

Operating System

Report Assignment Simulation Based

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.

The return value of each function is used to identify error conditions. Each function will return 0 under normal operation and will return -1 if an error occurs. A testing harness is provided in the source code download to test your implementation of the barrier.

Student Name: Anil Yarra

Student ID: 11711807

Section No: EE034

Roll No.: B55

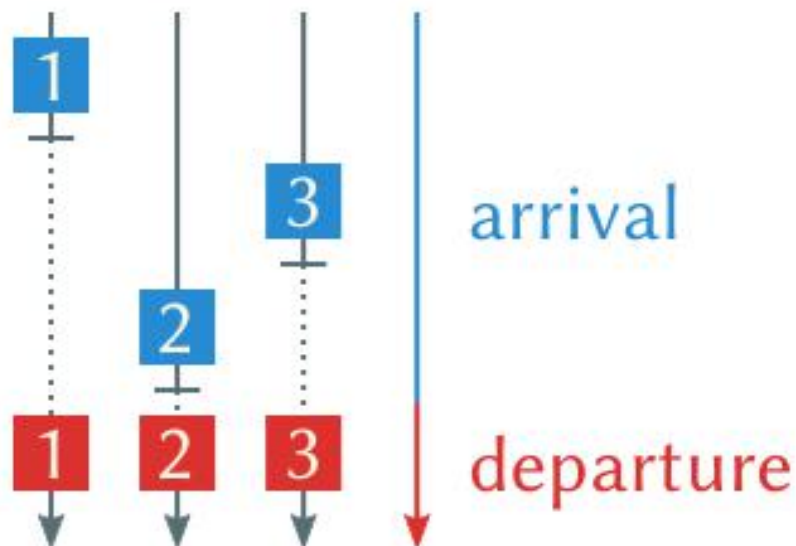
Email Address: anilyoung8@gmail.com

GitHub Link: https://github.com/anilyarra/Threads_Barriers/blob/master/ospro.c

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.



Threads

A thread is a flow of execution through the process code, with its own program counter that keeps track of which instruction to execute next, system registers which hold its current working variables, and a stack which contains the execution history.

A thread shares with its peer threads few information like code segment, data segment and open files. When one thread alters a code segment memory item, all other threads see that.

A thread is also called a **lightweight process**. Threads provide a way to improve application performance through parallelism. Threads represent a software approach to improving performance of operating system by reducing the overhead thread is equivalent to a classical process.

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:

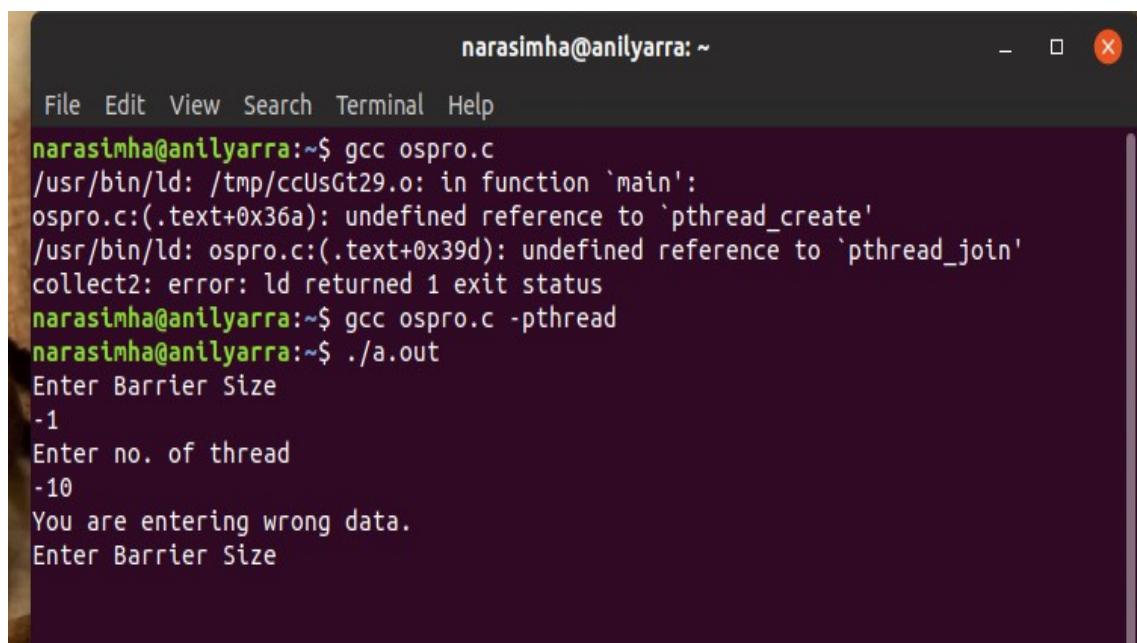
$O(n)$ complexity. "n" is no of thread_count

Compile And Run:

Use following commond to compile program -
gcc ospro.c -pthread
use following commond to run program- a.out

Test Cases :

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



```
narasimha@anilyarra: ~  
File Edit View Search Terminal Help  
narasimha@anilyarra:~$ gcc ospro.c  
/usr/bin/ld: /tmp/ccUsGt29.o: in function 'main':  
ospro.c:(.text+0x36a): undefined reference to 'pthread_create'  
/usr/bin/ld: ospro.c:(.text+0x39d): undefined reference to 'pthread_join'  
collect2: error: ld returned 1 exit status  
narasimha@anilyarra:~$ gcc ospro.c -pthread  
narasimha@anilyarra:~$ ./a.out  
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.

```
narasimha@anilyarra: ~  
File Edit View Search Terminal Help  
narasimha@anilyarra:~$ ./a.out  
Enter Barrier Size  
3  
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  
  
Barrier is Released  
  
I am task after barrier  
  
I am task after barrier  
  
I am task after barrier  
narasimha@anilyarra:~$
```

Case 3:

when no. of thread is less than size of barrier .

```
narasimha@anilyarra: ~  
File Edit View Search Terminal Help  
narasimha@anilyarra:~$ ./a.out  
Enter Barrier Size  
5  
Enter no. of thread  
3  
  
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  
  
I am task after barrier  
  
I am task after barrier  
  
I am task after barrier  
narasimha@anilyarra:~$
```

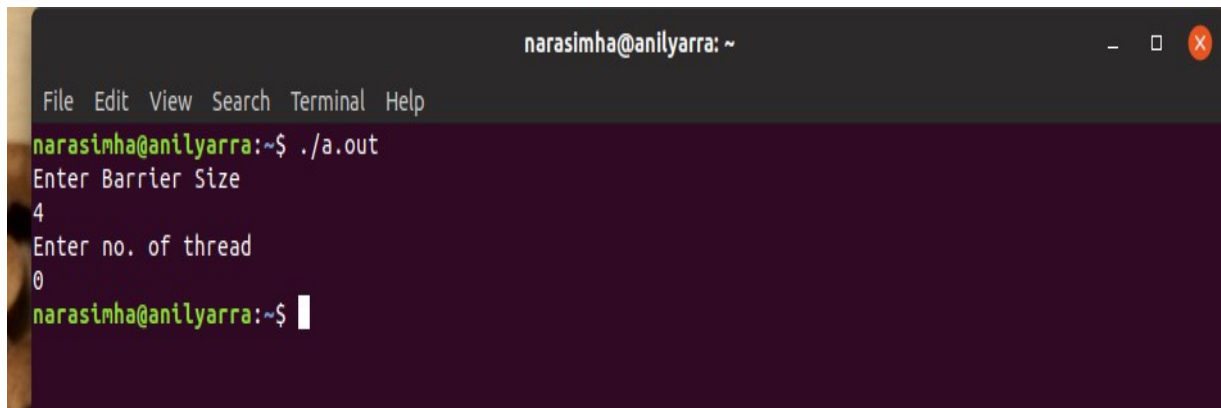
Case 4: when no. of thread is greater than size of Barrier.

```
narasimha@anilyarra: ~  
File Edit View Search Terminal Help  
narasimha@anilyarra:~$ ./a.out  
Enter Barrier Size  
3  
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  
  
I am task after barrier  
Barrier is Released  
  
I am task after barrier  
I am task after barrier  
I am task after barrier  
narasimha@anilyarra:~$
```

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

```
narasimha@anilyarra: ~  
File Edit View Search Terminal Help  
narasimha@anilyarra:~$ ./a.out  
Enter Barrier Size  
0  
Enter no. of thread  
3  
  
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  
narasimha@anilyarra:~$
```

Case 6: when thread equal to '0'.



```
narasimha@anilyarra: ~  
File Edit View Search Terminal Help  
narasimha@anilyarra:~$ ./a.out  
Enter Barrier Size  
4  
Enter no. of thread  
0  
narasimha@anilyarra:~$
```

The image shows a terminal window titled 'narasimha@anilyarra: ~'. The window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal content shows the user running './a.out'. The program prompts for 'Enter Barrier Size' and the user enters '4'. It then prompts for 'Enter no. of thread' and the user enters '0'. The prompt returns to 'narasimha@anilyarra:~\$'.