Synchronization Primitives

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 1)

Unknown directive type "currentmodule".

```
.. currentmodule:: asyncio
```

Source code: :source:`Lib/asyncio/locks.py`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 9); *backlink*

Unknown interpreted text role "source".

---

asyncio synchronization primitives are designed to be similar to those of the :mod:`threading` module with two important caveats:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 13); *backlink*

Unknown interpreted text role "mod".

- asyncio primitives are not thread-safe, therefore they should not be used for OS thread synchronization (use :mod:`threading` for that);

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 16); *backlink*

  Unknown interpreted text role "mod".

- methods of these synchronization primitives do not accept the *timeout* argument; use the :func:`asyncio.wait_for` function to perform operations with timeouts.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 20); *backlink*

  Unknown interpreted text role "func".

asyncio has the following basic synchronization primitives:

- :class:`Lock`

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 26); *backlink*

  Unknown interpreted text role "class".

- :class:`Event`

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 27); *backlink*

  Unknown interpreted text role "class".

- :class:`Condition`

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

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 28); *backlink*

Unknown interpreted text role "class".

- :class:`Semaphore`

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 29); *backlink*

  Unknown interpreted text role "class".

- :class:`BoundedSemaphore`

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 30); *backlink*

  Unknown interpreted text role "class".

- :class:`Barrier`

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 31); *backlink*

  Unknown interpreted text role "class".

Lock

Implements a mutex lock for asyncio tasks. Not thread-safe.

An asyncio lock can be used to guarantee exclusive access to a shared resource.

The preferred way to use a Lock is an :keyword:`async with` statement:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 47); *backlink*

Unknown interpreted text role "keyword".

```
lock = asyncio.Lock()

# ... later
async with lock:
    # access shared state
```

which is equivalent to:

```
lock = asyncio.Lock()

# ... later
await lock.acquire()
try:
    # access shared state
finally:
    lock.release()
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 67)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.10
   Removed the *loop* parameter.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 70)

Unknown directive type "coroutinemethod".

```
.. coroutinemethod:: acquire()

   Acquire the lock.

   This method waits until the lock is *unlocked*, sets it to
   *locked* and returns ``True``.

   When more than one coroutine is blocked in :meth:`acquire`
   waiting for the lock to be unlocked, only one coroutine
   eventually proceeds.

   Acquiring a lock is *fair*: the coroutine that proceeds will be
   the first coroutine that started waiting on the lock.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 84)

Unknown directive type "method".

```
.. method:: release()

   Release the lock.

   When the lock is *locked*, reset it to *unlocked* and return.

   If the lock is *unlocked*, a :exc:`RuntimeError` is raised.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 92)

Unknown directive type "method".

```
.. method:: locked()

   Return ``True`` if the lock is *locked*.
```

Event

An event object. Not thread-safe.

An asyncio event can be used to notify multiple asyncio tasks that some event has happened.

An Event object manages an internal flag that can be set to *true* with the :meth:`~Event.set` method and reset to *false* with the :meth:`clear` method. The :meth:`~Event.wait` method blocks until the flag is set to *true*. The flag is set to *false* initially.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 107); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 107); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 107); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 112)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.10
   Removed the *loop* parameter.
```

Example:

```
async def waiter(event):
    print('waiting for it ...')
    await event.wait()
    print('... got it!')

async def main():
    # Create an Event object.
    event = asyncio.Event()

    # Spawn a Task to wait until 'event' is set.
    waiter_task = asyncio.create_task(waiter(event))

    # Sleep for 1 second and set the event.
    await asyncio.sleep(1)
    event.set()

    # Wait until the waiter task is finished.
    await waiter_task

asyncio.run(main())
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 140)

Unknown directive type "coroutinemethod".

```
.. coroutinemethod:: wait()

   Wait until the event is set.

   If the event is set, return ``True`` immediately.
   Otherwise block until another task calls :meth:`~Event.set`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 147)

Unknown directive type "method".

```
.. method:: set()

   Set the event.

   All tasks waiting for event to be set will be immediately
   awakened.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 154)

Unknown directive type "method".

```
.. method:: clear()

   Clear (unset) the event.

   Tasks awaiting on :meth:`~Event.wait` will now block until the
   :meth:`~Event.set` method is called again.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 161)

Unknown directive type "method".

```
.. method:: is_set()

   Return ``True`` if the event is set.
```

Condition

A Condition object. Not thread-safe.

An asyncio condition primitive can be used by a task to wait for some event to happen and then get exclusive access to a shared resource.

In essence, a Condition object combines the functionality of an :class:`Event` and a :class:`Lock`. It is possible to have multiple Condition objects share one Lock, which allows coordinating exclusive access to a shared resource between different tasks interested in particular states of that shared resource.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 177); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 177); *backlink*

Unknown interpreted text role "class".

The optional *lock* argument must be a :class:`Lock` object or `None`. In the latter case a new Lock object is created automatically.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 183); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 187)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.10
   Removed the *loop* parameter.
```

The preferred way to use a Condition is an :keyword:`async with` statement:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 190); *backlink*

Unknown interpreted text role "keyword".

```
cond = asyncio.Condition()

# ... later
async with cond:
    await cond.wait()
```

which is equivalent to:

```
cond = asyncio.Condition()

# ... later
await cond.acquire()
try:
    await cond.wait()
finally:
    cond.release()
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 210)

Unknown directive type "coroutinemethod".

```
.. coroutinemethod:: acquire()

   Acquire the underlying lock.

   This method waits until the underlying lock is *unlocked*,
   sets it to *locked* and returns ``True``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 217)

Unknown directive type "method".

```
.. method:: notify(n=1)

   Wake up at most *n* tasks (1 by default) waiting on this
   condition.  The method is no-op if no tasks are waiting.

   The lock must be acquired before this method is called and
   released shortly after.  If called with an *unlocked* lock
   a :exc:`RuntimeError` error is raised.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 226)

Unknown directive type "method".

```
.. method:: locked()

   Return ``True`` if the underlying lock is acquired.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 230)

Unknown directive type "method".

```
.. method:: notify_all()

   Wake up all tasks waiting on this condition.

   This method acts like :meth:`notify`, but wakes up all waiting
   tasks.

   The lock must be acquired before this method is called and
   released shortly after.  If called with an *unlocked* lock
   a :exc:`RuntimeError` error is raised.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 241)

Unknown directive type "method".

```
.. method:: release()

   Release the underlying lock.

   When invoked on an unlocked lock, a :exc:`RuntimeError` is
   raised.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 248)

Unknown directive type "coroutinemethod".

```
.. coroutinemethod:: wait()

   Wait until notified.

   If the calling task has not acquired the lock when this method is
   called, a :exc:`RuntimeError` is raised.

   This method releases the underlying lock, and then blocks until
   it is awakened by a :meth:`notify` or :meth:`notify_all` call.
   Once awakened, the Condition re-acquires its lock and this method
   returns ``True``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 260)

Unknown directive type "coroutinemethod".

```
.. coroutinemethod:: wait_for(predicate)
```

```
Wait until a predicate becomes *true*.

The predicate must be a callable which result will be
interpreted as a boolean value.  The final value is the
return value.
```

Semaphore

A Semaphore object. Not thread-safe.

A semaphore manages an internal counter which is decremented by each :meth:`acquire` call and incremented by each :meth:`release` call. The counter can never go below zero; when :meth:`acquire` finds that it is zero, it blocks, waiting until some task calls :meth:`release`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)asyncio-sync.rst, line 276); ***backlink***

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)asyncio-sync.rst, line 276); ***backlink***

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)asyncio-sync.rst, line 276); ***backlink***

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)asyncio-sync.rst, line 276); ***backlink***

Unknown interpreted text role "meth".

The optional *value* argument gives the initial value for the internal counter (`1` by default). If the given value is less than `0` a :exc:`ValueError` is raised.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)asyncio-sync.rst, line 282); ***backlink***

Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)asyncio-sync.rst, line 286)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.10
   Removed the *loop* parameter.
```

The preferred way to use a Semaphore is an :keyword:`async with` statement:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)asyncio-sync.rst, line 289); ***backlink***

Unknown interpreted text role "keyword".

```
sem = asyncio.Semaphore(10)

# ... later
async with sem:
    # work with shared resource
```

which is equivalent to:

```
sem = asyncio.Semaphore(10)
```

```
# ... later
await sem.acquire()
try:
    # work with shared resource
finally:
    sem.release()
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 309)

Unknown directive type "coroutinemethod".

```
.. coroutinemethod:: acquire()

   Acquire a semaphore.

   If the internal counter is greater than zero, decrement
   it by one and return ``True`` immediately.  If it is zero, wait
   until a :meth:`release` is called and return ``True``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 317)

Unknown directive type "method".

```
.. method:: locked()

   Returns ``True`` if semaphore can not be acquired immediately.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 321)

Unknown directive type "method".

```
.. method:: release()

   Release a semaphore, incrementing the internal counter by one.
   Can wake up a task waiting to acquire the semaphore.

   Unlike :class:`BoundedSemaphore`, :class:`Semaphore` allows
   making more ``release()`` calls than ``acquire()`` calls.
```

BoundedSemaphore

A bounded semaphore object. Not thread-safe.

Bounded Semaphore is a version of :class:`Semaphore` that raises a :exc:`ValueError` in :meth:`~Semaphore.release` if it increases the internal counter above the initial *value*.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 337); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 337); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 337); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 341)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.10
   Removed the *loop* parameter.
```

Barrier

A barrier object. Not thread-safe.

A barrier is a simple synchronization primitive that allows to block until *parties* number of tasks are waiting on it. Tasks can wait on the :meth:`~Barrier.wait` method and would be blocked until the specified number of tasks end up waiting on :meth:`~Barrier.wait`. At that point all of the waiting tasks would unblock simultaneously.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 352); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 352); *backlink*

Unknown interpreted text role "meth".

:keyword:`async with` can be used as an alternative to awaiting on :meth:`~Barrier.wait`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 358); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 358); *backlink*

Unknown interpreted text role "meth".

The barrier can be reused any number of times.

Example:

```
async def example_barrier():
   # barrier with 3 parties
   b = asyncio.Barrier(3)

   # create 2 new waiting tasks
   asyncio.create_task(b.wait())
   asyncio.create_task(b.wait())

   await asyncio.sleep(0)
   print(b)

   # The third .wait() call passes the barrier
   await b.wait()
   print(b)
   print("barrier passed")

   await asyncio.sleep(0)
   print(b)

asyncio.run(example_barrier())
```

Result of this example is:

```
<asyncio.locks.Barrier object at 0x... [filling, waiters:2/3]>
<asyncio.locks.Barrier object at 0x... [draining, waiters:0/3]>
barrier passed
<asyncio.locks.Barrier object at 0x... [filling, waiters:0/3]>
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 395)

Unknown directive type "versionadded".

```
.. versionadded:: 3.11
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 397)

Unknown directive type "coroutinemethod".

```
.. coroutinemethod:: wait()

   Pass the barrier. When all the tasks party to the barrier have called
   this function, they are all unblocked simultaneously.

   When a waiting or blocked task in the barrier is cancelled,
   this task exits the barrier which stays in the same state.
   If the state of the barrier is "filling", the number of waiting task
   decreases by 1.

   The return value is an integer in the range of 0 to ``parties-1``, different
   for each task. This can be used to select a task to do some special
   housekeeping, e.g.::

      ...
      async with barrier as position:
         if position == 0:
            # Only one task print this
            print('End of *draining phasis*')

   This method may raise a :class:`BrokenBarrierError` exception if the
   barrier is broken or reset while a task is waiting.
   It could raise a :exc:`CancelledError` if a task is cancelled.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 421)

Unknown directive type "coroutinemethod".

```
.. coroutinemethod:: reset()

   Return the barrier to the default, empty state.  Any tasks waiting on it
   will receive the :class:`BrokenBarrierError` exception.

   If a barrier is broken it may be better to just leave it and create a new one.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 428)

Unknown directive type "coroutinemethod".

```
.. coroutinemethod:: abort()

   Put the barrier into a broken state.  This causes any active or future
   calls to :meth:`wait` to fail with the :class:`BrokenBarrierError`.
   Use this for example if one of the taks needs to abort, to avoid infinite
   waiting tasks.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 435)

Unknown directive type "attribute".

```
.. attribute:: parties

   The number of tasks required to pass the barrier.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 439)

Unknown directive type "attribute".

```
.. attribute:: n_waiting

   The number of tasks currently waiting in the barrier while filling.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 443)

Unknown directive type "attribute".

```
.. attribute:: broken

   A boolean that is ``True`` if the barrier is in the broken state.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 448)

Unknown directive type "exception".

```
.. exception:: BrokenBarrierError

   This exception, a subclass of :exc:`RuntimeError`, is raised when the
   :class:`Barrier` object is reset or broken.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-sync.rst, line 456)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.9

   Acquiring a lock using ``await lock`` or ``yield from lock`` and/or
   :keyword:`with` statement (``with await lock``, ``with (yield from
   lock)``) was removed.  Use ``async with lock`` instead.
```