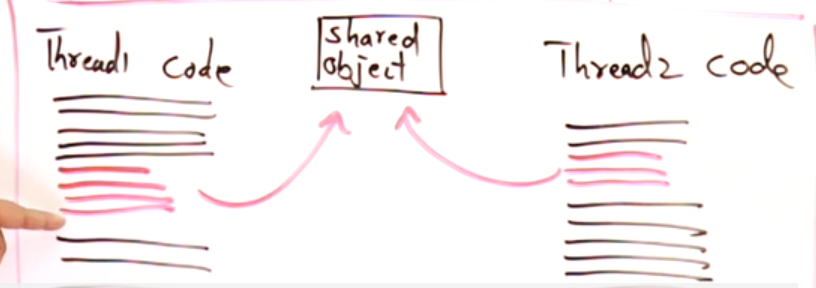
# Synchronization

All threads will have their individual stack. But object is created in heap.

## Resource sharing

## Critical Section



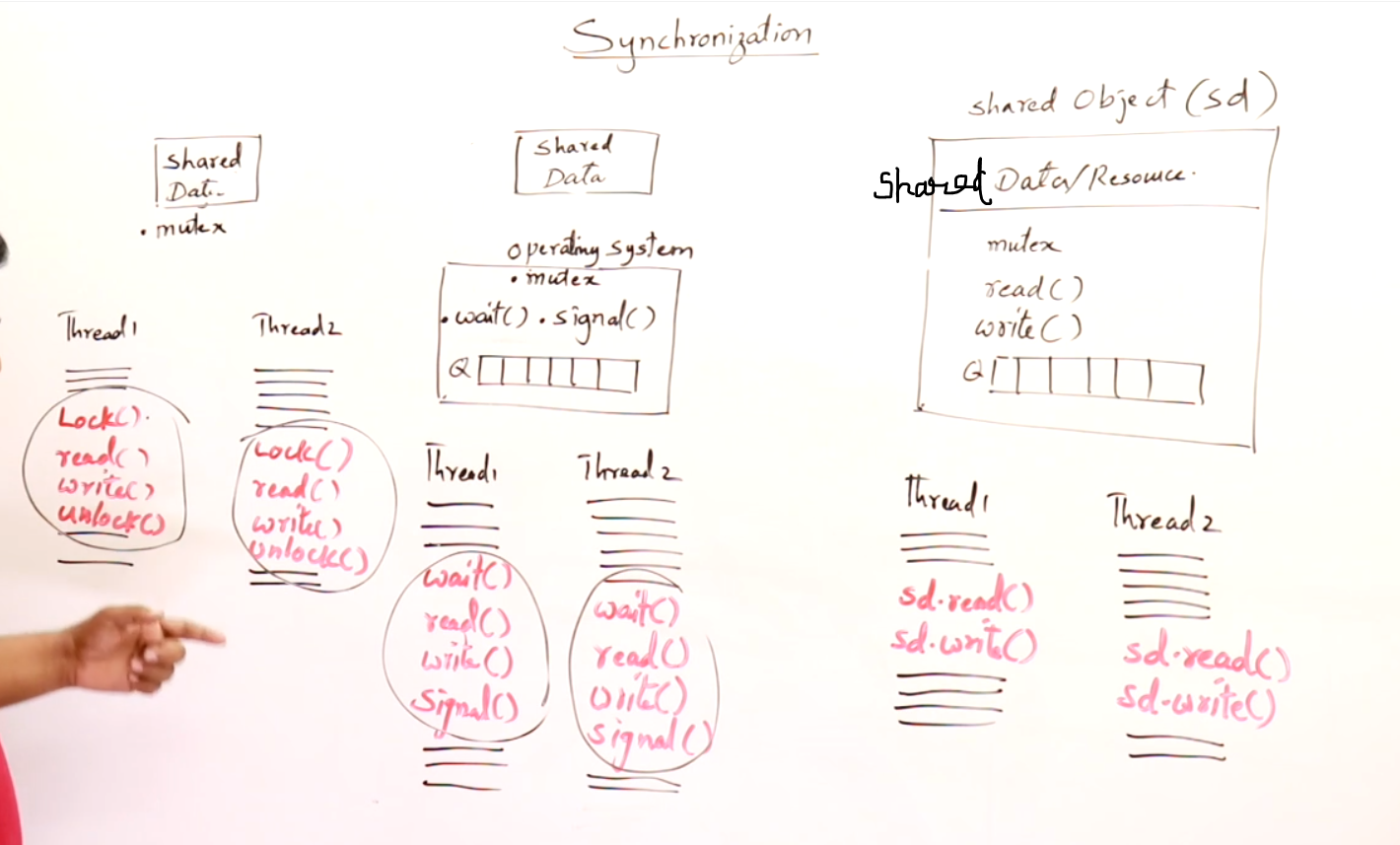
Red lines are the critical section part of their respective codes.

## Mutual Exclusion

Will not allow two threads to access the same resource simultaneously. This is known as Mutual exclusion. Preventing any other thread to access the same object is known as mutual exclusion.  
There should be some co-ordination between threads so that one thread can access the shared object at that instant.

Happening of one prevents the happening of other.  
Thread-1 accessing a resource will prevent thread-2 to access.

1. There should be some system which should take care of the shared resources by allowing only 1 thread at a time 🡪 Locking/mutex || semaphores || monitor.  
   Locking/mutex ( mutex will take care)  
   Semaphores (OS will take care)  
   Monitor (objects will take care)
2. Apart from the system there must be some co-ordination between threads (i.e t-1 and t-2 should communicate with each-other to access the shared resource. 🡪 Race condition || Inter-thread communication.



Locking/Mutex Semaphore Monitor

## Locking/Mutex

Here threads will lock() , read() , write() , unlock() automatically. Two / more threads will communicate with each-other and provide mutual exclusion.

## Semaphore

Threads knows how to read and write, OS will take care of locking [i.e wait()] and unlock [i.e signal()]. Operating System will achieve mutual exclusion.

## Monitor

Thread doesn’t know how to read and write. But it can perform by calling the methods. Objects in Object Oriented Programming(OOPS) will contain data and methods() and these objects provides mutual exclusion.

Eg: Barber Shop , Customers wants hair-cut, trim bears but customers don’t know how to do??  
So they simply calls the methods which they want, and stylists will make the customers sit in the queue and allow one-by-one when their turn comes.

## Race Condition

## Inter-thread communication