**Thread**

# class thread;

Class to represent individual threads of execution.

A thread of execution is a sequence of instructions that can be executed concurrently with other such sequences in multithreading environments, while sharing a same address space.

An initialized thread object represents an active thread of execution; Such a thread object is joinable, and has a unique thread id.

A default-constructed (non-initialized) thread object is not joinable, and its thread id is common for all non-joinable threads.

A joinable thread becomes not joinable if moved from, or if either join or detach are called on them.

# Member types

[**id**](http://www.cplusplus.com/reference/thread/thread/id/)

Thread id (public member type )

[**native\_handle\_type**](http://www.cplusplus.com/reference/thread/thread/native_handle_type/)

Native handle type (public member type )

# Member functions

**[(constructor)](#_Constructor)**

Construct thread (public member function )

**[(destructor)](#_Destructor)**

Thread destructor (public member function )

**[operator=](#_operator=)**

Move-assign thread (public member function )

**[get\_id](#_get_id)**

Get thread id (public member function )

**[joinable](#_joinable)**

Check if joinable (public member function )

**[join](#_join)**

Join thread (public member function )

[**detach**](http://www.cplusplus.com/reference/thread/thread/detach/)

Detach thread (public member function )

[**swap**](http://www.cplusplus.com/reference/thread/thread/swap/)

Swap threads (public member function )

[**native\_handle**](http://www.cplusplus.com/reference/thread/thread/native_handle/)

Get native handle (public member function )

[**hardware\_concurrency [static]**](http://www.cplusplus.com/reference/thread/thread/hardware_concurrency/)

Detect hardware concurrency (public static member function )

# Non-member overloads

[**swap (thread)**](http://www.cplusplus.com/reference/thread/thread/swap-free/)

Swap threads (function )

# Constructor

|  |  |
| --- | --- |
| default (1) | thread () noexcept; |
| initialization (2) | **template <class Fn, class... Args>**  **explicit thread (Fn&& fn, Args&&... args);** |
| copy [deleted] (3) | **thread (const thread&) = delete;** |
| move (4) | **thread (thread&& x) noexcept;** |

= delete means that the compiler will not generate those constructors for you. AFAIK this is only allowed on copy constructor and assignment operator.

1. default constructor

Construct a thread object that does not represent any thread of execution.

1. initialization constructor

Construct a thread object that represents a new joinable thread of execution.

The new thread of execution calls fn passing args as arguments (using decay copies of its lvalue or rvalue references).

The completion of this construction synchronizes with the beginning of the invocation of this copy of fn.

1. copy constructor

Deleted constructor form (thread objects cannot be copied).

1. move constructor

Construct a thread object that acquires the thread of execution represented by x (if any). This operation does not affect the execution of the moved thread in any way, it simply transfers its handler.

The x object no longer represents any thread of execution.

thread objects that are joinable shall either be joined or detached before they are destroyed.

#include <iostream>

#include <utility>

#include <thread>

#include <chrono>

#include <functional>

#include <atomic>

void f1(int n) {

for (int i = 0; i < 5; ++i) {

std::cout << "Thread 1 executing\n";

++n;

std::this\_thread::sleep\_for(std::chrono::milliseconds(10));

}

}

void f2(int& n) {

for (int i = 0; i < 5; ++i) {

std::cout << "Thread 2 executing\n";

++n;

std::this\_thread::sleep\_for(std::chrono::milliseconds(10));

}

}

int main() {

int n = 0;

std::thread t1; // t1 is not a thread

std::thread t2(f1, n + 1); // pass by value

std::thread t3(f2, std::ref(n)); // pass by reference

std::thread t4(std::move(t3)); // t4 is now running f2(). t3 is no longer a thread

t2.join();

t4.join();

std::cout << "Final value of n is " << n << '\n';

return 0;

}

Output:

Thread 1 executing

Thread 2 executing

Thread 1 executing

Thread 2 executing

Thread 1 executing

Thread 2 executing

Thread 1 executing

Thread 2 executing

Thread 2 executing

Thread 1 executing

Final value of n is 5

# Destructor

|  |
| --- |
| ~thread(); |

Destroys the thread object.

If \*this has an associated thread (joinable() == true), std::terminate() is called.

# operator=

|  |  |
| --- | --- |
| move (1) | thread& operator= (thread&& rhs) noexcept; |
| copy [deleted] (2) | **thread& operator= (const thread&) = delete;** |

**Return value** \*this

If the object is currently not joinable, it acquires the thread of execution represented by rhs (if any).

If it is joinable, terminate() is called. After the call, rhs no longer represents any thread of execution (as if default-constructed).

thread objects cannot be copied (2).

After this call, this->get\_id() is equal to the value of rhs.get\_id() prior to the call, and other no longer represents a thread of execution.

#include <iostream>

#include <utility>

#include <thread>

#include <chrono>

#include <functional>

#include <atomic>

void foo(int n) {

std::cout << "inside foo n : " << n << std::endl;

}

int main() {

std::thread t1(foo, 3); // pass by value

std::thread t2; // t1 is not a thread

std::cout << "t1.get\_id() : " << t1.get\_id() << std::endl;

std::cout << "t2.get\_id() : " << t2.get\_id() << std::endl;

t2 = std::thread(foo, 7);

std::cout << "t1.get\_id() : " << t1.get\_id() << std::endl;

std::cout << "t2.get\_id() : " << t2.get\_id() << std::endl;

t1.join();

t2.join();

//t4.join();

return 0;

}

Output:

t1.get\_id() : 127309200688896

t2.get\_id() : thread::id of a non-executing thread

inside foo n : 3

t1.get\_id() : 127309200688896

t2.get\_id() : 127309192009472

inside foo n : 7

# get\_id

|  |
| --- |
| id get\_id() const noexcept; |

Returns a value of std::thread::id identifying the thread associated with \*this.

If the thread object is joinable, the function returns a value that uniquely identifies the thread.

If the thread object is not joinable, the function returns a default-constructed object of member type thread::id.

**Return value**

A value of type std::thread::id identifying the thread associated with \*this. If there is no thread associated, default constructed std::thread::id is returned.

# joinable

# join