



西北工业大学  
NORTHWESTERN POLYTECHNICAL UNIVERSITY

# Lab 4 Report

**Report Subject: OS Experiment - Lab 4**

**Student ID: 2018380039**

**Student Name: Dikshya Kafle**

**Data: 2021/11/22**

# Computer Operating System Experiment

## Laboratory 4

### Threads

#### 1) Objective:

- ❖ Practice working with multi-threaded C programs.
- ❖ Practice working with C function pointers.

#### 2) Equipment:

- ❖ VirtualBox with Ubuntu Linux

#### 3) Experiments:

##### Experiment 1: data sharing

##### Questions:

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

##### Answer:

- ❖ LINE A: CHILD: value = 4
- ❖ LINE B: PARENT: value = 9

When we fork() a new process, the parent and child process are both separate processes and each one of them has its copy of the data. So editing the data in one process does not affect the variables of the other process. However, threads that belong to the same process share the same data. So within the same process, if a thread edits a variables, this change is visible to all the other threads.

In our case, myvalue=10 in the parent process and myvalue=5 in the child process (Two non-shared separate copies of data). Each thread performs value = myvalue -1 (value is also no shared between the 2 processes and each thread only edits the variable belonging to its process). As a result we get the following output:

*CHILD: value = 4*

*PARENT: value = 9*

```
dikshya@dikshya-VirtualBox: ~/Desktop/Lab 4 OS
dikshya@dikshya-VirtualBox:~/Desktop/Lab 4 OS$ touch thread1.c
dikshya@dikshya-VirtualBox:~/Desktop/Lab 4 OS$ gcc thread1.c -o thread -lpthread
dikshya@dikshya-VirtualBox:~/Desktop/Lab 4 OS$ ./thread
CHILD: value = 4
PARENT: value = 9
dikshya@dikshya-VirtualBox:~/Desktop/Lab 4 OS$
```

## Experiment 2: calculates various statistical values

❖ The code:

```
Open  various.c
~/Desktop/Lab 4 OS

1 #include<stdio.h>
2 #include<sys/types.h>
3 #include<pthread.h>
4 #include<stdlib.h>
5
6 #define TNUMBERS 3
7 #define MAXNUMBERS 50
8
9 void *findAverage(void *param);
10 void *findMax(void *param);
11 void *findMin(void *param);
12
13 int numbers[MAXNUMBERS];
14 float average=0;
15 int max=0;
16 int min=0;
17
18 int main(int argc, char *argv[]) {
19     pthread_t tid[TNUMBERS];
20
21     for(int i=1; i<argc; i++){numbers[i-1]=atoi(argv[i]);}
22
23     printf("Running Threads...\n");
24     pthread_create(&tid[0],NULL,findAverage,&argc);
25     pthread_create(&tid[1],NULL,findMax,&argc);
26     pthread_create(&tid[2],NULL,findMin,&argc);
27
28     for(int i=0;i<TNUMBERS;i++){pthread_join(tid[i],NULL);}
29
30     printf("Threading is done, threads returned!\n");
31 }
```

Open ▾

various.c  
~/Desktop/Lab 4 OS

```
30 printf("Threading is done, threads returned!\n");
31
32 printf("The average value: %.2f\n", average);
33 printf("The max value: %d\n", max);
34 printf("The min value: %d\n", min);
35 }
36
37 void *findAverage(void *param){
38     int n=*(int*)param-1;
39     int sum=0;
40
41     printf("Computing average value...\n");
42
43     for(int i=0; i<n;i++){
44         sum+=numbers[i];
45     }
46     average=(float)sum/n;
47     pthread_exit(0);}
48
49 void *findMax(void *param){
50     int n=*(int*)param-1;
51     int c, index=0;
52
53     printf("Computing max value...\n");
54
55     for(c =1; c<n; c++){
56         if(numbers[c]>numbers[index]){index=c;}
57     }
58     max=numbers[index];
59     pthread_exit(0);}
60 void *findMin(void *param){
```

```
60 void *findMin(void *param){
61     int n=*(int*)param-1;
62     int c, index=0;
63
64     printf("Computing max value...\n");
65
66     for(c =1; c<n; c++){
67         if(numbers[c]<numbers[index]){index=c;}
68     }
69     min=numbers[index];
70     pthread_exit(0);}
```

❖ The result:

```
dikshya@dikshya-VirtualBox: ~/Desktop/Lab 4 OS
dikshya@dikshya-VirtualBox:~/Desktop/Lab 4 OS$ touch various.c
dikshya@dikshya-VirtualBox:~/Desktop/Lab 4 OS$ gcc various.c -o various -lpthread
dikshya@dikshya-VirtualBox:~/Desktop/Lab 4 OS$ ./various 90 81 78 95 79 72 85
Running Threads...
Computing max value...
Computing max value...
Computing average value...
Threading is done, threads returned!
The average value: 82.86
The max value: 95
The min value: 72
dikshya@dikshya-VirtualBox:~/Desktop/Lab 4 OS$
```

The multithreaded program is passed a series of numbers on the command line and will then create three separate worker threads. One thread determines the average of the numbers, the second determines the maximum value, and the third determines the minimum value.

### Experiment 3: calculate the sum of number of squares

❖ The code:

```
sumserial.c
~/Desktop/Lab 4 OS

1 #include<stdio.h>
2 #include<stdlib.h>
3 #include<sys/time.h>
4
5 int main (int argc, char *argv[]){
6     struct timeval start, end;
7     int n = atoi(argv[1]);
8     int sum = 0;
9     gettimeofday(&start, NULL);
10    for(int i = 1; i<n; i++)
11    {
12        sum += (i * i);
13    }
14    gettimeofday(&end, NULL);
15    printf("The sum of squares in [1,%d]:%d |Time taken: %ld micro seconds\n", n, sum,
16        ((end.tv_sec * 1000000 + end.tv_usec) - (start.tv_sec * 1000000 + start.tv_usec)));
17    return 0;
18 }
```

❖ The result:


I use the range [1,10000) which sums the squares in integers from 1 to 9999. I take time stamp before and after the look execution. The results are as follow:

```
dikshya@dikshya-VirtualBox: ~/Desktop/Lab 4 OS
dikshya@dikshya-VirtualBox:~/Desktop/Lab 4 OS$ gcc sumserial.c -o sumserial -lpthread
dikshya@dikshya-VirtualBox:~/Desktop/Lab 4 OS$ ./sumserial 100000
The sum of squares in [1,100000):216474736 |Time taken: 350 micro seconds
dikshya@dikshya-VirtualBox:~/Desktop/Lab 4 OS$
```

```
dikshya@dikshya-VirtualBox: ~/Desktop/Lab 4 OS
dikshya@dikshya-VirtualBox:~/Desktop/Lab 4 OS$ gcc sumserial.c -o sumserial -lpthread
dikshya@dikshya-VirtualBox:~/Desktop/Lab 4 OS$ ./sumserial 100000
The sum of squares in [1,100000):216474736 |Time taken: 350 micro seconds
dikshya@dikshya-VirtualBox:~/Desktop/Lab 4 OS$ ./sumserial 100
The sum of squares in [1,100):328350 |Time taken: 1 micro seconds
dikshya@dikshya-VirtualBox:~/Desktop/Lab 4 OS$
```


#### Experiment 4: calculate the sum of number of squares using Pthread

❖ The code:

Open ▾ 

sum\_pthread.c  
~/Desktop/Lab 4 OS

```
1 #include<stdio.h>
2 #include<pthread.h>
3 #include<stdlib.h>
4 #include<sys/time.h>
5 #include<sys/types.h>
6
7 void *Sum(void *param);
8
9 struct thread_args{
10 int tid;
11 int a;
12 int b;
13 long int result;
14 };
15
16 int main (int argc, char *argv[]){
17 struct timeval start, end;
18 gettimeofday(&start, NULL);
19
20 int numthreads;
21 int number;
22 double totalSum=0;
23
24 numthreads = atoi(argv[1]);
25 number = atoi(argv[2]);
26
27 pthread_t tid[numthreads];
28 struct thread_args targs[numthreads];
29 printf("I am process in Interval [%d,%d]\n",1,number);
30 printf("Running threads...\n\n");
31
```

Open ▾ 

sum\_pthread.c  
~/Desktop/Lab 4 OS

```
31
32
33 for(int i=0; i<numthreads; i++){
34 targs[i].tid=i;
35 targs[i].a=(number)*(targs[i].tid)/(numthreads);
36 targs[i].b=(number)*(targs[i].tid+1)/(numthreads);
37
38 //For last thread
39 if(i==numthreads-1){targs[i].b=number;}
40
41 pthread_create(&tid[i],NULL,Sum,&targs[i]);
42 }
43
44 for(int i=0; i<numthreads;i++){
45 pthread_join(tid[i],NULL);}
46
47 printf("Threads exited!\n");
48 printf("Process collecting information...\n");
49
50 for(int i=0; i<numthreads;i++){
51 totalSum+=targs[i].result;}
52
53 gettimeofday(&end, NULL);
54 printf("The sum of squares %.2f\nTime taken: %ld in micro seconds\n",totalSum,((end.tv_sec*1000000+end.tv_usec)-
   (start.tv_sec*1000000+start.tv_usec)));
55
```

```

55
56 return 0;
57 }
58
59
60 void *Sum(void *param){
61 int start = (*((struct thread_args*) param)).a;
62 int end = (*((struct thread_args*) param)).b;
63 int id = (*((struct thread_args*) param)).tid;
64 long int sum=0;
65
66 printf("I am thread %d in Interval [%d,%d]\n",id,start,end);
67
68
69 for(int i=1;i<end;i++){sum+=(i*i);}
70
71 (*((struct thread_args*) param)).result=sum;
72 printf("I am thread %d\nThe sum: %ld\n\n",id,(*((struct thread_args*) param)).result);
73 pthread_exit(0);
74 }

```

### ❖ The Result:

```

dikshya@dikshya-VirtualBox: ~/Desktop/Lab 4 OS
dikshya@dikshya-VirtualBox:~/Desktop/Lab 4 OS$ touch sum_pthread.c
dikshya@dikshya-VirtualBox:~/Desktop/Lab 4 OS$ gcc sum_pthread.c -o sum_pthread
dikshya@dikshya-VirtualBox:~/Desktop/Lab 4 OS$ ./sum_pthread 5 1000
I am process in Interval [1,1000)
Running threads....

I am thread 2 in Interval [400,600)
I am thread 2
The sum: 71820100

I am thread 1 in Interval [200,400)
I am thread 1
The sum: 21253400

I am thread 0 in Interval [0,200)
I am thread 0
The sum: 2646700

I am thread 3 in Interval [600,800)
I am thread 3
The sum: 170346800

I am thread 4 in Interval [800,1000)
I am thread 4
The sum: 332833500

Threads exited!
Process collecting information...
The sum of squares 598900500.00
Time taken: 901 in micro seconds
dikshya@dikshya-VirtualBox:~/Desktop/Lab 4 OS$

```



Comparing with the serial implementation:

```
dikshya@dikshya-VirtualBox: ~/Desktop/Lab 4 OS
dikshya@dikshya-VirtualBox:~/Desktop/Lab 4 OS$ gcc sumserial.c -o senserial -lpthread
dikshya@dikshya-VirtualBox:~/Desktop/Lab 4 OS$ ./sumserial 100
The sum of squares in [1,100):328350 |Time taken: 0 micro seconds
dikshya@dikshya-VirtualBox:~/Desktop/Lab 4 OS$
```

## Experiment 5: analysis the performance

❖ The code:

```
summation.c
27
28 if(argc < 3 ){
29 printf("Usage: ./sum_threads <numthreads> <number> ");
30 return 1;
31 } else{
32 numthreads = atoi(argv[1]);
33 number = atoi(argv[2]);
34 }
35
36 pthread_t  tid[numthreads];
37 struct thread_args targs[numthreads];
38 printf("I am Process | range: [%d,%d]\n",1,number);
39 printf("Running Threads...\n\n");
40
41
42
43 for(int i=0; i<numthreads;i++ ){
44 //Setting up the args
45 targs[i].tid = i;
46 targs[i].a = (number)*(targs[i].tid)/(numthreads);
47 targs[i].b = (number)*(targs[i].tid+1)/(numthreads);
48
49 //Special for last thread
50 if(i == numthreads-1 ){
51 targs[i].b = number;
52 }
53
54 pthread_create(&tid[i],NULL,Sum, &targs[i]);
```

```
27
28 if(argc < 3 ){
29 printf("Usage: ./sum_threads <numthreads> <number> ");
30 return 1;
31 } else{
32 numthreads = atoi(argv[1]);
33 number = atoi(argv[2]);
34 }
35
36 pthread_t  tid[numthreads];
37 struct thread_args targs[numthreads];
38 printf("I am Process | range: [%d,%d]\n",1,number);
39 printf("Running Threads...\n\n");
40
41
42
43 for(int i=0; i<numthreads;i++ ){
44 //Setting up the args
45 targs[i].tid = i;
46 targs[i].a = (number)*(targs[i].tid)/(numthreads);
47 targs[i].b = (number)*(targs[i].tid+1)/(numthreads);
48
49 //Special for last thread
50 if(i == numthreads-1 ){
51 targs[i].b = number;
52 }
53
54 pthread_create(&tid[i],NULL,Sum, &targs[i]);
```

❖ The results:

```

dikshya@dikshya-VirtualBox:~/Desktop/Lab 4 OS$ ./summation 5 20
I am Process | range: [1,20)
Running Threads...

I am thread 0 | range: [0,4)
I am thread 0 | Sum: 14.00 | Time Take: 9

I am thread 2 | range: [8,12)
I am thread 2 | Sum: 366.00 | Time Take: 5

I am thread 1 | range: [4,8)
I am thread 1 | Sum: 126.00 | Time Take: 2

I am thread 3 | range: [12,16)
I am thread 3 | Sum: 734.00 | Time Take: 4

I am thread 4 | range: [16,20)
I am thread 4 | Sum: 1230.00 | Time Take: 8

Threads Exited!
Process collecting information...
Total Sum is: 2470.00 | Taken Time: 733 mirco seconds
dikshya@dikshya-VirtualBox:~/Desktop/Lab 4 OS$

```

#### ❖ Questions:

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

#### Answer:

The running time is longer whenever the number of threads increases. Since the division of the superset into subsets take a long time, hence I believe that multithreading is not a good option for the sum of squares, multithreading could be more efficient matrix multiplication and such.

- ❖ 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.

#### Answer: Ok