

Mutex vs Semaphore: What's the Difference?

What is Semaphore?

Semaphore is simply a variable that is non-negative and shared between threads. A semaphore is a signaling mechanism, and a thread that is waiting on a semaphore can be signaled by another thread. It uses two atomic operations, 1)wait, and 2) signal for the process synchronization.

A semaphore either allows or disallows access to the resource, which depends on how it is set up.

In this tutorial, you will learn:

- [What is Mutex?](#)
- [Use of Semaphore](#)
- [Use of Mutex](#)
- [Difference between Semaphore vs. Mutex](#)
- [Common Misconceptions about Mutex and Semaphore](#)
- [Advantages of Semaphore](#)
- [Advantages of Mutex](#)
- [Disadvantage of Semaphores](#)
- [Disadvantages of Mutex](#)

What is Mutex?

The full form of Mutex is Mutual Exclusion Object. It is a special type of binary semaphore which used for controlling access to the shared resource. It includes a priority inheritance mechanism to avoid extended priority inversion problems. It allows current higher priority tasks to be kept in the blocked state for the shortest time possible. However, priority inheritance does not correct priority- inversion but only minimizes its effect.

KEY DIFFERENCE

- Mutex is a locking mechanism whereas Semaphore is a signaling mechanism
- Mutex is just an object while Semaphore is an integer
- Mutex has no subtype whereas Semaphore has two types, which are counting semaphore and binary semaphore.
- Semaphore supports wait and signal operations modification, whereas Mutex is only modified by the process that may request or release a resource.
- Semaphore value is modified using wait () and signal () operations, on the other hand, Mutex operations are locked or unlocked.

Use of Semaphore

In the case of a single buffer, we can separate the 4 KB buffer into four 1 KB buffers. Semaphore can be associated with these four buffers. This allows users and producers to work on different buffers at the same time.

Use of Mutex

A mutex provides mutual exclusion, which can be either producer or consumer that can have the key (mutex) and proceed with their work. As long as producer fills buffer, the user needs to wait, and vice versa. In Mutex lock, all the time, only a single thread can work with the entire buffer.

Difference between Semaphore vs. Mutex



[./images/1/120319_0816_MutexvsSema1.png](#)

Parameters	Semaphore	Mutex
Mechanism	It is a type of signaling mechanism.	It is a locking mechanism.
Data Type	Semaphore is an integer variable.	Mutex is just an object.
Modification	The wait and signal operations can modify a semaphore.	It is modified only by the process that may request or release a resource.
Resource management	If no resource is free, then the process requires a resource that should execute wait operation. It should wait until the count of the semaphore is greater than 0.	If it is locked, the process has to wait. The process should be kept in a queue. This needs to be accessed only when the mutex is unlocked.
Thread	You can have multiple program threads.	You can have multiple program threads in mutex but not simultaneously.
Ownership	Value can be changed by any process releasing or obtaining the resource.	Object lock is released only by the process, which has obtained the lock on it.
Types	Types of Semaphore are counting semaphore and binary semaphore.	Mutex has no subtypes.
Operation	Semaphore value is modified using wait () and signal () operation.	Mutex object is locked or unlocked.

Parameters	Semaphore	Mutex
Resources Occupancy	It is occupied if all resources are being used and the process requesting for resource performs wait () operation and blocks itself until semaphore count becomes >1.	In case if the object is already locked, the process requesting resources waits and is queued by the system before lock is released.

Common Facts about Mutex and Semaphore

Here, are few common facts about Mutex and Semaphore:

- Only one task can acquire the mutex. So, there is ownership associated with a mutex, and only the owner can release the mutex.
- The reasons for using mutex and semaphore are different maybe because of similarity in their implementation, a mutex would be referred to as binary semaphore.
- One highly known misconception is that Mutexes and Semaphores are almost same, with the only difference being that a Mutex is capable of counting to 1, while Semaphores able to count from 0 to N.
- There is always uncertainty between binary semaphore and mutex. You may hear that a mutex is a binary semaphore, which is not correct.

Advantages of Semaphore

Here, are pros/benefits of using Semaphore:

- It allows more than one thread to access the critical section
- Semaphores are machine-independent.
- Semaphores are implemented in the machine-independent code of the microkernel.
- They do not allow multiple processes to enter the critical section.
- As there is busy waiting in semaphore, there is never a wastage of process time and resources.

- They are machine-independent, which should be run in the machine-independent code of the microkernel.
- They allow flexible management of resources.

Advantages of Mutex

Here, are important pros/benefits of Mutex

- Mutexes are just simple locks obtained before entering its critical section and then releasing it.
- Since only one thread is in its critical section at any given time, there are no race conditions, and data always remain consistent.

Disadvantage of Semaphores

Here, are cons/drawback of semaphore

- One of the biggest limitations of a semaphore is priority inversion.
- The operating system has to keep track of all calls to wait and signal semaphore.
- Their use is never enforced, but it is by convention only.
- In order to avoid deadlocks in semaphore, the Wait and Signal operations require to be executed in the correct order.
- Semaphore programming is a complex method, so there are chances of not achieving mutual exclusion.
- It is also not a practical method for large scale use as their use leads to loss of modularity.
- Semaphore is more prone to programmer error.
- It may cause deadlock or violation of mutual exclusion due to programmer error.


Disadvantages of Mutex

Here, are cons/drawback of Mutex

- If a thread obtains a lock and goes to sleep or it is preempted, then the other thread may not be able to move forward. This may lead to starvation.
- It can't be locked or unlocked from a different context than the one that acquired it.
- Only one thread should be allowed in the critical section at a time.
- The normal implementation may lead to busy waiting state, which wastes CPU time.

YOU MIGHT LIKE:

BLOG

(/free-winzip-
 alternative.html)
(/free-winzip-


alternative.html)

[15 Best FREE Winzip Alternatives \(ZIP File Software\) in 2021](#)

(/free-winzip-
alternative.html)

JENKINS

(/jenkins-alternative.html)


 (/jenkins-
alternative.html)

[15 Best Jenkins Alternatives \(Open Source & Paid\) in 2021](#)

(/jenkins-alternative.html)

SDLC


(/best-iphone-data-
recovery-software.html)

 (/best-iphone-data-
recovery-
software.html)

[15 BEST iPhone Data Recovery Software & Apps in 2021 \(Free/Paid\)](#)

(/best-iphone-data-recovery-
software.html)

SDLC


(/difference-between-ar-
 vr.html) (/difference-
between-ar-vr.html)

[AR Vs VR: Difference Between Augmented Reality, Virtual Reality](#)

(/difference-between-ar-
vr.html)

R PROGRAMMING

(/r-t-test-one-sample.html)


 (/r-t-test-one-
sample.html)

[T Test in R: One Sample and Paired \(with Example\)](#)

(/r-t-test-one-sample.html)

PERL

(/perl-tutorials.html)

 (/perl-
tutorials.html)

[Perl Tutorial for Beginners: Learn in 1 Day](#)

(/perl-tutorials.html)

Operating System Tutorial

[Virtual Memory in OS \(/virtual-memory-in-operating-system.html\)](#)

[Banker's Algorithm \(/bankers-algorithm-in-operating-system.html\)](#)

[Mutex Vs Semaphore \(/mutex-vs-semaphore.html\)](#)

[Process vs Thread \(/difference-between-process-and-thread.html\)](#)

f [\(https://www.facebook.com/guru99com/\)](https://www.facebook.com/guru99com/).

t [\(https://twitter.com/guru99com/\)](https://twitter.com/guru99com/). **in**

[\(https://www.linkedin.com/company/guru99/\)](https://www.linkedin.com/company/guru99/).



[. \(https://www.youtube.com/channel/UC19i1XD6k88KqHlET8atqEQ\)](https://www.youtube.com/channel/UC19i1XD6k88KqHlET8atqEQ).



[. \(https://forms.aweber.com/form/46/724807646.htm\)](https://forms.aweber.com/form/46/724807646.htm).

About

[About Us \(/about-us.html\)](/about-us.html)

[Advertise with Us \(/advertise-us.html\)](/advertise-us.html)

[Write For Us \(/become-an-instructor.html\)](/become-an-instructor.html)

[Contact Us \(/contact-us.html\)](/contact-us.html)

Career Suggestion

[SAP Career Suggestion Tool \(/best-sap-module.html\)](/best-sap-module.html)

[Software Testing as a Career \(/software-testing-career-complete-guide.html\)](/software-testing-career-complete-guide.html)

Interesting

[eBook \(/ebook-pdf.html\)](/ebook-pdf.html)

[Blog \(/blog/\)](/blog/)

[Quiz \(/tests.html\)](/tests.html)

[SAP eBook \(/sap-ebook-pdf.html\)](/sap-ebook-pdf.html)

Execute online

[Execute Java Online \(/try-java-editor.html\)](/try-java-editor.html)

[Execute Javascript \(/execute-javascript-online.html\)](/execute-javascript-online.html)

[Execute HTML \(/execute-html-online.html\)](/execute-html-online.html)

[Execute Python \(/execute-python-online.html\)](/execute-python-online.html)

© Copyright - Guru99 2021

[Privacy Policy \(/privacy-policy.html\)](/privacy-policy.html) | [Affiliate
Disclaimer \(/affiliate-earning-disclaimer.html\)](#) | [ToS
\(/terms-of-service.html\)](#)