**Semaphores in Process Synchronization**

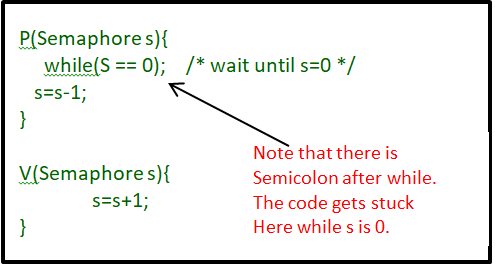
Semaphore was proposed by Dijkstra in 1965 which is a very significant technique to manage concurrent processes by using a simple integer value, which is known as a semaphore.

Semaphore is simply an integer variable that is shared between threads. This variable is used to solve the critical section problem and to achieve process synchronization in the multiprocessing environment.   
Semaphores are of two types:

1. **Binary Semaphore –**   
   This is also known as mutex lock. It can have only two values – 0 and 1. Its value is initialized to 1. It is used to implement the solution of critical section problems with multiple processes.
2. **Counting Semaphore –**   
   Its value can range over an unrestricted domain. It is used to control access to a resource that has multiple instances.

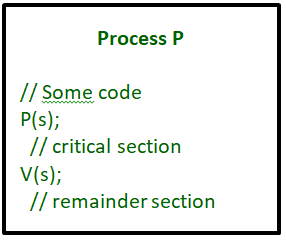
Now let us see how it does so.

First, look at two operations that can be used to access and change the value of the semaphore variable.



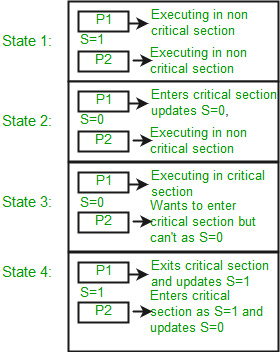
**Some point regarding P and V operation :**

1. P operation is also called wait, sleep, or down operation, and V operation is also called signal, wake-up, or up operation.
2. Both operations are atomic and semaphore(s) is always initialized to one. Here atomic means that variable on which read, modify and update happens at the same time/moment with no pre-emption i.e. in-between read, modify and update no other operation is performed that may change the variable.
3. A critical section is surrounded by both operations to implement process synchronization. See the below image. The critical section of Process P is in between P and V operation.



Now, let us see how it implements mutual exclusion. Let there be two processes P1 and P2 and a semaphore s is initialized as 1. Now if suppose P1 enters in its critical section then the value of semaphore s becomes 0. Now if P2 wants to enter its critical section then it will wait until s > 0, this can only happen when P1 finishes its critical section and calls V operation on semaphore s.

This way mutual exclusion is achieved. Look at the below image for details which is Binary semaphore.



**Limitations :**

1. One of the biggest limitations of semaphore is priority inversion.
2. Deadlock, suppose a process is trying to wake up another process which is not in a sleep state. Therefore, a deadlock may block indefinitely.
3. The operating system has to keep track of all calls to wait and to signal the semaphore.