library

**Multi-threading**

**Atomic and thread support**

Support for atomics and threads:

**Headers**

[**<atomic>**](http://www.cplusplus.com/reference/atomic/)

Atomic (header)

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

Thread (header)

[**<mutex>**](http://www.cplusplus.com/reference/mutex/)

Mutex (header)

[**<condition\_variable>**](http://www.cplusplus.com/reference/condition_variable/)

Condition variable (header)

[**<future>**](http://www.cplusplus.com/reference/future/)

Future (header)

header

# <atomic>

**Atomic**

Atomic types are types that encapsulate a value whose access is guaranteed to not cause data races and can be used to synchronize memory accesses among different threads.  
  
This header declares two C++ classes, [atomic](http://www.cplusplus.com/atomic) and [atomic\_flag](http://www.cplusplus.com/atomic_flag), that implement all the features of atomic types in self-contained classes. The header also declares an entire set of *C-style* types and functions compatible with the atomic support in C.

### Classes

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

Atomic (class template )

[**atomic\_flag**](http://www.cplusplus.com/reference/atomic/atomic_flag/)

Atomic flag (class )

### Types

[**memory\_order**](http://www.cplusplus.com/reference/atomic/memory_order/)

Memory order (enum )

#### C-style atomic types

The following *atomic types* are also defined in this header; each with the same behavior as the respective instantiation of [atomic](http://www.cplusplus.com/atomic) for the listed *contained type*.

|  |  |  |
| --- | --- | --- |
| **contained type** | **atomic type** | **description** |
| bool | atomic\_bool |  |
| char | atomic\_char | atomics for [*fundamental integral types*](http://www.cplusplus.com/is_fundamental). These are either typedefs of the corresponding full specialization of the [atomic](http://www.cplusplus.com/atomic) class template or a base class of such specialization. |
| signed char | atomic\_schar |
| unsigned char | atomic\_uchar |
| short | atomic\_short |
| unsigned short | atomic\_ushort |
| int | atomic\_int |
| unsigned int | atomic\_uint |
| long | atomic\_long |
| unsigned long | atomic\_ulong |
| long long | atomic\_llong |
| unsigned long long | atomic\_ullong |
| wchar\_t | atomic\_wchar\_t |
| char16\_t | atomic\_char16\_t |
| char32\_t | atomic\_char32\_t |
| intmax\_t | atomic\_intmax\_t | atomics for [*width-based integrals*](http://www.cplusplus.com/cinttypes) (those defined in [<cinttypes>](http://www.cplusplus.com/%3Ccinttypes%3E)). Each of these is either an alias of one of the above *atomics for fundamental integral types* or of a full specialization of the [atomic](http://www.cplusplus.com/atomic) class template with an *extended integral type*.  Where *N* is one in 8, 16, 32, 64, or any other type width supported by the library. |
| uintmax\_t | atomic\_uintmax\_t |
| int\_least*N*\_t | atomic\_int\_least*N*\_t |
| uint\_least*N*\_t | atomic\_uint\_least*N*\_t |
| int\_fast*N*\_t | atomic\_int\_fast*N*\_t |
| uint\_fast*N*\_t | atomic\_uint\_fast*N*\_t |
| intptr\_t | atomic\_intptr\_t |
| uintptr\_t | atomic\_uintptr\_t |
| size\_t | atomic\_size\_t |
| ptrdiff\_t | atomic\_ptrdiff\_t |

### Functions

[**kill\_dependency**](http://www.cplusplus.com/reference/atomic/kill_dependency/)

Kill dependency (function )

[**atomic\_thread\_fence**](http://www.cplusplus.com/reference/atomic/atomic_thread_fence/)

Thread fence (function )

[**atomic\_signal\_fence**](http://www.cplusplus.com/reference/atomic/atomic_signal_fence/)

Signal fence (function )

#### Functions for atomic objects (C-style)

[**atomic\_is\_lock\_free**](http://www.cplusplus.com/reference/atomic/atomic_is_lock_free/)

Is lock-free (function )

[**atomic\_init**](http://www.cplusplus.com/reference/atomic/atomic_init/)

Initialize atomic object (function )

[**atomic\_store**](http://www.cplusplus.com/reference/atomic/atomic_store/)

Modify contained value (function )

[**atomic\_store\_explicit**](http://www.cplusplus.com/reference/atomic/atomic_store_explicit/)

Modify contained value (explicit memory order) (function )

[**atomic\_load**](http://www.cplusplus.com/reference/atomic/atomic_load/)

Read contained value (function )

[**atomic\_load\_explicit**](http://www.cplusplus.com/reference/atomic/atomic_load_explicit/)

Read contained value (explicit memory order) (function )

[**atomic\_exchange**](http://www.cplusplus.com/reference/atomic/atomic_exchange/)

Read and modify contained value (function )

[**atomic\_exchange\_explicit**](http://www.cplusplus.com/reference/atomic/atomic_exchange_explicit/)

Read and modify contained value (explicit memory order) (function )

[**atomic\_compare\_exchange\_weak**](http://www.cplusplus.com/reference/atomic/atomic_compare_exchange_weak/)

Compare and exchange contained value (weak) (function )

[**atomic\_compare\_exchange\_weak\_explicit**](http://www.cplusplus.com/reference/atomic/atomic_compare_exchange_weak_explicit/)

Compare and exchange contained value (weak, explicit) (function )

[**atomic\_compare\_exchange\_strong**](http://www.cplusplus.com/reference/atomic/atomic_compare_exchange_strong/)

Compare and exchange contained value (strong) (function )

[**atomic\_compare\_exchange\_strong\_explicit**](http://www.cplusplus.com/reference/atomic/atomic_compare_exchange_strong_explicit/)

Compare and exchange contained value (strong, explicit) (function )

[**atomic\_fetch\_add**](http://www.cplusplus.com/reference/atomic/atomic_fetch_add/)

Add to contained value (function )

[**atomic\_fetch\_add\_explicit**](http://www.cplusplus.com/reference/atomic/atomic_fetch_add_explicit/)

Add to contained value (explicit memory order) (function )

[**atomic\_fetch\_sub**](http://www.cplusplus.com/reference/atomic/atomic_fetch_sub/)

Subtract from contained value (function )

[**atomic\_fetch\_sub\_explicit**](http://www.cplusplus.com/reference/atomic/atomic_fetch_sub_explicit/)

Subtract from contained value (explicit memory order) (function )

[**atomic\_fetch\_and**](http://www.cplusplus.com/reference/atomic/atomic_fetch_and/)

Apply bitwise AND to contained value (function )

[**atomic\_fetch\_and\_explicit**](http://www.cplusplus.com/reference/atomic/atomic_fetch_and_explicit/)

Apply bitwise AND to contained value (explicit memory order) (function )

[**atomic\_fetch\_or**](http://www.cplusplus.com/reference/atomic/atomic_fetch_or/)

Apply bitwise OR to contained value (function )

[**atomic\_fetch\_or\_explicit**](http://www.cplusplus.com/reference/atomic/atomic_fetch_or_explicit/)

Apply bitwise OR to contained value (explicit memory order) (function )

[**atomic\_fetch\_xor**](http://www.cplusplus.com/reference/atomic/atomic_fetch_xor/)

Apply bitwise XOR to contained value (function )

[**atomic\_fetch\_xor\_explicit**](http://www.cplusplus.com/reference/atomic/atomic_fetch_xor_explicit/)

Apply bitwise XOR to contained value (explicit memory order) (function )

#### Functions for atomic flags (C-style)

[**atomic\_flag\_test\_and\_set**](http://www.cplusplus.com/reference/atomic/atomic_flag_test_and_set/)

Test and set atomic flag (function )

[**atomic\_flag\_test\_and\_set\_explicit**](http://www.cplusplus.com/reference/atomic/atomic_flag_test_and_set_explicit/)

Test and set atomic flag (explicit memory order) (function )

[**atomic\_flag\_clear**](http://www.cplusplus.com/reference/atomic/atomic_flag_clear/)

Clear atomic flag (function )

[**atomic\_flag\_clear\_explicit**](http://www.cplusplus.com/reference/atomic/atomic_flag_clear_explicit/)

Clear atomic flag (explicit memory order) (function )

### Macro functions

[**ATOMIC\_VAR\_INIT**](http://www.cplusplus.com/reference/atomic/ATOMIC_VAR_INIT/)

Initialization of atomic variable (macro )

[**ATOMIC\_FLAG\_INIT**](http://www.cplusplus.com/reference/atomic/ATOMIC_FLAG_INIT/)

Initialization of atomic flag (macro )

### Macro constants

|  |  |  |
| --- | --- | --- |
| **macro** | **relative to types** | **defined as** |
| ATOMIC\_BOOL\_LOCK\_FREE | bool | 0 if the types are never lock-free. 1 it the types are sometimes lock-free. 2 if the types are always lock-free.  Consistent with the value returned by [atomic::is\_lock\_free](http://www.cplusplus.com/atomic::is_lock_free). |
| ATOMIC\_CHAR\_LOCK\_FREE | char  signed char  unsigned char |
| ATOMIC\_SHORT\_LOCK\_FREE | short  unsigned short |
| ATOMIC\_INT\_LOCK\_FREE | int  unsigned int |
| ATOMIC\_LONG\_LOCK\_FREE | long  unsigned long |
| ATOMIC\_LLONG\_LOCK\_FREE | long long  unsigned long long |
| ATOMIC\_WCHAR\_T\_LOCK\_FREE | wchar\_t |
| ATOMIC\_CHAR16\_T\_LOCK\_FREE | char16\_t |
| ATOMIC\_CHAR32\_T\_LOCK\_FREE | char32\_t |
| ATOMIC\_POINTER\_LOCK\_FREE | U\*  *(for any type U)* |

header

**<condition\_variable>**

**Condition variable**

Header that declares *condition variable* types:

**Classes**

[**condition\_variable**](http://www.cplusplus.com/reference/condition_variable/condition_variable/)

Condition variable (class )

[**condition\_variable\_any**](http://www.cplusplus.com/reference/condition_variable/condition_variable_any/)

Condition variable (any lock) (class )

**Enum classes**

[**cv\_status**](http://www.cplusplus.com/reference/condition_variable/cv_status/)

Condition variable status (enum class )

**Functions**

[**notify\_all\_at\_thread\_exit**](http://www.cplusplus.com/reference/condition_variable/notify_all_at_thread_exit/)

Notify all at thread exit (function )

header

# <future>

**Future**

Header with facilities that allow asynchronous access to values set by specific providers, possibly in a different thread.  
  
Each of these providers (which are either [promise](http://www.cplusplus.com/promise) or [packaged\_task](http://www.cplusplus.com/packaged_task) objects, or calls to [async](http://www.cplusplus.com/async)) share access to a *shared state* with a [future](http://www.cplusplus.com/future) object: the point where the *provider* makes the *shared state* ready is synchronized with the point the[future](http://www.cplusplus.com/future) object accesses the *shared state*. 

### Classes

#### Providers

[**promise**](http://www.cplusplus.com/reference/future/promise/)

Promise (class template )

[**packaged\_task**](http://www.cplusplus.com/reference/future/packaged_task/)

Packaged task (class template )

#### Futures

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

Future (class template )

[**shared\_future**](http://www.cplusplus.com/reference/future/shared_future/)

Shared future (class template )

#### Other types

[**future\_error**](http://www.cplusplus.com/reference/future/future_error/)

Future error exception (class )

[**future\_errc**](http://www.cplusplus.com/reference/future/future_errc/)

Error conditions for future objects (enum class )

[**future\_status**](http://www.cplusplus.com/reference/future/future_status/)

Return value for timed future operations (enum class )

[**launch**](http://www.cplusplus.com/reference/future/launch/)

Launching policy for async (enum class )

### Functions

#### Providers

[**async**](http://www.cplusplus.com/reference/future/async/)

Call function asynchronously (function template )

#### Other functions

[**future\_category**](http://www.cplusplus.com/reference/future/future_category/)

Return future category (function )

header

# <mutex>

**Mutex**

Header with facilities that allow *mutual exclusion* (mutex) of concurrent execution of critical sections of code, allowing to explicitly avoid data races.  
  
It contains *mutex types*, *lock types* and specific *functions*:

* **Mutex types** are [*lockable types*](http://www.cplusplus.com/Lockable) used to protect access to a *critical section* of code: [*locking*](http://www.cplusplus.com/mutex::lock) a mutex prevents other threads from locking it (exclusive access) until it is [*unlocked*](http://www.cplusplus.com/mutex::unlock): [mutex](http://www.cplusplus.com/mutex), [recursive\_mutex](http://www.cplusplus.com/recursive_mutex), [timed\_mutex](http://www.cplusplus.com/timed_mutex),[recursive\_timed\_mutex](http://www.cplusplus.com/recursive_timed_mutex).
* **Locks** are objects that manage a mutex by associating its access to their own lifetime: [lock\_guard](http://www.cplusplus.com/lock_guard), [unique\_lock](http://www.cplusplus.com/unique_lock).
* **Functions** to lock mutiple mutexes simultaneously ([try\_lock](http://www.cplusplus.com/try_lock), [lock](http://www.cplusplus.com/lock)) and to directly prevent concurrent execution of a specific function ([call\_once](http://www.cplusplus.com/call_once)).

### Classes

#### Mutexes

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

Mutex class (class )

[**recursive\_mutex**](http://www.cplusplus.com/reference/mutex/recursive_mutex/)

Recursive mutex class (class )

[**timed\_mutex**](http://www.cplusplus.com/reference/mutex/timed_mutex/)

Timed mutex class (class )

[**recursive\_timed\_mutex**](http://www.cplusplus.com/reference/mutex/recursive_timed_mutex/)

Recursive timed mutex (class )

#### Locks

[**lock\_guard**](http://www.cplusplus.com/reference/mutex/lock_guard/)

Lock guard (class template )

[**unique\_lock**](http://www.cplusplus.com/reference/mutex/unique_lock/)

Unique lock (class template )

#### Other types

[**once\_flag**](http://www.cplusplus.com/reference/mutex/once_flag/)

Flag argument type for call\_once (class )

[**adopt\_lock\_t**](http://www.cplusplus.com/reference/mutex/adopt_lock_t/)

Type of adopt\_lock (class )

[**defer\_lock\_t**](http://www.cplusplus.com/reference/mutex/defer_lock_t/)

Type of defer\_lock (class )

[**try\_to\_lock\_t**](http://www.cplusplus.com/reference/mutex/try_to_lock_t/)

Type of try\_to\_lock (class )

### Functions

[**try\_lock**](http://www.cplusplus.com/reference/mutex/try_lock/)

Try to lock multiple mutexes (function template )

[**lock**](http://www.cplusplus.com/reference/mutex/lock/)

Lock multiple mutexes (function template )

[**call\_once**](http://www.cplusplus.com/reference/mutex/call_once/)

Call function once (public member function )

header

# <thread>

**Thread**

Header that declares the [thread](http://www.cplusplus.com/thread) class and the [this\_thread](http://www.cplusplus.com/this_thread) namespace:

### Classes

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

Thread (class )

### Classes

[**this\_thread**](http://www.cplusplus.com/reference/thread/this_thread/)

This thread (namespace )