

# Lab 4 Task 1 + Task 2

**Name: Muhammad Sherjeel Akhtar**

**Roll No: 20p-0101**

**Subject: Operating Systems Lab**

**Submitted To Respected Sir: Muhammad Ahsan**

**Section: BCS-4A**

**#####786#####**

## Task 1:

```
#include <pthread.h>
#include <stdio.h>
#define ARRAY_SIZE 5
struct thread_data{
    int *array1;
    int *array2;
    int *result;
    int index;
};
void *add_element(void *threadArg){
    struct thread_data *my_data;
    my_data=(struct thread_data *) threadArg;
    int index=my_data->index;
    my_data->result[index]=my_data->array1[index]+my_data->array2[index];
    pthread_exit(NULL);
}
```

```

}
int main() {
    pthread_t threads[ARRAY_SIZE];
    int array1[ARRAY_SIZE]={1,2,3,4,5};
    int array2[ARRAY_SIZE]={6,7,8,9,10};
    int result[ARRAY_SIZE];
    struct thread_data thread_data_array[ARRAY_SIZE];
    printf("Array 1: ");
    for (int i=0;i<ARRAY_SIZE;i++){
        printf("%d ",array1[i]);
    }
    printf("\n");
    printf("Array 2: ");
    for (int i=0;i<ARRAY_SIZE;i++){
        printf("%d ",array2[i]);
    }
    printf("\n");
    for (int i=0;i<ARRAY_SIZE;i++){
        thread_data_array[i].array1=array1;
        thread_data_array[i].array2=array2;
        thread_data_array[i].result=result;
        thread_data_array[i].index=i;
        pthread_create(&threads[i],NULL,add_element,(void *) &thread_data_array[i]);
    }
    for (int i=0;i<ARRAY_SIZE;i++){
        pthread_join(threads[i],NULL);
    }
    printf("Result:");
    for (int i=0;i<ARRAY_SIZE;i++){
        printf("%d ",result[i]);
    }
    printf("\n");
    printf("Exiting\n");
    return 0;
}

```

**Visually:**

```

C task_1.c > main()
1  #include <pthread.h>
2  #include <stdio.h>
3  #define ARRAY_SIZE 5
4  struct thread_data{
5      int *array1;
6      int *array2;
7      int *result;
8      int index;
9  };
10 void *add_element(void *threadArg){
11     struct thread_data *my_data;
12     my_data=(struct thread_data *) threadArg;
13     int index=my_data->index;
14     my_data->result[index]=my_data->array1[index]+my_data->array2[index];
15     pthread_exit(NULL);
16 }

```

```

17 int main(){
18     pthread_t threads[ARRAY_SIZE];
19     int array1[ARRAY_SIZE]={1,2,3,4,5};
20     int array2[ARRAY_SIZE]={6,7,8,9,10};
21     int result[ARRAY_SIZE];
22     struct thread_data thread_data_array[ARRAY_SIZE];
23     printf("Array 1: ");
24     for (int i=0;i<ARRAY_SIZE;i++){
25         printf("%d ",array1[i]);
26     }
27     printf("\n");
28     printf("Array 2: ");
29     for (int i=0;i<ARRAY_SIZE;i++){
30         printf("%d ",array2[i]);
31     }
32     printf("\n");
33     for (int i=0;i<ARRAY_SIZE;i++){
34         thread_data_array[i].array1=array1;
35         thread_data_array[i].array2=array2;
36         thread_data_array[i].result=result;
37         thread_data_array[i].index=i;
38         pthread_create(&threads[i],NULL,add_element,(void *) &thread_data_array[i]);
39     }

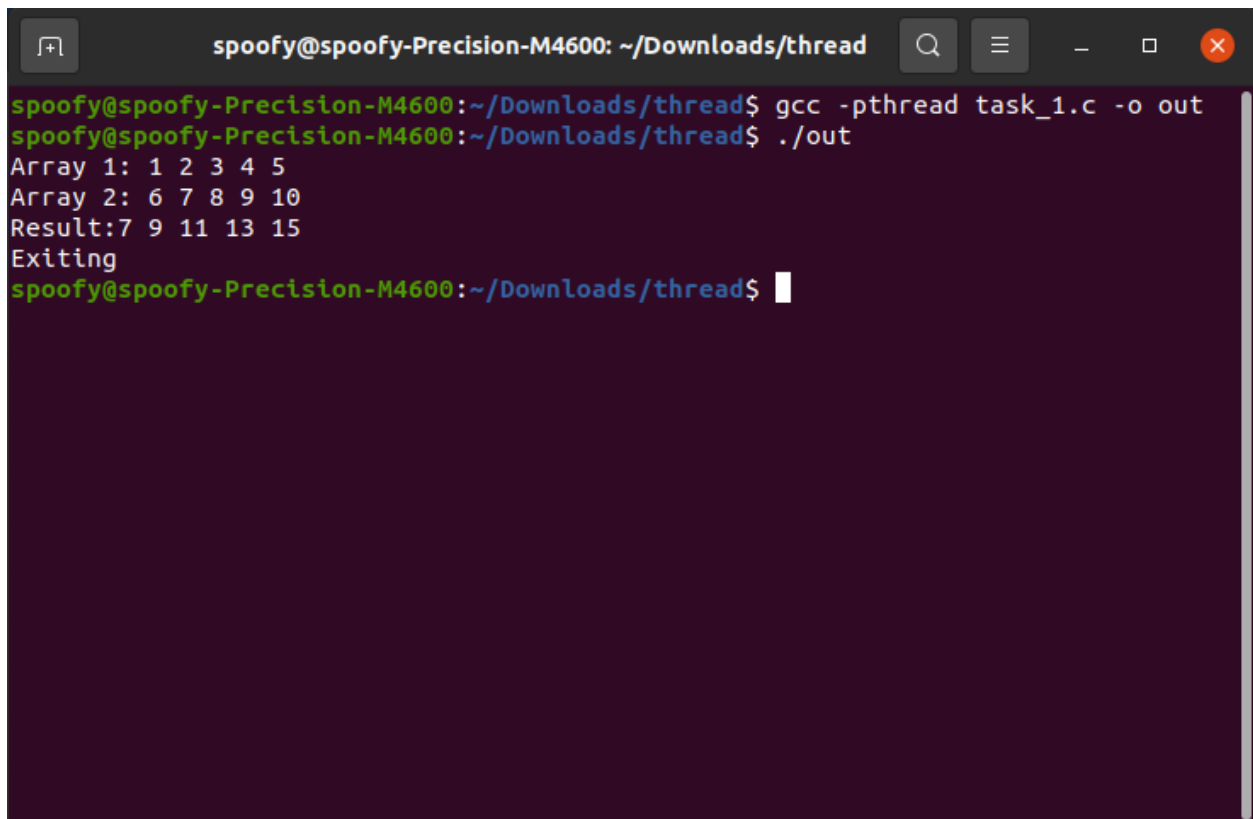
```

```

printf("\n");
for (int i=0;i<ARRAY_SIZE;i++){
    thread_data_array[i].array1=array1;
    thread_data_array[i].array2=array2;
    thread_data_array[i].result=result;
    thread_data_array[i].index=i;
    pthread_create(&threads[i],NULL,add_element,(void *) &thread_data_array[i]);
}
for (int i=0;i<ARRAY_SIZE;i++){
    pthread_join(threads[i],NULL);
}
printf("Result:");
for (int i=0;i<ARRAY_SIZE;i++){
    printf("%d ",result[i]);
}
printf("\n");
printf("Exiting\n");
return 0;
}

```

## Running:



```

spoofy@spoofy-Precision-M4600: ~/Downloads/thread
spoofy@spoofy-Precision-M4600:~/Downloads/thread$ gcc -pthread task_1.c -o out
spoofy@spoofy-Precision-M4600:~/Downloads/thread$ ./out
Array 1: 1 2 3 4 5
Array 2: 6 7 8 9 10
Result:7 9 11 13 15
Exiting
spoofy@spoofy-Precision-M4600:~/Downloads/thread$

```

## Task 2:

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
struct thread_data {
    int width;
    int height;
    int area;
    int perimeter;
};
void *compute_area(void *arg) {
    struct thread_data *data=(struct thread_data*) arg;
    data->area=data->width*data->height;
    printf("Thread %ld computed the area:%d\n",pthread_self(),data->area);
    pthread_exit(NULL);
}
void *compute_perimeter(void *arg) {
    struct thread_data *data=(struct thread_data*) arg;
    data->perimeter=2*(data->width+data->height);
    printf("Thread %ld computed the perimeter:%d\n",pthread_self(),data->perimeter);
    pthread_exit(NULL);
}
int main() {
    pthread_t threads[2];
    struct thread_data data={0,0,0,0};
    printf("Enter the width and height of the rectangle:");
    scanf("%d %d",&data.width,&data.height);
    pthread_create(&threads[0],NULL,compute_area,(void*)&data);
    pthread_create(&threads[1],NULL,compute_perimeter,(void*)&data);
    pthread_join(threads[0],NULL);
    pthread_join(threads[1],NULL);
    printf("The area of the rectangle is:%d\n",data.area);
    printf("The perimeter of the rectangle is:%d\n",data.perimeter);
    return 0;
}
```

## Visually:

```

C task_2.c > main()
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <pthread.h>
4  struct thread_data {
5      int width;
6      int height;
7      int area;
8      int perimeter;
9  };
10 void *compute_area(void *arg) {
11     struct thread_data *data=(struct thread_data*) arg;
12     data->area=data->width*data->height;
13     printf("Thread %ld computed the area:%d\n",pthread_self(),data->area);
14     pthread_exit(NULL);
15 }
16 void *compute_perimeter(void *arg) {
17     struct thread_data *data=(struct thread_data*) arg;
18     data->perimeter=2*(data->width+data->height);
19     printf("Thread %ld computed the perimeter:%d\n",pthread_self(),data->perimeter);
20     pthread_exit(NULL);

```

```

22 int main() {
23     pthread_t threads[2];
24     struct thread_data data={0,0,0,0};
25     printf("Enter the width and height of the rectangle:");
26     scanf("%d %d",&data.width,&data.height);
27     pthread_create(&threads[0],NULL,compute_area,(void*)&data);
28     pthread_create(&threads[1],NULL,compute_perimeter,(void*)&data);
29     pthread_join(threads[0],NULL);
30     pthread_join(threads[1],NULL);
31     printf("The area of the rectangle is:%d\n",data.area);
32     printf("The perimeter of the rectangle is:%d\n",data.perimeter);
33     return 0;
34 }

```

## Running:

```

spoofy@spoofy-Precision-M4600:~/Downloads/thread$ gcc -pthread task_2.c -o out1
spoofy@spoofy-Precision-M4600:~/Downloads/thread$ ./out1
Enter the width and height of the rectangle:10
10
Thread 140411212662528 computed the area:100
Thread 140411204269824 computed the perimeter:40
The area of the rectangle is:100
The perimeter of the rectangle is:40
spoofy@spoofy-Precision-M4600:~/Downloads/thread$

```

***FIN.***