

Operating System

Report Assignment Simulation Based

16th Question

16. A barrier is a tool for synchronizing the activity of a number of threads. When a thread reaches a barrier point, it cannot proceed until all other threads have reached this point as well. When the last thread reaches the barrier point, all threads are released and can resume concurrent execution. Assume that the barrier is initialized to N —the number of threads that must wait at the barrier point:

```
init(N);
```

Each thread then performs some work until it reaches the barrier point:

```
/* do some work for awhile */barrier point();
```

```
/* do some work for awhile */
```

Using synchronization tools described in this chapter, construct a barrier that implements the following API :

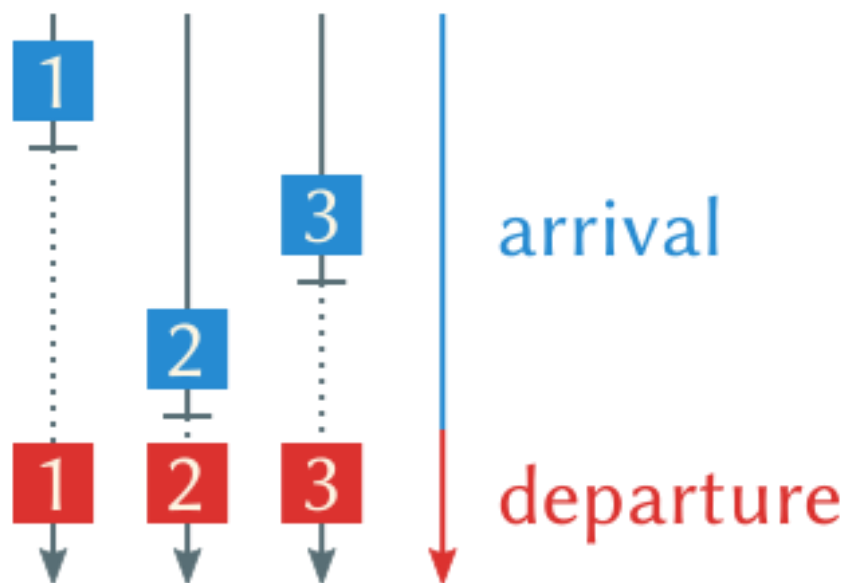
- `int init(int n)` —Initializes the barrier to the specified size.
- `int barrier point(void)` —Identifies the barrier point. All threads are released from the barrier when the last thread reaches this point.

Student Name:	Aditya Kumar
Student ID:	11703629
Section No:	EE029
Roll No.:	B37
Email Address:	er.aksingh110@gmail.com
GitHub Link:	https://github.com/adityasingh110/OS_Assignment_Q16.git

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:

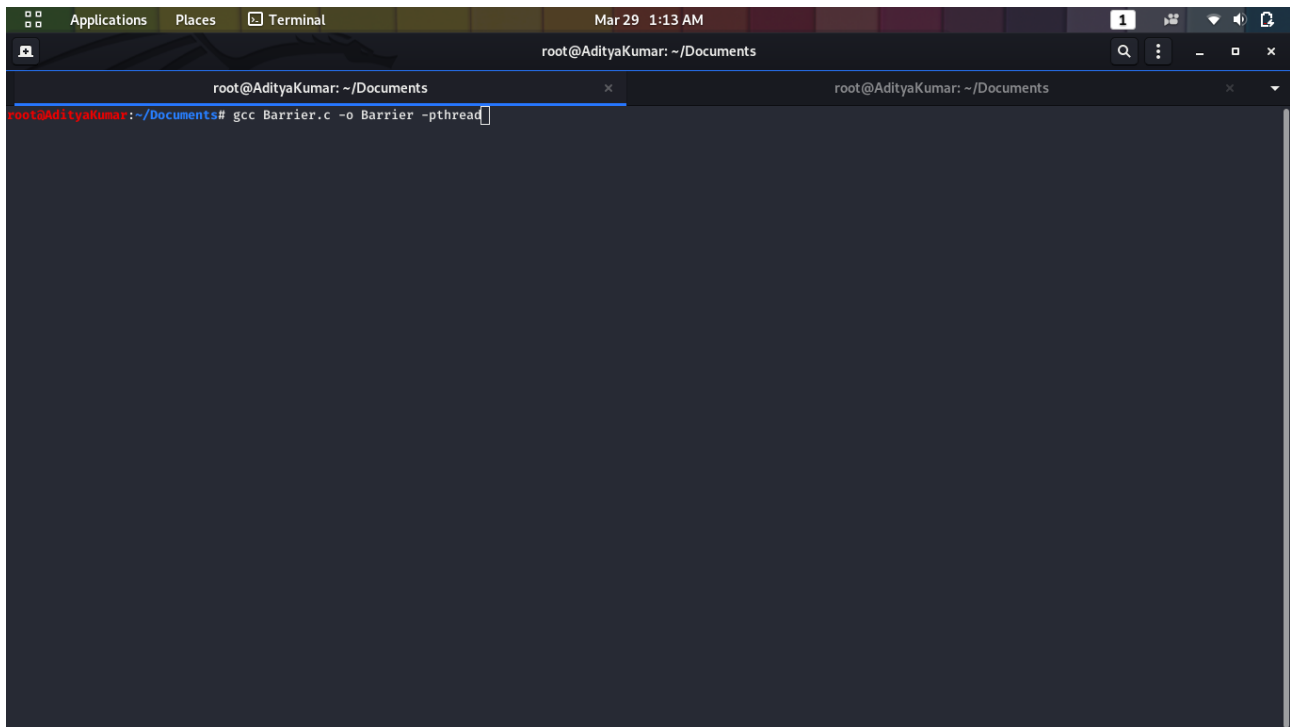
1. Initialize barrier_size and thread_count;
2. Create threads
3. Threads doing some work
4. Threads waiting at the barrier.
5. Barrier is released when last thread comes at the thread.
6. All threads complete their task and exit.
7. Exit.

Complexity:

$O(n)$ complexity. "n" is no of thread_count.

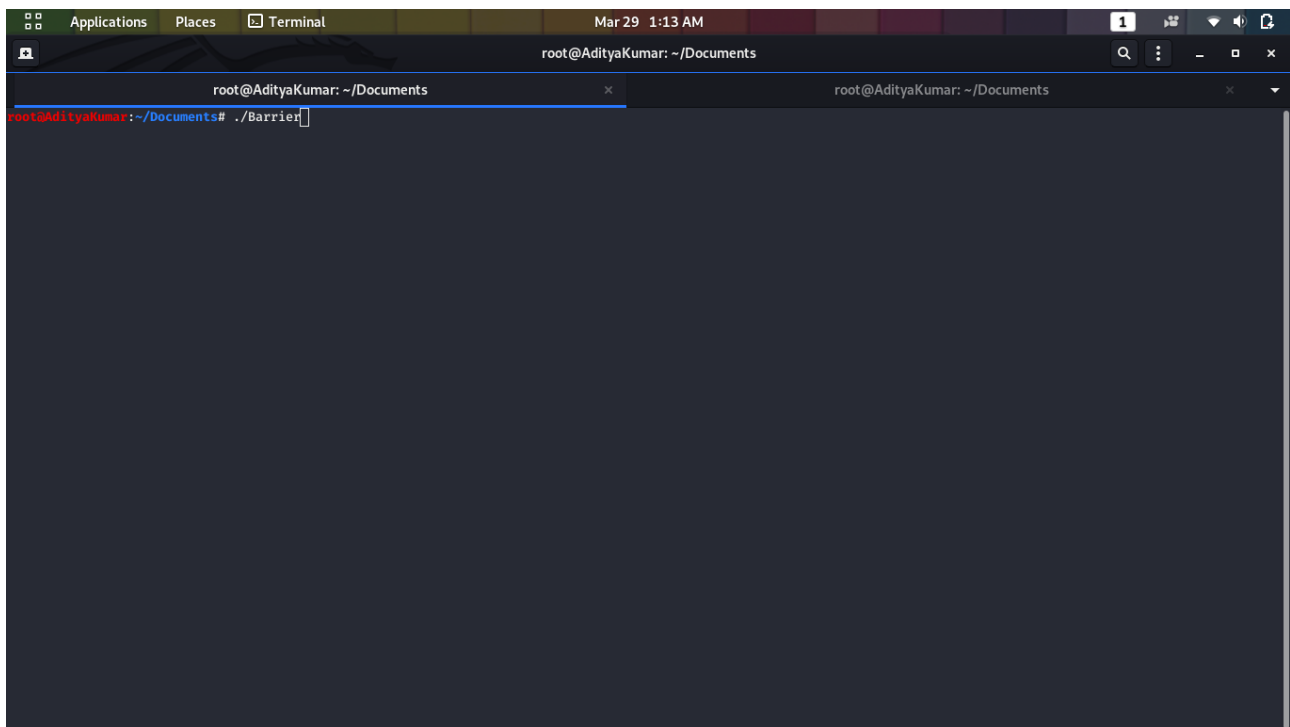
Compile and Run:

Use following command to compile program -
gcc Barrier.c -o Barrier -pthread

A screenshot of a Linux terminal window. The window has a title bar with 'Applications', 'Places', and 'Terminal' tabs. The main title is 'root@AdityaKumar: ~/Documents'. The terminal shows the command 'gcc Barrier.c -o Barrier -pthread' being entered at the prompt 'root@AdityaKumar:~/Documents#'. The cursor is at the end of the command.

```
root@AdityaKumar:~/Documents# gcc Barrier.c -o Barrier -pthread
```

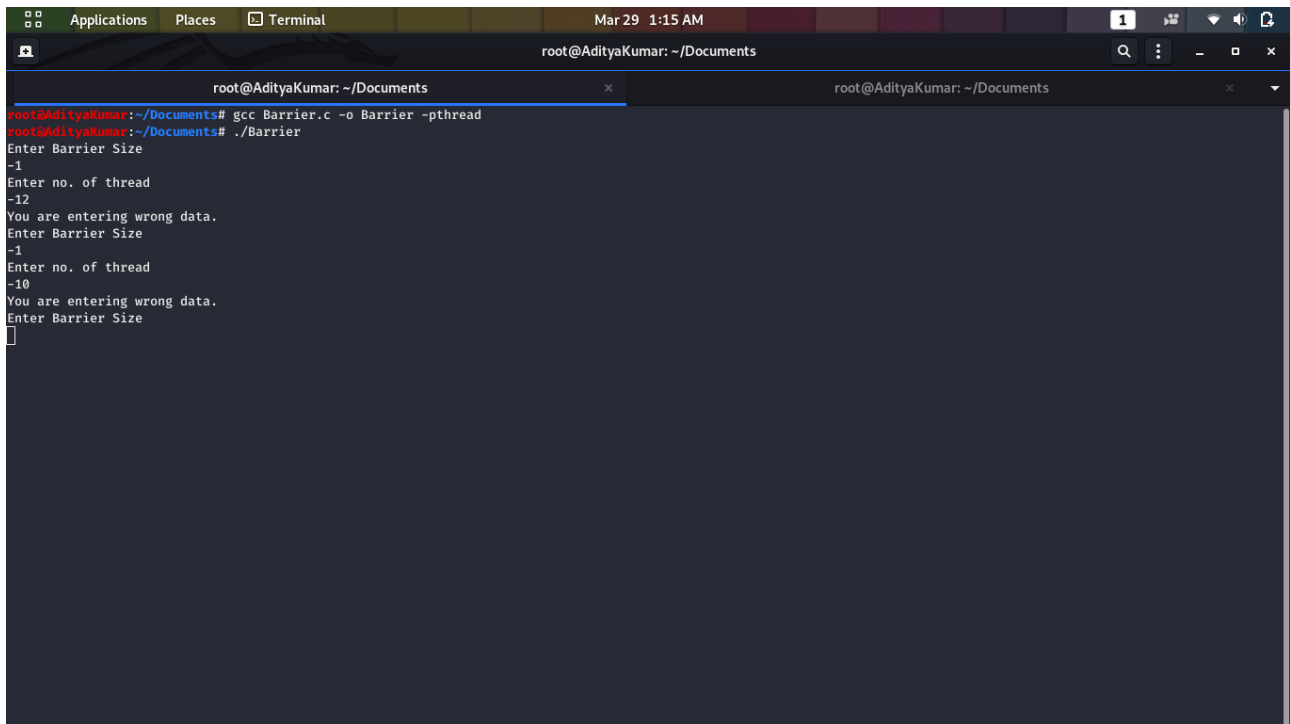
use following command to run program-
./Barrier

A screenshot of a Linux terminal window, similar to the one above. The title bar shows 'Applications', 'Places', and 'Terminal' tabs. The main title is 'root@AdityaKumar: ~/Documents'. The terminal shows the command './Barrier' being entered at the prompt 'root@AdityaKumar:~/Documents#'. The cursor is at the end of the command.

```
root@AdityaKumar:~/Documents# ./Barrier
```

Test Cases:-

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

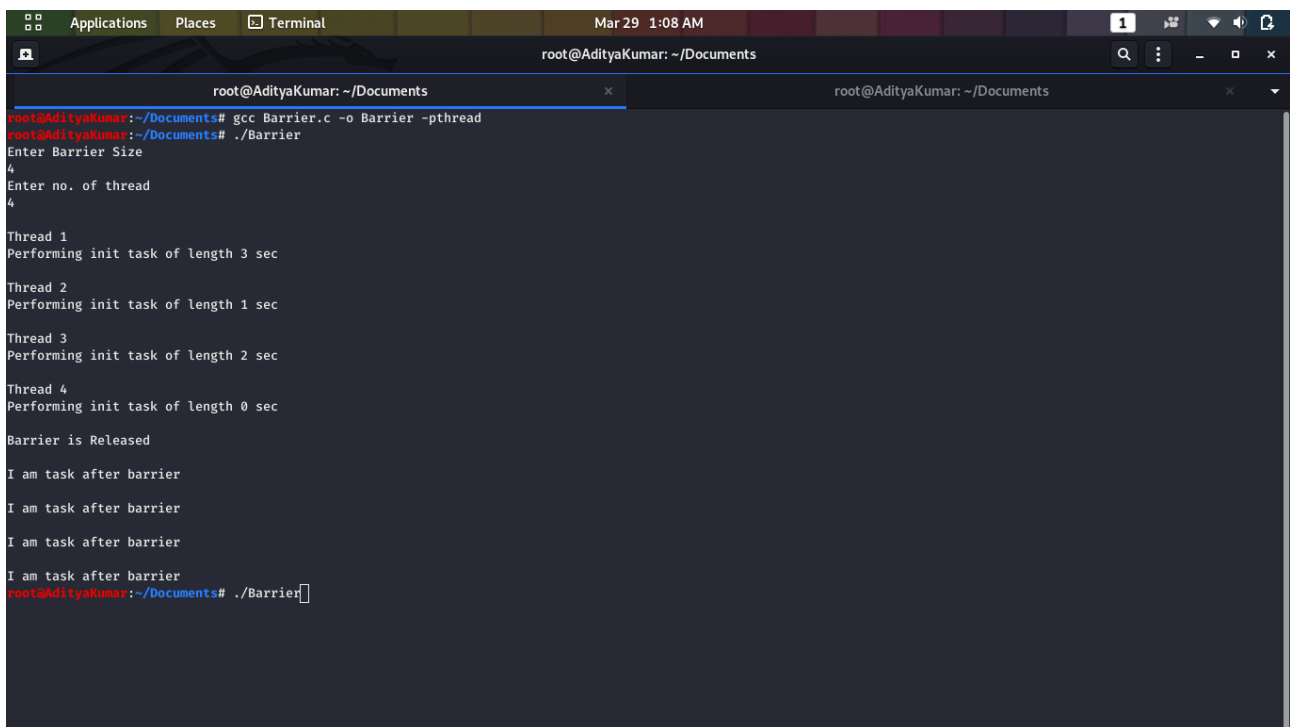


A terminal window titled 'root@AdityaKumar: ~/Documents' showing the execution of a program. The user enters '-1' for barrier size and '-12' for number of threads, both of which are rejected with the message 'You are entering wrong data.' The user then enters '-1' for barrier size and '-10' for number of threads, which are also rejected. The terminal shows the following output:

```
root@AdityaKumar: ~/Documents# gcc Barrier.c -o Barrier -pthread
root@AdityaKumar: ~/Documents# ./Barrier
Enter Barrier Size
-1
Enter no. of thread
-12
You are entering wrong data.
Enter Barrier Size
-1
Enter no. of thread
-10
You are entering wrong data.
Enter Barrier Size

```

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



A terminal window titled 'root@AdityaKumar: ~/Documents' showing the execution of the program with 4 threads and a barrier size of 4. The threads perform tasks of different lengths (3, 1, 2, and 0 seconds) and then wait at the barrier. After the barrier is released, each thread prints 'I am task after barrier'. The terminal shows the following output:

```
root@AdityaKumar: ~/Documents# gcc Barrier.c -o Barrier -pthread
root@AdityaKumar: ~/Documents# ./Barrier
Enter Barrier Size
4
Enter no. of thread
4

Thread 1
Performing init task of length 3 sec

Thread 2
Performing init task of length 1 sec

Thread 3
Performing init task of length 2 sec

Thread 4
Performing init task of length 0 sec

Barrier is Released

I am task after barrier
I am task after barrier
I am task after barrier
I am task after barrier
root@AdityaKumar: ~/Documents# ./Barrier
```

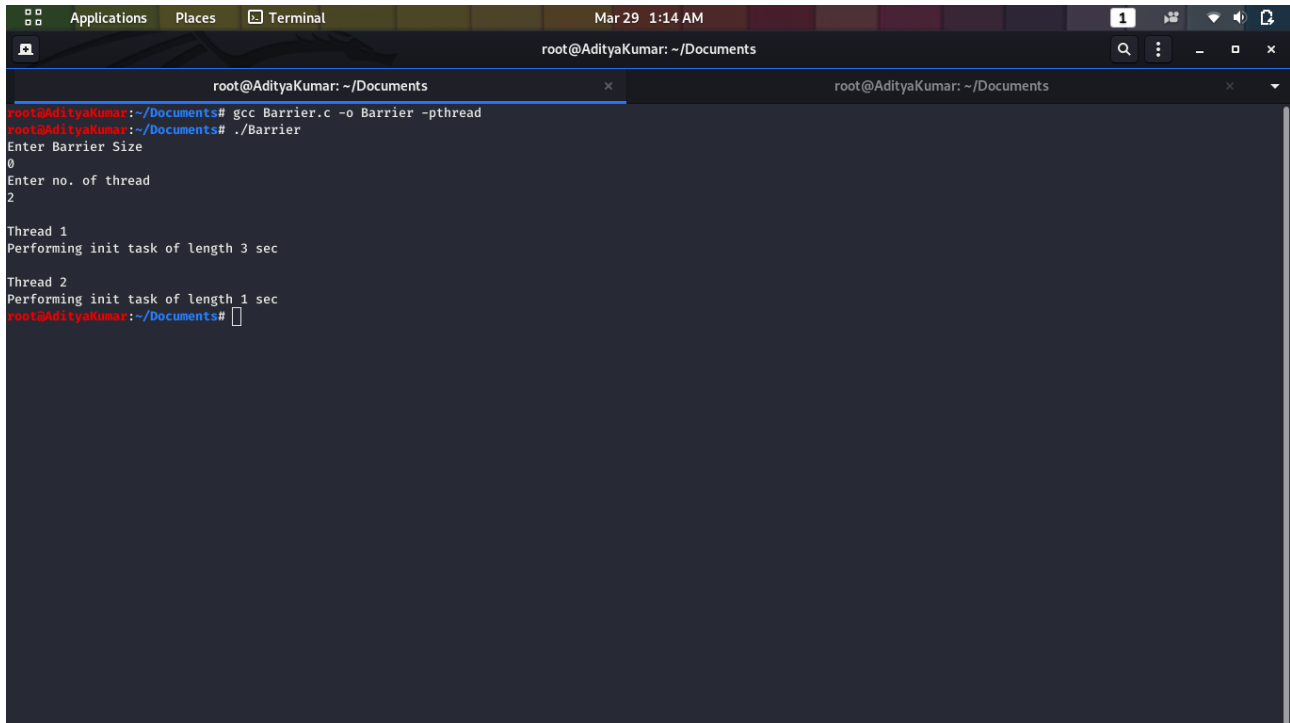
Case 3: when no. of thread is less than size of barrier .

```
root@AdityaKumar: ~/Documents
root@AdityaKumar:~/Documents# gcc Barrier.c -o Barrier -pthread
root@AdityaKumar:~/Documents# ./Barrier
Enter Barrier Size
6
Enter no. of thread
3
Thread 1
Performing init task of length 3 sec
Thread 3
Performing init task of length 2 sec
Thread 2
Performing init task of length 1 sec
I am task after barrier
I am task after barrier
I am task after barrier
root@AdityaKumar:~/Documents#
```

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

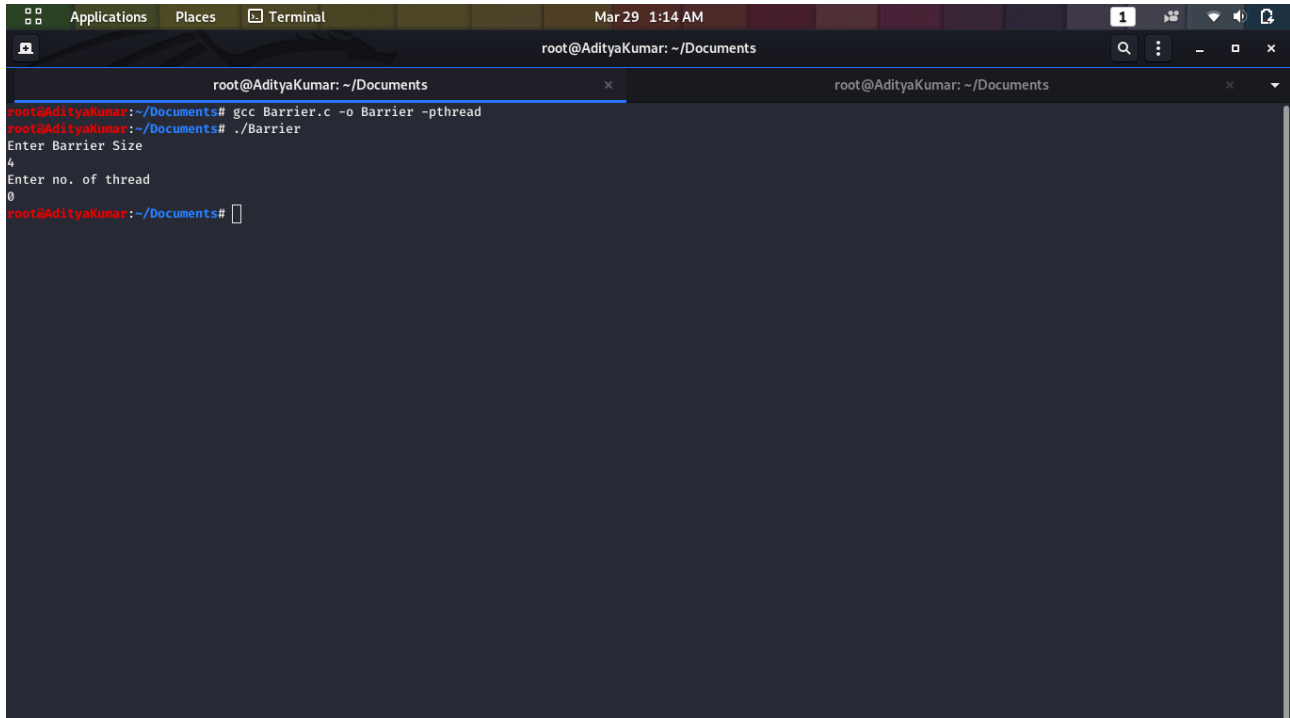
```
root@AdityaKumar: ~/Documents
root@AdityaKumar:~/Documents# gcc Barrier.c -o Barrier -pthread
root@AdityaKumar:~/Documents# ./Barrier
Enter Barrier Size
2
Enter no. of thread
4
Thread 1
Performing init task of length 3 sec
Thread 2
Performing init task of length 1 sec
Thread 3
Performing init task of length 2 sec
Thread 4
Performing init task of length 0 sec
Barrier is Released
I am task after barrier
I am task after barrier
Barrier is Released
I am task after barrier
I am task after barrier
root@AdityaKumar:~/Documents#
```

Case 5: when size of Barrier equal to '0'.

A terminal window titled 'root@AdityaKumar: ~/Documents' showing the execution of a C program. The user enters '0' for the barrier size and '2' for the number of threads. Thread 1 performs a 3-second task, and Thread 2 performs a 1-second task. The program then terminates.

```
root@AdityaKumar: ~/Documents
root@AdityaKumar:~/Documents# gcc Barrier.c -o Barrier -pthread
root@AdityaKumar:~/Documents# ./Barrier
Enter Barrier Size
0
Enter no. of thread
2
Thread 1
Performing init task of length 3 sec
Thread 2
Performing init task of length 1 sec
root@AdityaKumar:~/Documents#
```

Case 6: when thread equal to '0'.

A terminal window titled 'root@AdityaKumar: ~/Documents' showing the execution of the same C program. The user enters '4' for the barrier size and '0' for the number of threads. The program then terminates.

```
root@AdityaKumar: ~/Documents
root@AdityaKumar:~/Documents# gcc Barrier.c -o Barrier -pthread
root@AdityaKumar:~/Documents# ./Barrier
Enter Barrier Size
4
Enter no. of thread
0
root@AdityaKumar:~/Documents#
```

GitHub Link: https://github.com/adityasingh110/OS_Assignment_Q16.git