Assignment#5
Saad Ahmad
(20P-0051)

Code:

task1.c

Os > task1.c > ...

```c
#include<unistd.h>
#include<sys/types.h>
#include<errno.h>
#include<stdio.h>
#include<stdlib.h>
#include<pthread.h>
#include<string.h>
#include<semaphore.h>

#define NUM_RUNS 10000000
sem_t mutex;
void handler(void *ptr);
int counter;

int main()
{
    int i[2];
    pthread_t thread_a;
    pthread_t thread_b;
    sem_init(&mutex, 0, 1);

    i[0] = 0;
    i[1] = 1;

    pthread_create(&thread_a , NULL , (void *) &handler, (void *) &i[0]);
    pthread_create(&thread_b , NULL , (void *) &handler, (void *) &i[1]);

    pthread_join(thread_a , NULL);
    pthread_join(thread_b , NULL);
    sem_destroy(&mutex);

    printf("-----------------------------------------------------------------\n");

    printf("Final counter value:     %d\n" , counter);
    printf("Error:                   %d\n" , (NUM_RUNS*2 - counter));

    exit(0);
}
```

```c

void handler(void *ptr)
{
    int iter = 0;
    int thread_num;
    thread_num = *((int *) ptr);

    printf("String thread: %d \n ",  thread_num);

    sem_wait(&mutex);               // UP

    while (iter < NUM_RUNS)
    {
        counter++;
        iter += 1;
    }
    sem_post(&mutex);               // Down

    printf("Thread %d, counter = %d \n" , thread_num , counter);
    pthread_exit(0);

}


```

Output:

```
ayyzenn@Saad-A:~/Desktop/Os$ gcc task1.c -lpthread
ayyzenn@Saad-A:~/Desktop/Os$ ./a.out
String thread: 0
 String thread: 1
 Thread 0, counter = 10000000
Thread 1, counter = 20000000
------------------------------------------------------------
Final counter value:     20000000
Error:                   0
ayyzenn@Saad-A:~/Desktop/Os$
```

Explanation:

When thread 1 enter the critical region then the value of mutex is set 0 and if there is pre-emption then the thread 2 in put to sleep. And until the thread 1 has completed his process the thread 2 will remain in sleep state.