**Mutex**

Mutex or Mutual Exclusion Object is used to give access to a resource to only one process at a time. The mutex object allows all the processes to use the same resource but at a time, only one process is allowed to use the resource. Mutex uses the lock-based technique to handle the critical section problem.

Whenever a process requests for a resource from the system, then the system will create a mutex object with a unique name or ID. So, whenever the process wants to use that resource, then the process occupies a lock on the object. After locking, the process uses the resource and finally releases the mutex object. After that, other processes can create the mutex object in the same manner and use it.

By locking the object, that particular resource is allocated to that particular process and no other process can take that resource. So, in the critical section, no other processes are allowed to use the shared resource. In this way, the process synchronization can be achieved with the help of a mutex object.

**Semaphore**

Semaphore is an integer variable **S**, that is initialized with the number of resources present in the system and is used for process synchronization. It uses two functions to change the value of **S** i.e. ***wait()*** and ***signal()***. Both these functions are used to modify the value of semaphore but the functions allow only one process to change the value at a particular time i.e. no two processes can change the value of semaphore simultaneously. There are two categories of semaphores i.e. **Counting semaphores** and **Binary semaphores**.

In **Counting semaphores**, firstly, the semaphore variable is initialized with the number of resources available. After that, whenever a process needs some resource, then the wait() function is called and the value of the semaphore variable is decreased by one. The process then uses the resource and after using the resource, the signal() function is called and the value of the semaphore variable is increased by one. So, when the value of the semaphore variable goes to 0 i.e all the resources are taken by the process and there is no resource left to be used, then if some other process wants to use resources then that process has to wait for its turn. In this way, we achieve the process synchronization.

In **Binary semaphores**, the value of the semaphore variable will be 0 or 1. Initially, the value of semaphore variable is set to 1 and if some process wants to use some resource then the wait() function is called and the value of the semaphore is changed to 0 from 1. The process then uses the resource and when it releases the resource then the signal() function is called and the value of the semaphore variable is increased to 1. If at a particular instant of time, the value of the semaphore variable is 0 and some other process wants to use the same resource then it has to wait for the release of the resource by the previous process. In this way, process synchronization can be achieved. It is similar to mutex but here locking is not performed.

**Difference between Mutex and Semaphore**

Till now, we have learned about mutex and semaphore. Most of you might have guessed the difference between these two. Let's have a look into the difference between mutex and semaphore:

* Mutex uses a locking mechanism i.e. if a process wants to use a resource then it locks the resource, uses it and then release it. But on the other hand, semaphore uses a signalling mechanism where wait() and signal() methods are used to show if a process is releasing a resource or taking a resource.
* A mutex is an object but semaphore is an integer variable.
* In semaphore, we have wait() and signal() functions. But in mutex, there is no such function.
* A mutex object allows multiple process threads to access a single shared resource but only one at a time. On the other hand, semaphore allows multiple process threads to access the finite instance of the resource until available.
* In mutex, the lock can be acquired and released by the same process at a time. But the value of the semaphore variable can be modified by any process that needs some resource but only one process can change the value at a time.