and 5/5$
```

Thread is 3 and the data is 100. Time is Thread 0 **(116 microseconds) ,** Thread 1**(78 microseconds) and** Thread2**(252 microseconds)**



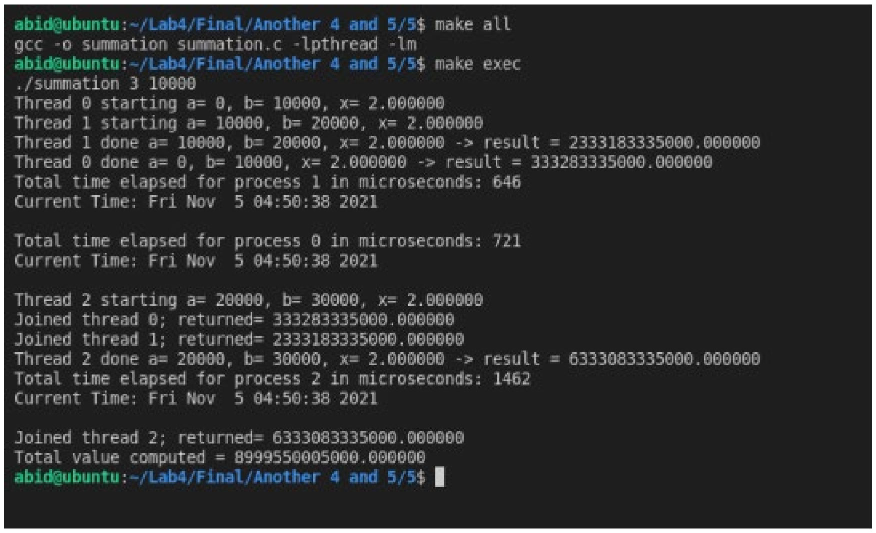
```
abid@ubuntu:~/Lab4/Final/Another 4 and 5/5$ make all
gcc -o summation summation.c -lpthread -lm
abid@ubuntu:~/Lab4/Final/Another 4 and 5/5$ make exec
./summation 3 1000
Thread 1 starting a= 1000, b= 2000, x= 2.000000
Thread 1 done a= 1000, b= 2000, x= 2.000000 -> result = 2331833500.000000
Thread 0 starting a= 0, b= 1000, x= 2.000000
Total time elapsed for process 1 in microseconds: 470
Current Time: Fri Nov  5 04:47:16 2021

Thread 0 done a= 0, b= 1000, x= 2.000000 -> result = 332833500.000000
Total time elapsed for process 0 in microseconds: 592
Current Time: Fri Nov  5 04:47:16 2021

Thread 2 starting a= 2000, b= 3000, x= 2.000000
Thread 2 done a= 2000, b= 3000, x= 2.000000 -> result = 6330833500.000000
Total time elapsed for process 2 in microseconds: 97
Current Time: Fri Nov  5 04:47:16 2021

Joined thread 0; returned= 332833500.000000
Joined thread 1; returned= 2331833500.000000
Joined thread 2; returned= 6330833500.000000
Total value computed = 8995500500.000000
abid@ubuntu:~/Lab4/Final/Another 4 and 5/5$
```

Thread is 3 and the data is 1000. Time is Thread 0 **(592 microseconds) ,** Thread 1**(470 microseconds) and** Thread2**(97 microseconds)**



```
abid@ubuntu:~/Lab4/Final/Another 4 and 5/5$ make all
gcc -o summation summation.c -lpthread -lm
abid@ubuntu:~/Lab4/Final/Another 4 and 5/5$ make exec
./summation 3 10000
Thread 0 starting a= 0, b= 10000, x= 2.000000
Thread 1 starting a= 10000, b= 20000, x= 2.000000
Thread 1 done a= 10000, b= 20000, x= 2.000000 -> result = 2333183335000.000000
Thread 0 done a= 0, b= 10000, x= 2.000000 -> result = 333283335000.000000
Total time elapsed for process 1 in microseconds: 646
Current Time: Fri Nov  5 04:50:38 2021

Total time elapsed for process 0 in microseconds: 721
Current Time: Fri Nov  5 04:50:38 2021

Thread 2 starting a= 20000, b= 30000, x= 2.000000
Joined thread 0; returned= 333283335000.000000
Joined thread 1; returned= 2333183335000.000000
Thread 2 done a= 20000, b= 30000, x= 2.000000 -> result = 6333083335000.000000
Total time elapsed for process 2 in microseconds: 1462
Current Time: Fri Nov  5 04:50:38 2021

Joined thread 2; returned= 6333083335000.000000
Total value computed = 8999550005000.000000
abid@ubuntu:~/Lab4/Final/Another 4 and 5/5$
```

Thread is 3 and the data is **10000**. Time is Thread 0 **(721 microseconds)** **,** Thread 1**(646 microseconds)** **and** Thread2**(1462 microseconds)**

1) When you increase the number of threads to get the summation, whether the running time is shorter or not? Why?

Solution:

As I am using VMware where there is 2 core is allocated for that software.We can notice that,when there is less thread at the beginning with small data,The running time is shorter.But,when the thread and data start increasing it tends to take more time.
For each thread, the OS needs to schedule the execution of the thread. The more threads, the more time the OS spends with the scheduling. Also, there is no guarantee that a thread will be the only program executed by a CPU core. Worst case, all threads are executed on the same core. The resource sharing contingencies might occur too.

2) If there is any problem, explain what causes it and how you can avoid it. Also, remember to modify your source code and submit a version that sure prints out the termination time for each thread.

Solution:

One of the issue is that I am using VMware that has 2 core resource allocated to it.As we tend to tend to add more threads and more data then the running time start to increase.If we use fewer amount of thread and data then we can notice the running time is less.In this way,we can avoid this problem.

Also, we modified the source code and create a better version that sure prints out the termination time for each thread.

Problems:

I was confused with the problem and how to use VScode in linux and new add-ons in the makefile. Couldn't understand how to do the makefile.Not familiar with the concept of thread and implement the code.

Solution:

To solve those problems I looked for information in internet. In order to understand some questions and procedure I also asked the teacher to help me understand them. And provided instructions helped to solve some of my errors during the experiment.

Conclusion:

At the beginning, I was unfamiliar with pipe and how to implement it. Gradually, reading lot of article and reading teachers ppt then I solved those problem one by one. In this experiment ,small helps and suggestions from the teacher was very helpful that saved my time.I enjoyed the practical and learned lot of interesting things.

Attachments:

1) ABID ALI_2019380141_OS(Lab 4).docx
2) ABID ALI_2019380141_OS(Lab 4).pdf
3)Code_ABID ALI_2019380141(Lab-4).zip