Search: Go

Not logged in

Reference <mutex> mutex

register log in

class
# std::mutex

<mutex>

`class mutex;`

Mutex class

A *mutex* is a *lockable object* that is designed to signal when critical sections of code need exclusive access, preventing other threads with the same protection from executing concurrently and access the same memory locations.

`mutex` objects provide *exclusive ownership* and do not support recursivity (i.e., a thread shall not lock a `mutex` it already owns) -- see `recursive_mutex` for an alternative class that does.

It is guaranteed to be a standard-layout class.

Member types

member type	description
`native_handle_type`	Type returned by `native_handle` (only defined if library implementation supports it)

Member functions

(constructor)	Construct mutex (public member function)
lock	Lock mutex (public member function)
try_lock	Lock mutex if not locked (public member function)
unlock	Unlock mutex (public member function)
native_handle	Get native handle (public member function)

Example

```cpp
// mutex example
#include <iostream>       // std::cout
#include <thread>         // std::thread
#include <mutex>          // std::mutex

std::mutex mtx;           // mutex for critical section

void print_block (int n, char c) {
  // critical section (exclusive access to std::cout signaled by locking mtx):
  mtx.lock();
  for (int i=0; i<n; ++i) { std::cout << c; }
  std::cout << '\n';
  mtx.unlock();
}

int main ()
{
  std::thread th1 (print_block,50,'*');
  std::thread th2 (print_block,50,'$');

  th1.join();
  th2.join();

  return 0;
}
```

Possible output (order of lines may vary, but characters are never mixed):

```
**************************************************
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```