SFML

* [Main Page](http://docs.google.com/index.htm)
* [Modules](http://docs.google.com/modules.htm)
* [Classes](http://docs.google.com/annotated.htm)
* [Files](http://docs.google.com/files.htm)
* [Class List](http://docs.google.com/annotated.htm)
* [Class Index](http://docs.google.com/classes.htm)
* [Class Hierarchy](http://docs.google.com/hierarchy.htm)
* [Class Members](http://docs.google.com/functions.htm)
* **sf**
* [Mutex](http://docs.google.com/classsf_1_1Mutex.htm)

[Public Member Functions](#_gjdgxs) | [List of all members](http://docs.google.com/classsf_1_1Mutex-members.htm)

sf::Mutex Class Reference

[System module](http://docs.google.com/group__system.htm)

Blocks concurrent access to shared resources from multiple threads. [More...](http://docs.google.com/classsf_1_1Mutex.htm#details)

#include <[Mutex.hpp](http://docs.google.com/Mutex_8hpp_source.htm)>

Inheritance diagram for sf::Mutex:



| Public Member Functions | |
| --- | --- |
|  | [Mutex](http://docs.google.com/classsf_1_1Mutex.htm#a9bd52a48320fd7b6db8a78037aad276e) () |
|  | Default constructor. |
|  | |
|  | [~Mutex](http://docs.google.com/classsf_1_1Mutex.htm#a9f76a67b7b6d3918131a692179b4e3f2) () |
|  | Destructor. |
|  | |
| void | [lock](http://docs.google.com/classsf_1_1Mutex.htm#a1a16956a6bbea764480c1b80f2e45763) () |
|  | [Lock](http://docs.google.com/classsf_1_1Lock.htm) the mutex. |
|  | |
| void | [unlock](http://docs.google.com/classsf_1_1Mutex.htm#ade71268ffc5e80756652058b01c23c33) () |
|  | Unlock the mutex. |
|  | |

## Detailed Description

Blocks concurrent access to shared resources from multiple threads.

[Mutex](http://docs.google.com/classsf_1_1Mutex.htm) stands for "MUTual EXclusion".

A mutex is a synchronization object, used when multiple threads are involved.

When you want to protect a part of the code from being accessed simultaneously by multiple threads, you typically use a mutex. When a thread is locked by a mutex, any other thread trying to lock it will be blocked until the mutex is released by the thread that locked it. This way, you can allow only one thread at a time to access a critical region of your code.

Usage example:

Database database; // this is a critical resource that needs some protection

[sf::Mutex](http://docs.google.com/classsf_1_1Mutex.htm) mutex;

void thread1()

{

mutex.[lock](http://docs.google.com/classsf_1_1Mutex.htm#a1a16956a6bbea764480c1b80f2e45763)(); // this call will block the thread if the mutex is already locked by thread2

database.write(...);

mutex.[unlock](http://docs.google.com/classsf_1_1Mutex.htm#ade71268ffc5e80756652058b01c23c33)(); // if thread2 was waiting, it will now be unblocked

}

void thread2()

{

mutex.[lock](http://docs.google.com/classsf_1_1Mutex.htm#a1a16956a6bbea764480c1b80f2e45763)(); // this call will block the thread if the mutex is already locked by thread1

database.write(...);

mutex.[unlock](http://docs.google.com/classsf_1_1Mutex.htm#ade71268ffc5e80756652058b01c23c33)(); // if thread1 was waiting, it will now be unblocked

}

Be very careful with mutexes. A bad usage can lead to bad problems, like deadlocks (two threads are waiting for each other and the application is globally stuck).

To make the usage of mutexes more robust, particularly in environments where exceptions can be thrown, you should use the helper class [sf::Lock](http://docs.google.com/classsf_1_1Lock.htm) to lock/unlock mutexes.

SFML mutexes are recursive, which means that you can lock a mutex multiple times in the same thread without creating a deadlock. In this case, the first call to [lock()](http://docs.google.com/classsf_1_1Mutex.htm#a1a16956a6bbea764480c1b80f2e45763) behaves as usual, and the following ones have no effect. However, you must call [unlock()](http://docs.google.com/classsf_1_1Mutex.htm#ade71268ffc5e80756652058b01c23c33) exactly as many times as you called [lock()](http://docs.google.com/classsf_1_1Mutex.htm#a1a16956a6bbea764480c1b80f2e45763). If you don't, the mutex won't be released.

See Also[sf::Lock](http://docs.google.com/classsf_1_1Lock.htm)

Definition at line [47](http://docs.google.com/Mutex_8hpp_source.htm#l00047) of file [Mutex.hpp](http://docs.google.com/Mutex_8hpp_source.htm).

## Constructor & Destructor Documentation

| sf::Mutex::Mutex | ( |  | ) |  |
| --- | --- | --- | --- | --- |

Default constructor.

| sf::Mutex::~Mutex | ( |  | ) |  |
| --- | --- | --- | --- | --- |

Destructor.

## Member Function Documentation

| void sf::Mutex::lock | ( |  | ) |  |
| --- | --- | --- | --- | --- |

[Lock](http://docs.google.com/classsf_1_1Lock.htm) the mutex.

If the mutex is already locked in another thread, this call will block the execution until the mutex is released.

See Also[unlock](http://docs.google.com/classsf_1_1Mutex.htm#ade71268ffc5e80756652058b01c23c33)

| void sf::Mutex::unlock | ( |  | ) |  |
| --- | --- | --- | --- | --- |

Unlock the mutex.

See Also[lock](http://docs.google.com/classsf_1_1Mutex.htm#a1a16956a6bbea764480c1b80f2e45763)

The documentation for this class was generated from the following file:

* [Mutex.hpp](http://docs.google.com/Mutex_8hpp_source.htm)

Copyright � Laurent Gomila  ::  Documentation generated by [doxygen](http://www.doxygen.org/)  ::