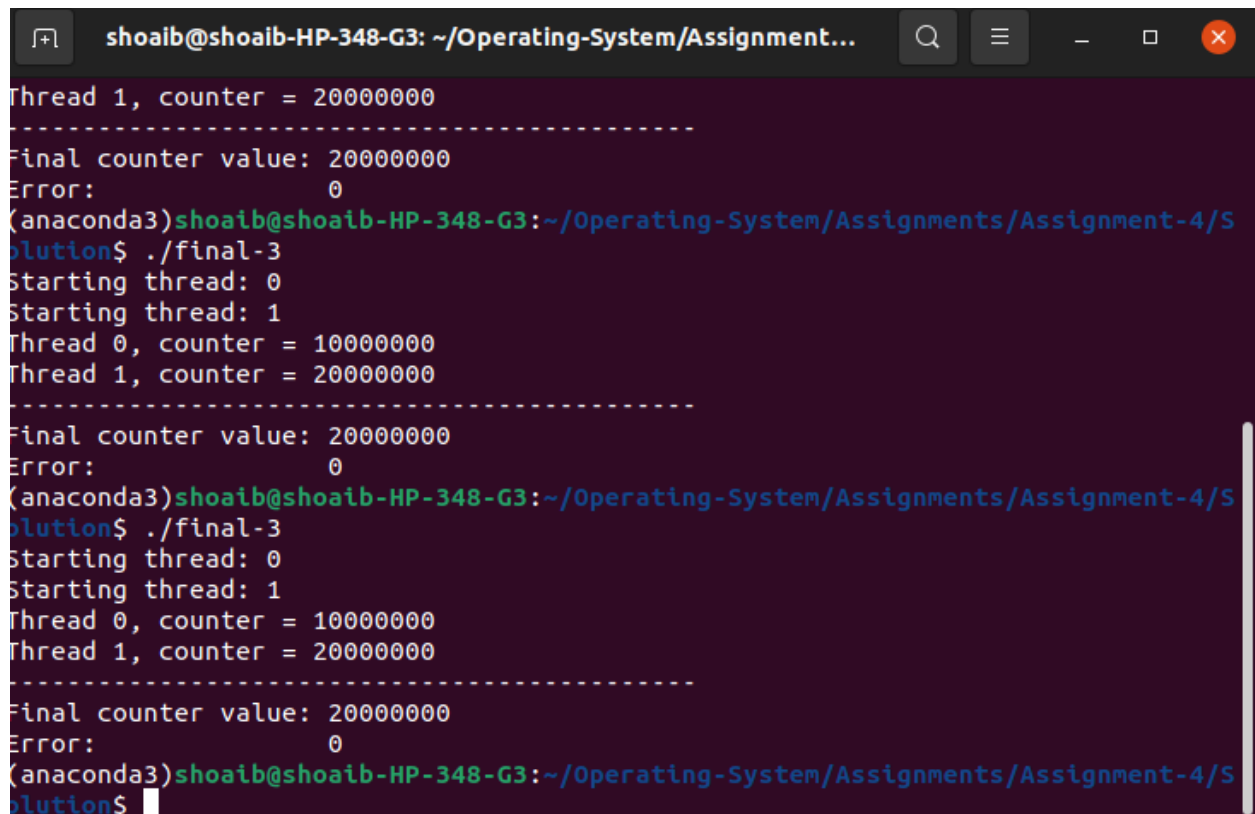


SHOAIB AKHTAR
20P-0147
BS-CS (4B)

Assignment-5

We have used the semaphores in this assignment to solve the error...

A terminal window with a dark purple background and white text. The window title is 'shoaib@shoaib-HP-348-G3: ~/Operating-System/Assignment...'. The terminal shows the output of a program that uses semaphores. It displays thread creation, counter updates, and final values. The output is repeated three times, showing consistent results. The program appears to be 'final-3' as indicated by the command './final-3'. The output shows 'Thread 1, counter = 20000000', 'Final counter value: 20000000', and 'Error: 0'. The prompt '(anaconda3)shoaib@shoaib-HP-348-G3:~/Operating-System/Assignments/Assignment-4/Solution\$' is visible at the end of each execution block.

```
shoaib@shoaib-HP-348-G3: ~/Operating-System/Assignment...
Thread 1, counter = 20000000
-----
Final counter value: 20000000
Error: 0
(anaconda3)shoaib@shoaib-HP-348-G3:~/Operating-System/Assignments/Assignment-4/S
olution$ ./final-3
Starting thread: 0
Starting thread: 1
Thread 0, counter = 10000000
Thread 1, counter = 20000000
-----
Final counter value: 20000000
Error: 0
(anaconda3)shoaib@shoaib-HP-348-G3:~/Operating-System/Assignments/Assignment-4/S
olution$ ./final-3
Starting thread: 0
Starting thread: 1
Thread 0, counter = 10000000
Thread 1, counter = 20000000
-----
Final counter value: 20000000
Error: 0
(anaconda3)shoaib@shoaib-HP-348-G3:~/Operating-System/Assignments/Assignment-4/S
olution$
```

Picture of code

```
int counter;
sem_t mutex;
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);

    printf("-----\n");
    printf("Final counter value: %d\n", counter);
    printf("Error: %d\n", (NUM_RUNS*2-counter));
    sem_destroy(&mutex);
    exit(0);
}

void handler( void *ptr)
{
    int iter = 0;
    int thread_num;
    thread_num = *((int *) ptr);
    printf("Starting thread: %d \n", thread_num);
    sem_wait(&mutex);
    while ( iter < NUM_RUNS)
    {
        counter++;
        iter += 1;
    }
    sem_post(&mutex);
    printf("Thread %d, counter = %d \n", thread_num, counter);
    pthread_exit(0);
}
```

