Operating System

Report
Assignment Simulation Based

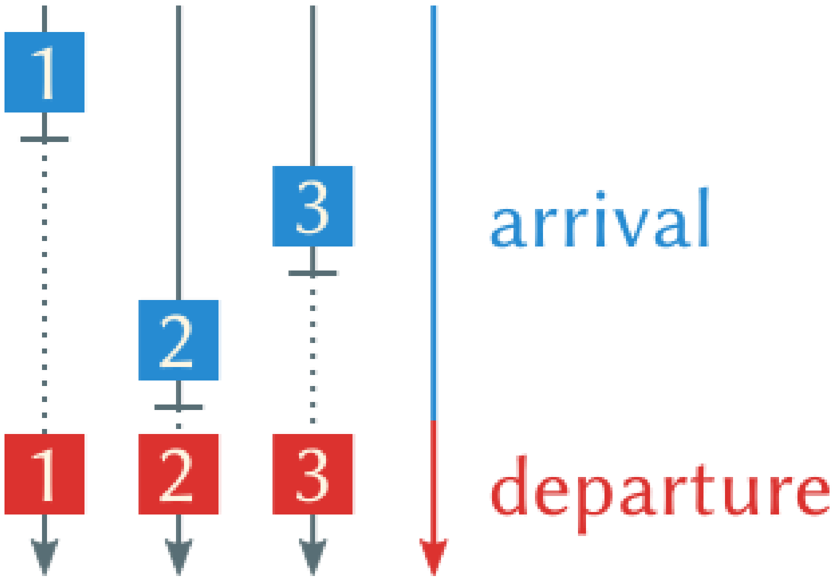
Student Name: K.GUNAVARDHAN
Student ID: 11705326
Section No: EE032
Roll No.: 37
Email Address:vardhankgv@gmail.com
GitHub Link: github.com/Vardhankgv/OS_Project

Question Number: 16

Barriers

A barrier is a type of synchronization method. A barrier for a group of threads or processes in the source code means any thread/process must stop at this point and cannot proceed until all other threads/processes reach this barrier.

A barrier is a method to implement synchronization. Synchronization ensures that concurrently executing threads or processes do not execute specific portions of the program at the same time. When a barrier is inserted at a specific point in a program for a group of threads [processes], any thread [process] must stop at this point and cannot proceed until all other threads [processes] reach this barrier.



Algorithm:

1. initialize barrier_size and thread_count;
2. create threads
3. threads doing some work
4. threads waiting at the barrier.
5. barrier is released when last thread comes at the thread.
6. all threads complete thier task and exit.
7. exit.

Complexity:

O (n) complexity. “n” is no of thread_count.

Compile And Run:

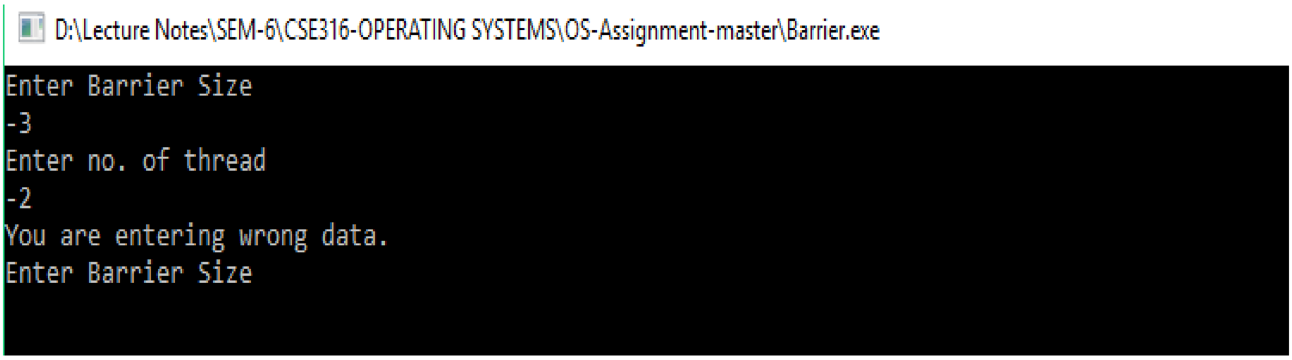
```
#include<stdio.h>
#include<pthread.h>
#include<stdlib.h>
#include <unistd.h>

pthread_mutex_t lock = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t  finish_cond  = PTHREAD_COND_INITIALIZER;
int barrier = 0;
int thread_count;
int barrier_size;
int counter=0;
int invoke_barrier = 0;
int i,j;

/*.
* Initialize barrier with total number of threads.
 */
void barrier_init(int n_threads)
{
   if ( thread_count < barrier_size ) { barrier = thread_count; return; }
   barrier = n_threads;
}

int decrement()
{
   if (barrier == 0) {

      return 0;
   }

   if(pthread_mutex_lock(&lock) != 0)
   {
      perror("Failed to take lock.");
      return -1;
   }
```

```
    barrier--;

    if(pthread_mutex_unlock(&lock) != 0)
    {
        perror("Failed to unlock.");
        return -1;
    }

    return 0;
}

/*
*  wait for other threads to complete.
 */
int wait_barrier()
{
    if(decrement() < 0)
    {
        return -1;
    }

    while (barrier)
    {
        if(pthread_mutex_lock(&lock) != 0)
        {
            perror("\n Error in locking mutex");
            return -1;
        }

        if(pthread_cond_wait(&finish_cond, &lock) != 0)
        {
            perror("\n Error in cond wait.");
            return -1;
        }
    }

    /*
     * last thread will execute this.
     */
    if(0 == barrier)
    {
        if(pthread_mutex_unlock(&lock) != 0)
        {
            perror("\nError in locking mutex");
            return -1;
        }
        if(pthread_cond_signal(&finish_cond) != 0)
        {
            perror("\n Error while signaling.");
            return -1;
        }
    }

    return 0;
}
```

```
void * barrier_point(void *numthreads)
{

     int r = rand() % 5;

     printf("\nThread %d \nPerforming init task of length %d sec\n",++counter,r);
     sleep(r);

     wait_barrier();
     if (barrier_size!=0) {
       if ((thread_count - (invoke_barrier++) ) % barrier_size == 0) {
         printf("\nBarrier is Released\n");
       }
       printf("\nI am task after barrier\n");
     }
 return  NULL;
}

int main()
{int i,j;

    printf("Enter Barrier Size\n");
    scanf("%d", &barrier_size);

    printf("Enter no. of thread\n");
    scanf("%d", &thread_count);

      //Checking valid input
if (barrier_size>=0 && thread_count>=0) {
        pthread_t tid[thread_count];

        barrier_init(barrier_size);

        for(i =0; i < thread_count; i++)
        {
            pthread_create(&(tid[i]), NULL, &barrier_point, &thread_count);
        }
 for( j = 0; j < thread_count; j++)
        {
            pthread_join(tid[j], NULL);
        }
    }
    //when user give wrong input then this section will execute.
    else{
      printf("You are entering wrong data.\n");
      main();
    }

    return 0;
}
```

Test Cases :

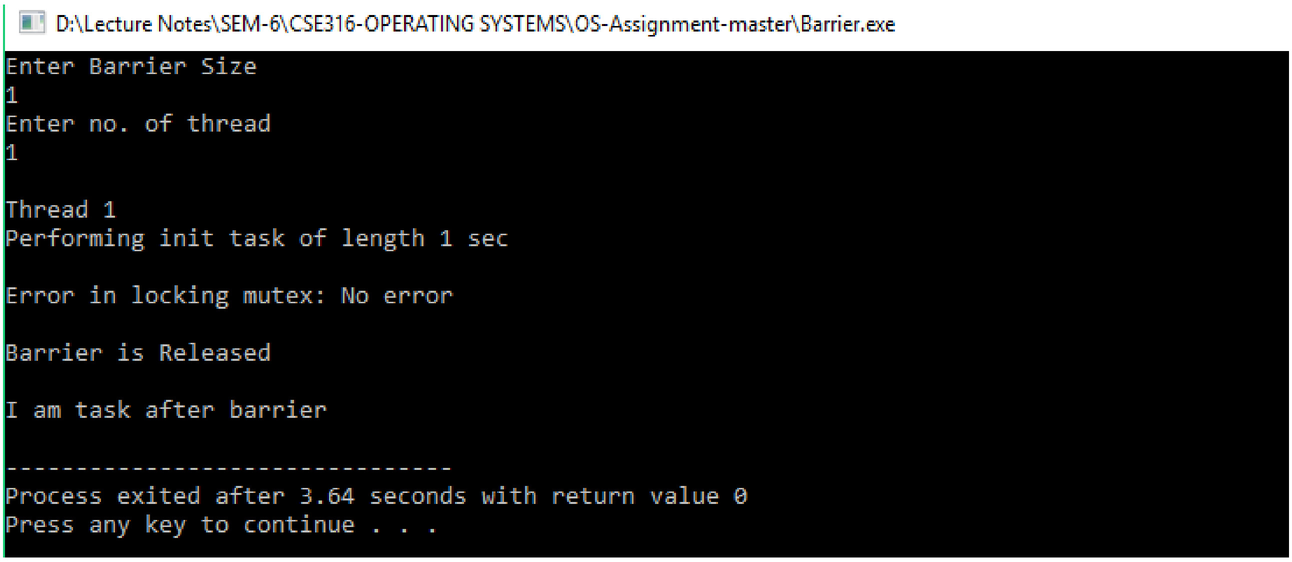
Case 1: when user enter invalid input like – string, double, float, negative no. etc.



D:\Lecture Notes\SEM-6\CSE316-OPERATING SYSTEMS\OS-Assignment-master\Barrier.exe

```
Enter Barrier Size
-3
Enter no. of thread
-2
You are entering wrong data.
Enter Barrier Size
```

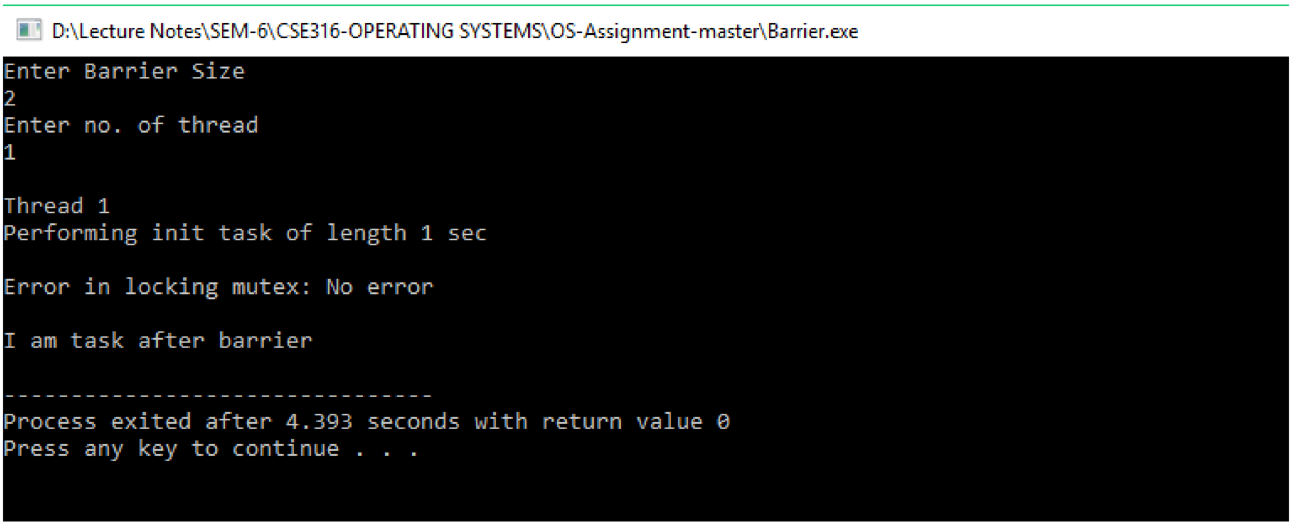
Case 2: when no. of thread equal to size of barrier.



D:\Lecture Notes\SEM-6\CSE316-OPERATING SYSTEMS\OS-Assignment-master\Barrier.exe

```
Enter Barrier Size
1
Enter no. of thread
1

Thread 1
Performing init task of length 1 sec

Error in locking mutex: No error

Barrier is Released

I am task after barrier

--------------------------------
Process exited after 3.64 seconds with return value 0
Press any key to continue . . .
```

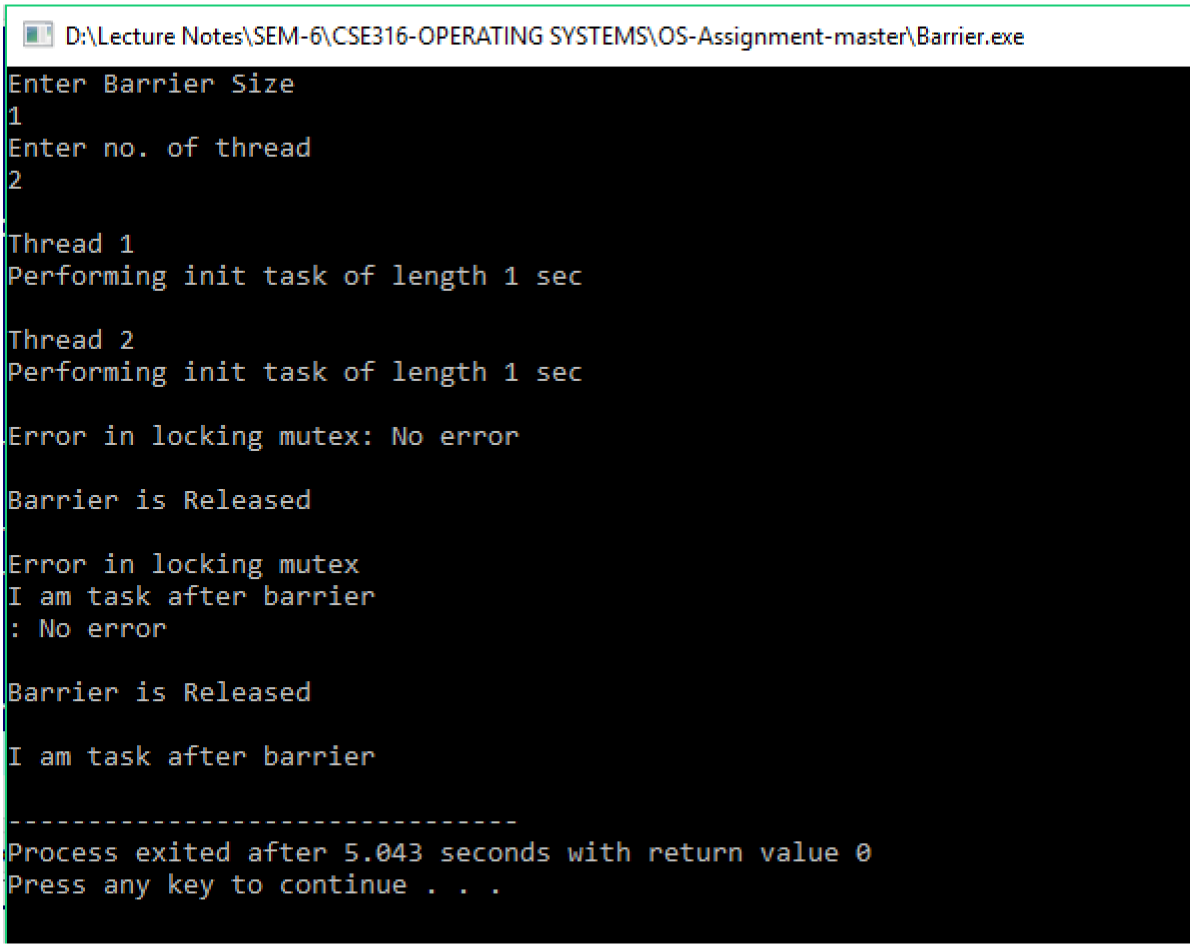
Case 3: when no. of thread is less than size of barrier .



D:\Lecture Notes\SEM-6\CSE316-OPERATING SYSTEMS\OS-Assignment-master\Barrier.exe

```
Enter Barrier Size
2
Enter no. of thread
1

Thread 1
Performing init task of length 1 sec

Error in locking mutex: No error

I am task after barrier

--------------------------------
Process exited after 4.393 seconds with return value 0
Press any key to continue . . .
```

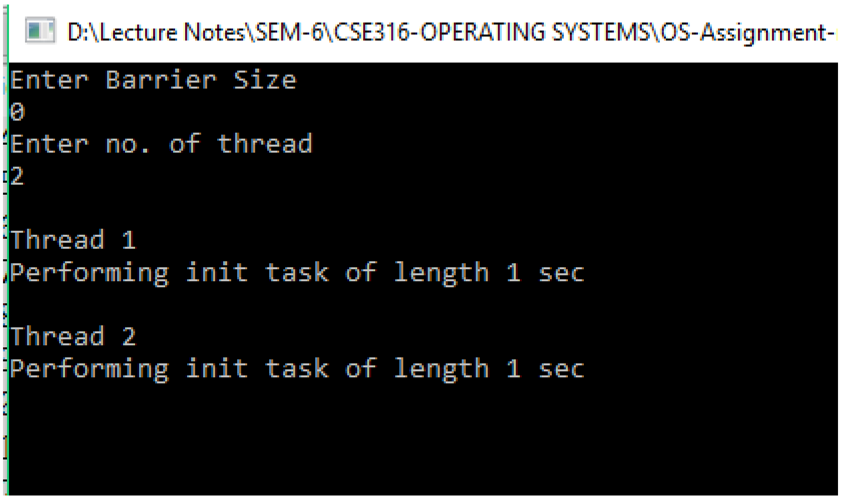
Case 4: when no. of thread is greater than size of Barrier.



D:\Lecture Notes\SEM-6\CSE316-OPERATING SYSTEMS\OS-Assignment-master\Barrier.exe

```
Enter Barrier Size
1
Enter no. of thread
2

Thread 1
Performing init task of length 1 sec

Thread 2
Performing init task of length 1 sec

Error in locking mutex: No error

Barrier is Released

Error in locking mutex
I am task after barrier
: No error

Barrier is Released

I am task after barrier

--------------------------------
Process exited after 5.043 seconds with return value 0
Press any key to continue . . .
```

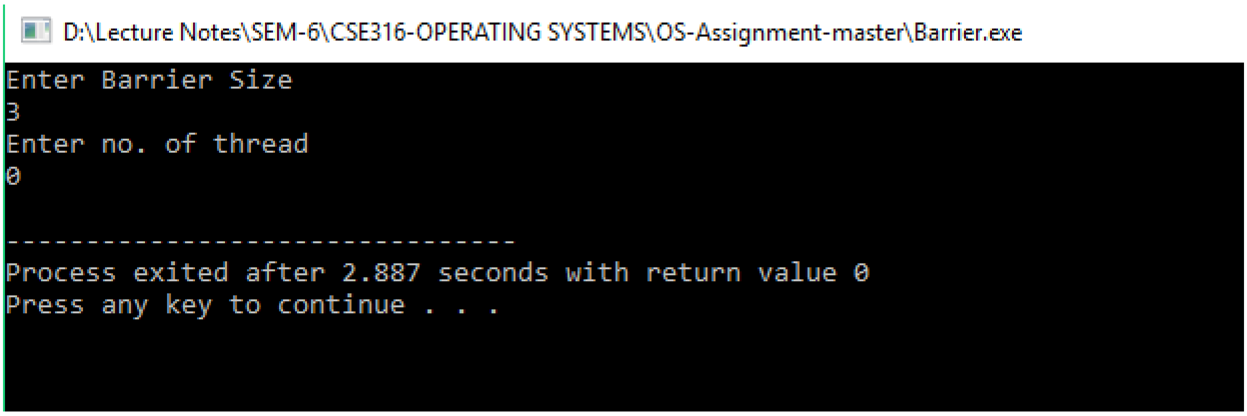
Case 5: when size of Barrier equal to ‘0’.



D:\Lecture Notes\SEM-6\CSE316-OPERATING SYSTEMS\OS-Assignment-

```
Enter Barrier Size
0
Enter no. of thread
2

Thread 1
Performing init task of length 1 sec

Thread 2
Performing init task of length 1 sec
```

Case 6: when thread equal to ‘0’.



D:\Lecture Notes\SEM-6\CSE316-OPERATING SYSTEMS\OS-Assignment-master\Barrier.exe

```
Enter Barrier Size
3
Enter no. of thread
0

--------------------------------
Process exited after 2.887 seconds with return value 0
Press any key to continue . . .
```

GitHub Link: github.com/Vardhankgv/OS_Project.