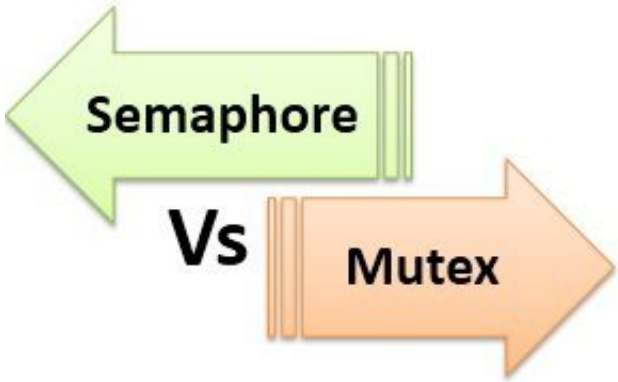
Difference Between Semaphore and Mutex

December 21, 2016 — Leave a Comment



Process synchronization plays an important role in maintaining the consistency of shared data. Both the software and hardware solutions are present for handling critical section problem. But hardware solutions for critical section problem are quite difficult to implement. In today's article, we will discuss two software based solution to handle critical section problem i.e. Semaphore and Mutex.

The basic difference between semaphore and mutex is that semaphore is a signalling mechanism i.e. processes perform wait() and signal() operation to indicate whether they are

acquiring or releasing the resource, while Mutex is locking mechanism, the process has to acquire the lock on mutex object if it wants to acquire the resource. There are some more differences between semaphore and mutex, let us discuss them with the help of comparison chart shown below.

Content: Semaphore Vs Mutex

Comparison Chart

BASIS FOR COMPARISON	SEMAPHORE	MUTEX
Basic	Semaphore is a signalling mechanism.	Mutex is a locking mechanism.
Existence	Semaphore is an integer variable.	Mutex is an object.
Function	Semaphore allow multiple program threads to access a finite instance of resources.	Mutex allow multiple program thread to access a single resource but not simultaneously.
Ownership	Semaphore value can be changed by any process acquiring or releasing the resource.	Mutex object lock is released only by the process that has acquired the lock on it.
Categorize	Semaphore can be categorized into counting semaphore and binary semaphore.	Mutex is not categorized further.
Operation	Semaphore value is modified using wait() and signal() operation.	Mutex object is locked or unlocked by the process requesting or releasing the resource.

Resources Occupied	If all resources are being used, the process requesting for resource performs wait() operation and block itself till semaphore count become greater than one.	If a mutex object is already locked, the process requesting for resources waits and queued by the system till lock is released.

Definition of Semaphore

Semaphore is a process synchronization tool. Semaphore is typically an **integer variable S** that is initialized to the number of resources present in the system and the value of semaphore can be **modified** only by two functions **wait()** and **signal()** apart from initialization.

The wait() and signal() operation modify the value of the semaphore **indivisibly**. It means that when a process is modifying the value of the semaphore, no other process can simultaneously modify the value of the semaphore. Semaphore are distinguished by the operating system in two categories **Counting semaphores** and **Binary semaphore**.

In **Counting Semaphore**, the semaphore S value is initialized to the **number of resource**s present in the system. Whenever a process wants to access the resource it performs **wait()** operation on the semaphore and **decrements** the value of semaphore by one. When it releases the resource, it performs **signal()** operation on the semaphore and **increments** the value of semaphore by one. When the semaphore count goes to 0, it means all resources are occupied by the processes. If a process need to use a resource when semaphore count is 0, it executes wait() and get **blocked** until the value of semaphore becomes greater than 0.

In **Binary semaphore**, the value of semaphore ranges between **0** and **1**. It is similar to mutex lock, but mutex is a locking mechanism whereas, the semaphore is a signalling mechanism. In binary semaphore, if a process wants to access the resource it performs **wait()** operation on the semaphore and decrements the value of semaphore from 1 to 0. When it releases the resource, it performs a **signal()** operation on the semaphore and increments its value to 1. If the value of semaphore is 0 and a process want to access the resource it performs **wait()** operation and block itself till the current process utilizing the resources releases the resource.

Definition of Mutex

Mutual Exclusion Object is shortly termed as Mutex. From the term mutual exclusion, we can understand that only one process at a time can access the given resource. The mutex object allows the multiple program threads to use the same resource but one at a time not simultaneously.

When a program starts it request the system to creates a mutex object for a given resource. The system creates the mutex object with a unique name or ID. Whenever the program thread wants to use the resource it occupies lock on mutex object, utilizes the resource and after use, it releases the lock on mutex object. Then the next process is allowed to acquire the lock on mutex object.

Meanwhile, a process has acquired the lock on mutex object no other thread/process can access that resource. If the mutex object is already locked, the process desiring to acquire the lock on mutex object has to wait and is queued up by the system till the mutex object is unlocked.

Key Differences Between Semaphore and Mutex

1. Semaphore is a **signalling** mechanism as wait() and signal() operation performed on semaphore variable indicates whether a process is acquiring the resource or releasing the resource. On the other hands, the mutex is a **locking** mechanism, as to acquire a resource, a process needs to lock the mutex object and while releasing a resource process has to unlock mutex object.
2. Semaphore is typically an **integer** variable whereas, mutex is an **object**.
3. Semaphore allows multiple program threads to access the **finite instance of resources**. On the other hands, Mutex allows multiple program threads to access a **single shared resource** but one at a time.
4. Semaphore variable value can be modified by **any** process that acquires or

releases resource by performing wait() and signal() operation. On the other hands, lock acquired on the mutex object can be released **only** by the process that has acquired the lock on mutex object.

5. Semaphore are of two types **counting semaphore and binary semaphore** which is quite similar to the mutex.
6. Semaphore variable value is modified by **wait()** and **signal()** operation apart from initialization. However, the mute object is locked or unlocked by the process acquiring or releasing the resource.
7. If all the resources are acquired by the process, and no resource is free then the process desiring to acquire resource performs wait() operation on semaphore variable and **blocks** itself till the count of semaphore become greater than 0. But if a mutex object is already locked then the process desiring to acquire resource **waits** and get **queued** by the system till the resource is released and mutex object gets unlocked.

Conclusion:

Semaphore is a better option in case there are multiple instances of resources available. In the case of single shared resource mutex is a better choice.

Related Differences:

1. **Difference Between sleep() and wait() Method in Java**
2. **Difference Between Analog and Digital Signal**
3. **Difference Between Preemptive and Non-Preemptive Scheduling in OS**

4. **Difference Between fork() and vfork()**
5. **Difference Between Process and Thread in Java**

Filed Under: Operating System

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

I'm not a robot

reCAPTCHA

Privacy - Terms

POST COMMENT

Search the site ...

TOP 10 DIFFERENCES

- Difference between Synchronous and Asynchronous Transmission
- Difference Between Preemptive and Non-Preemptive Scheduling in OS
- Difference Between Logical and Physical Address in Operating System
- Difference Between while and do-while Loop
- Difference between Simplex, Half duplex and Full Duplex Transmission Modes
- Difference Between One-Dimensional (1D) and Two-Dimensional (2D) Array
- Difference Between LAN, MAN and WAN
- Difference Between Go-Back-N and Selective Repeat Protocol
- Difference Between Circuit Switching and Packet Switching
- Difference Between Pure ALOHA and Slotted ALOHA

RECENT ADDITION

- Difference Between RAM and ROM Memory
- Difference Between Supercomputer and Mainframe Computer
- Difference Between Primary and Secondary Memory
- Difference Between Loosely Coupled and Tightly Coupled Multiprocessor System
- Difference Between Magnetic Tape and Magnetic Disk

CATEGORIES

DBMS

Networking

Operating System

Programming

Tech Differences
96 likes

Like Page

Share