C++
Information
Tutorials
Reference
Articles
Forum

Reference
C library:
Containers:
Input/Output:
Multi-threading:
<atomic>
<condition_variable>
<future>
<mutex>
<thread>
Other:

<thread>
classes:
thread
namespaces:
this_thread

this_thread
functions within namespace:
get_id
sleep_for
sleep_until
yield

function

std::this_thread::sleep_for

<thread>

```
template <class Rep, class Period>
  void sleep_for (const chrono::duration<Rep,Period>& rel_time);
```

Sleep for time span

Blocks execution of the calling thread during the span of time specified by *rel_time*.

The execution of the current thread is stopped until at least *rel_time* has passed from now. Other threads continue their execution.

Parameters

rel_time
: The time span after which the calling thread shall resume its execution.
Note that multi-threading management operations may cause certain delays beyond this.
`duration` is an object that represents a specific *relative time*.

Return value

none

Example

```
// this_thread::sleep_for example
#include <iostream>       // std::cout, std::endl
#include <thread>         // std::this_thread::sleep_for
#include <chrono>         // std::chrono::seconds

int main()
{
  std::cout << "countdown:\n";
  for (int i=10; i>0; --i) {
    std::cout << i << std::endl;
    std::this_thread::sleep_for (std::chrono::seconds(1));
  }
  std::cout << "Lift off!\n";

  return 0;
}
```

Output (after 10 seconds):

```
countdown:
10
9
8
7
6
5
4
3
2
1
Lift off!
```

Exception safety

If the type of *rel_time* never throws exceptions (like the instantiations of `duration` in header `<chrono>`), this function never throws exceptions (no-throw guarantee).

See also

yield	Yield to other threads (function)
sleep_until	Sleep until time point (function)