:mod:`_thread` --- Low-level threading API

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library) _thread.rst, line 1); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library) _thread.rst, line 4)

Unknown directive type "module".

```
.. module:: _thread
   :synopsis: Low-level threading API.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library) _thread.rst, line 7)

Unknown directive type "index".

```
.. index::
   single: light-weight processes
   single: processes, light-weight
   single: binary semaphores
   single: semaphores, binary
```

This module provides low-level primitives for working with multiple threads (also called :dfn:`light-weight processes` or :dfn:`tasks`) --- multiple threads of control sharing their global data space. For synchronization, simple locks (also called :dfn:`mutexes` or :dfn:`binary semaphores`) are provided. The :mod:`threading` module provides an easier to use and higher-level threading API built on top of this module.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library) _thread.rst, line 15); *backlink*

Unknown interpreted text role "dfn".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library) _thread.rst, line 15); *backlink*

Unknown interpreted text role "dfn".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library) _thread.rst, line 15); *backlink*

Unknown interpreted text role "dfn".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library) _thread.rst, line 15); *backlink*

Unknown interpreted text role "dfn".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library) _thread.rst, line 15); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library) _thread.rst, line 22)

Unknown directive type "index".

```
.. index::
   single: pthreads
```

```
   pair: threads; POSIX
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)_thread.rst, line 26)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.7
   This module used to be optional, it is now always available.
```

This module defines the following constants and functions:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)_thread.rst, line 31)

Unknown directive type "exception".

```
.. exception:: error

   Raised on thread-specific errors.

   .. versionchanged:: 3.3
      This is now a synonym of the built-in :exc:`RuntimeError`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)_thread.rst, line 39)

Unknown directive type "data".

```
.. data:: LockType

   This is the type of lock objects.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)_thread.rst, line 44)

Unknown directive type "function".

```
.. function:: start_new_thread(function, args[, kwargs])

   Start a new thread and return its identifier.  The thread executes the
   function *function* with the argument list *args* (which must be a tuple).
   The optional *kwargs* argument specifies a dictionary of keyword arguments.

   When the function returns, the thread silently exits.

   When the function terminates with an unhandled exception,
   :func:`sys.unraisablehook` is called to handle the exception. The *object*
   attribute of the hook argument is *function*. By default, a stack trace is
   printed and then the thread exits (but other threads continue to run).

   When the function raises a :exc:`SystemExit` exception, it is silently
   ignored.

   .. versionchanged:: 3.8
      :func:`sys.unraisablehook` is now used to handle unhandled exceptions.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)_thread.rst, line 64)

Unknown directive type "function".

```
.. function:: interrupt_main(signum=signal.SIGINT, /)

   Simulate the effect of a signal arriving in the main thread.
   A thread can use this function to interrupt the main thread, though
   there is no guarantee that the interruption will happen immediately.
```

```
   If given, *signum* is the number of the signal to simulate.
   If *signum* is not given, :data:`signal.SIGINT` is simulated.

   If the given signal isn't handled by Python (it was set to
   :data:`signal.SIG_DFL` or :data:`signal.SIG_IGN`), this function does
   nothing.

   .. versionchanged:: 3.10
      The *signum* argument is added to customize the signal number.

   .. note::
      This does not emit the corresponding signal but schedules a call to
      the associated handler (if it exists).
      If you want to truly emit the signal, use :func:`signal.raise_signal`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)_thread.rst, line 86)

Unknown directive type "function".

```
.. function:: exit()

   Raise the :exc:`SystemExit` exception.  When not caught, this will cause the
   thread to exit silently.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)_thread.rst, line 100)

Unknown directive type "function".

```
.. function:: allocate_lock()

   Return a new lock object.  Methods of locks are described below.  The lock is
   initially unlocked.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)_thread.rst, line 106)

Unknown directive type "function".

```
.. function:: get_ident()

   Return the 'thread identifier' of the current thread.  This is a nonzero
   integer.  Its value has no direct meaning; it is intended as a magic cookie to
   be used e.g. to index a dictionary of thread-specific data.  Thread identifiers
   may be recycled when a thread exits and another thread is created.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)_thread.rst, line 114)

Unknown directive type "function".

```
.. function:: get_native_id()

   Return the native integral Thread ID of the current thread assigned by the kernel.
   This is a non-negative integer.
   Its value may be used to uniquely identify this particular thread system-wide
   (until the thread terminates, after which the value may be recycled by the OS).

   .. availability:: Windows, FreeBSD, Linux, macOS, OpenBSD, NetBSD, AIX.

   .. versionadded:: 3.8
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)_thread.rst, line 126)

Unknown directive type "function".

```
.. function:: stack_size([size])

   Return the thread stack size used when creating new threads.  The optional
   *size* argument specifies the stack size to be used for subsequently created
   threads, and must be 0 (use platform or configured default) or a positive
   integer value of at least 32,768 (32 KiB). If *size* is not specified,
   0 is used.  If changing the thread stack size is
   unsupported, a :exc:`RuntimeError` is raised.  If the specified stack size is
   invalid, a :exc:`ValueError` is raised and the stack size is unmodified.  32 KiB
   is currently the minimum supported stack size value to guarantee sufficient
   stack space for the interpreter itself.  Note that some platforms may have
   particular restrictions on values for the stack size, such as requiring a
   minimum stack size > 32 KiB or requiring allocation in multiples of the system
   memory page size - platform documentation should be referred to for more
   information (4 KiB pages are common; using multiples of 4096 for the stack size is
   the suggested approach in the absence of more specific information).

   .. availability:: Windows, systems with POSIX threads.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)_thread.rst, line 146)

Unknown directive type "data".

```
.. data:: TIMEOUT_MAX

   The maximum value allowed for the *timeout* parameter of
   :meth:`Lock.acquire`. Specifying a timeout greater than this value will
   raise an :exc:`OverflowError`.

   .. versionadded:: 3.2
```

Lock objects have the following methods:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)_thread.rst, line 158)

Unknown directive type "method".

```
.. method:: lock.acquire(waitflag=1, timeout=-1)

   Without any optional argument, this method acquires the lock unconditionally, if
   necessary waiting until it is released by another thread (only one thread at a
   time can acquire a lock --- that's their reason for existence).

   If the integer *waitflag* argument is present, the action depends on its
   value: if it is zero, the lock is only acquired if it can be acquired
   immediately without waiting, while if it is nonzero, the lock is acquired
   unconditionally as above.

   If the floating-point *timeout* argument is present and positive, it
   specifies the maximum wait time in seconds before returning.  A negative
   *timeout* argument specifies an unbounded wait.  You cannot specify
   a *timeout* if *waitflag* is zero.

   The return value is ``True`` if the lock is acquired successfully,
   ``False`` if not.

   .. versionchanged:: 3.2
      The *timeout* parameter is new.

   .. versionchanged:: 3.2
      Lock acquires can now be interrupted by signals on POSIX.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)_thread.rst, line 184)

Unknown directive type "method".

```
.. method:: lock.release()

   Releases the lock.  The lock must have been acquired earlier, but not
```

```
necessarily by the same thread.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)_thread.rst, line 190)

Unknown directive type "method".

```
.. method:: lock.locked()

   Return the status of the lock: ``True`` if it has been acquired by some thread,
   ``False`` if not.
```

In addition to these methods, lock objects can also be used via the :keyword:`with` statement, e.g.:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)_thread.rst, line 195); *backlink*

Unknown interpreted text role "keyword".

```
import _thread

a_lock = _thread.allocate_lock()

with a_lock:
    print("a_lock is locked while this executes")
```

Caveats:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)_thread.rst, line 207)

Unknown directive type "index".

```
.. index:: module: signal
```

- Threads interact strangely with interrupts: the :exc:`KeyboardInterrupt` exception will be received by an arbitrary thread. (When the :mod:`signal` module is available, interrupts always go to the main thread.)

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)_thread.rst, line 209); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)_thread.rst, line 209); *backlink*

Unknown interpreted text role "mod".

- Calling :func:`sys.exit` or raising the :exc:`SystemExit` exception is equivalent to calling :func:`_thread.exit`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)_thread.rst, line 213); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)_thread.rst, line 213); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)_thread.rst, line 213); *backlink*

Unknown interpreted text role "func".

- It is not possible to interrupt the :meth:`acquire` method on a lock --- the :exc:`KeyboardInterrupt` exception will happen after the lock has been acquired.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)_thread.rst, line 216); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)_thread.rst, line 216); *backlink*

Unknown interpreted text role "exc".

- When the main thread exits, it is system defined whether the other threads survive. On most systems, they are killed without executing :keyword:`try` ... :keyword:`finally` clauses or executing object destructors.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)_thread.rst, line 219); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)_thread.rst, line 219); *backlink*

Unknown interpreted text role "keyword".

- When the main thread exits, it does not do any of its usual cleanup (except that :keyword:`try` ... :keyword:`finally` clauses are honored), and the standard I/O files are not flushed.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)_thread.rst, line 224); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)_thread.rst, line 224); *backlink*

Unknown interpreted text role "keyword".