Report
CS 3523

The barrier is implemented using a class. The class has:

- Constructor to initialise semaphores.
- Deconstructor to destroy semaphores.
- init() to initialise the value of size.
- barrier_point() to set the barrier at the point.

Two semaphores have been used to implement the barrier:

1) Mutex (Mutual Exclusion) – This semaphore is used for the increment of count which keeps track of how many threads have reached the barrier. This needs to be locked while incrementing because while one thread is writing to it , other might be reading from it which may again lead to a race condition. This has been initialised to 1, to indicate it is unlocked and any thread can write count after locking it.

2) Barrier – This semaphore is initialised to 0, to indicate that the barrier is locked in the beginning and no thread may pass through. When the count is 1, then the barrier signals to wakeup a thread and as that thread passes on, it wakes up another thread and so every thread pass the barrier.

The exponential distribution with mean ‘t’ is achieved using the std::exponential distribution<> which comes with C++. The threads are put to sleep according to the values received from it. The main program waits for all the threads to finish execution after which the class deconstructor destroys the semaphores and program exits.

Ayush Prakash
CS15BTECH11008