Subprocesses

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

Unknown directive type "currentmodule".

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

Source code: :source:`Lib/asyncio/subprocess.py`, :source:`Lib/asyncio/base_subprocess.py`

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

Unknown interpreted text role "source".

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

Unknown interpreted text role "source".

This section describes high-level async/await asyncio APIs to create and manage subprocesses.

Here's an example of how asyncio can run a shell command and obtain its result:

```
import asyncio

async def run(cmd):
    proc = await asyncio.create_subprocess_shell(
        cmd,
        stdout=asyncio.subprocess.PIPE,
        stderr=asyncio.subprocess.PIPE)

    stdout, stderr = await proc.communicate()

    print(f'[{cmd!r} exited with {proc.returncode}]')
    if stdout:
        print(f'[stdout]\n{stdout.decode()}')
    if stderr:
        print(f'[stderr]\n{stderr.decode()}')

asyncio.run(run('ls /zzz'))
```

will print:

```
['ls /zzz' exited with 1]
[stderr]
ls: /zzz: No such file or directory
```

Because all asyncio subprocess functions are asynchronous and asyncio provides many tools to work with such functions, it is easy to execute and monitor multiple subprocesses in parallel. It is indeed trivial to modify the above example to run several commands simultaneously:

```
async def main():
    await asyncio.gather(
        run('ls /zzz'),
        run('sleep 1; echo "hello"'))

asyncio.run(main())
```

See also the Examples subsection.

Creating Subprocesses

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

Unknown directive type "coroutinefunction".

```
.. coroutinefunction:: create_subprocess_exec(program, *args, stdin=None, \
```

```
                          stdout=None, stderr=None, limit=None, **kwds)

   Create a subprocess.

   The *limit* argument sets the buffer limit for :class:`StreamReader`
   wrappers for :attr:`Process.stdout` and :attr:`Process.stderr`
   (if :attr:`subprocess.PIPE` is passed to *stdout* and *stderr* arguments).

   Return a :class:`~asyncio.subprocess.Process` instance.

   See the documentation of :meth:`loop.subprocess_exec` for other
   parameters.

   .. 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-subprocess.rst, line 82)

Unknown directive type "coroutinefunction".

```
.. coroutinefunction:: create_subprocess_shell(cmd, stdin=None, \
                          stdout=None, stderr=None, limit=None, **kwds)

   Run the *cmd* shell command.

   The *limit* argument sets the buffer limit for :class:`StreamReader`
   wrappers for :attr:`Process.stdout` and :attr:`Process.stderr`
   (if :attr:`subprocess.PIPE` is passed to *stdout* and *stderr* arguments).

   Return a :class:`~asyncio.subprocess.Process` instance.

   See the documentation of :meth:`loop.subprocess_shell` for other
   parameters.

   .. important::

      It is the application's responsibility to ensure that all whitespace and
      special characters are quoted appropriately to avoid `shell injection
      <https://en.wikipedia.org/wiki/Shell_injection#Shell_injection>`_
      vulnerabilities. The :func:`shlex.quote` function can be used to properly
      escape whitespace and special shell characters in strings that are going
      to be used to construct shell commands.

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

Note

Subprocesses are available for Windows if a :class:`ProactorEventLoop` is used. See :ref:`Subprocess Support on Windows <asyncio-windows-subprocess>` for details.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-subprocess.rst, line 110); *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-subprocess.rst, line 110); *backlink*

Unknown interpreted text role "ref".

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

Unknown directive type "seealso".

```
.. seealso::
```

```
asyncio also has the following *low-level* APIs to work with subprocesses:
:meth:`loop.subprocess_exec`, :meth:`loop.subprocess_shell`,
:meth:`loop.connect_read_pipe`, :meth:`loop.connect_write_pipe`,
as well as the :ref:`Subprocess Transports <asyncio-subprocess-transports>`
and :ref:`Subprocess Protocols <asyncio-subprocess-protocols>`.
```

Constants

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

Unknown directive type "data".

```
.. data:: asyncio.subprocess.PIPE

   Can be passed to the *stdin*, *stdout* or *stderr* parameters.

   If *PIPE* is passed to *stdin* argument, the
   :attr:`Process.stdin <asyncio.subprocess.Process.stdin>` attribute
   will point to a :class:`StreamWriter` instance.

   If *PIPE* is passed to *stdout* or *stderr* arguments, the
   :attr:`Process.stdout <asyncio.subprocess.Process.stdout>` and
   :attr:`Process.stderr <asyncio.subprocess.Process.stderr>`
   attributes will point to :class:`StreamReader` instances.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-subprocess.rst, line 139)

Unknown directive type "data".

```
.. data:: asyncio.subprocess.STDOUT

   Special value that can be used as the *stderr* argument and indicates
   that standard error should be redirected into standard output.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-subprocess.rst, line 144)

Unknown directive type "data".

```
.. data:: asyncio.subprocess.DEVNULL

   Special value that can be used as the *stdin*, *stdout* or *stderr* argument
   to process creation functions.  It indicates that the special file
   :data:`os.devnull` will be used for the corresponding subprocess stream.
```

Interacting with Subprocesses

Both :func:`create_subprocess_exec` and :func:`create_subprocess_shell` functions return instances of the *Process* class. *Process* is a high-level wrapper that allows communicating with subprocesses and watching for their completion.

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

Unknown interpreted text role "func".

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

Unknown interpreted text role "func".

An object that wraps OS processes created by the :func:`create_subprocess_exec` and :func:`create_subprocess_shell` functions.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-

main\Doc\library\(cpython-main)(Doc)(library)asyncio-subprocess.rst, line 161); *backlink*

Unknown interpreted text role "func".

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

Unknown interpreted text role "func".

This class is designed to have a similar API to the :class:`subprocess.Popen` class, but there are some notable differences:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-subprocess.rst, line 165); *backlink*

Unknown interpreted text role "class".

- unlike Popen, Process instances do not have an equivalent to the :meth:`~subprocess.Popen.poll` method;

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-subprocess.rst, line 169); *backlink*

  Unknown interpreted text role "meth".

- the :meth:`~asyncio.subprocess.Process.communicate` and :meth:`~asyncio.subprocess.Process.wait` methods don't have a *timeout* parameter: use the :func:`wait_for` function;

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-subprocess.rst, line 172); *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-subprocess.rst, line 172); *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-subprocess.rst, line 172); *backlink*

  Unknown interpreted text role "func".

- the :meth:`Process.wait() <asyncio.subprocess.Process.wait>` method is asynchronous, whereas :meth:`subprocess.Popen.wait` method is implemented as a blocking busy loop;

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-subprocess.rst, line 176); *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-subprocess.rst, line 176); *backlink*

  Unknown interpreted text role "meth".

- the *universal_newlines* parameter is not supported.

This class is :ref:`not thread safe <asyncio-multithreading>`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-subprocess.rst, line 182); *backlink*

Unknown interpreted text role "ref".

See also the :ref:`Subprocess and Threads <asyncio-subprocess-threads>` section.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-subprocess.rst, line 184); *backlink*

Unknown interpreted text role "ref".

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

Unknown directive type "coroutinemethod".

```
.. coroutinemethod:: wait()

   Wait for the child process to terminate.

   Set and return the :attr:`returncode` attribute.

   .. note::

      This method can deadlock when using ``stdout=PIPE`` or
      ``stderr=PIPE`` and the child process generates so much output
      that it blocks waiting for the OS pipe buffer to accept
      more data. Use the :meth:`communicate` method when using pipes
      to avoid this condition.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-subprocess.rst, line 201)

Unknown directive type "coroutinemethod".

```
.. coroutinemethod:: communicate(input=None)

   Interact with process:

   1. send data to *stdin* (if *input* is not ``None``);
   2. read data from *stdout* and *stderr*, until EOF is reached;
   3. wait for process to terminate.

   The optional *input* argument is the data (:class:`bytes` object)
   that will be sent to the child process.

   Return a tuple ``(stdout_data, stderr_data)``.

   If either :exc:`BrokenPipeError` or :exc:`ConnectionResetError`
   exception is raised when writing *input* into *stdin*, the
   exception is ignored.  This condition occurs when the process
   exits before all data are written into *stdin*.

   If it is desired to send data to the process' *stdin*,
   the process needs to be created with ``stdin=PIPE``.  Similarly,
   to get anything other than ``None`` in the result tuple, the
   process has to be created with ``stdout=PIPE`` and/or
   ``stderr=PIPE`` arguments.

   Note, that the data read is buffered in memory, so do not use
   this method if the data size is large or unlimited.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-subprocess.rst, line 228)

Unknown directive type "method".

```
.. method:: send_signal(signal)

   Sends the signal *signal* to the child process.

   .. note::

      On Windows, :py:data:`SIGTERM` is an alias for :meth:`terminate`.
      ``CTRL_C_EVENT`` and ``CTRL_BREAK_EVENT`` can be sent to processes
      started with a *creationflags* parameter which includes
```

```
``CREATE_NEW_PROCESS_GROUP``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-subprocess.rst, line 239)

Unknown directive type "method".

```
.. method:: terminate()

   Stop the child process.

   On POSIX systems this method sends :py:data:`signal.SIGTERM` to the
   child process.

   On Windows the Win32 API function :c:func:`TerminateProcess` is
   called to stop the child process.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-subprocess.rst, line 249)

Unknown directive type "method".

```
.. method:: kill()

   Kill the child process.

   On POSIX systems this method sends :py:data:`SIGKILL` to the child
   process.

   On Windows this method is an alias for :meth:`terminate`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-subprocess.rst, line 258)

Unknown directive type "attribute".

```
.. attribute:: stdin

   Standard input stream (:class:`StreamWriter`) or ``None``
   if the process was created with ``stdin=None``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-subprocess.rst, line 263)

Unknown directive type "attribute".

```
.. attribute:: stdout

   Standard output stream (:class:`StreamReader`) or ``None``
   if the process was created with ``stdout=None``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-subprocess.rst, line 268)

Unknown directive type "attribute".

```
.. attribute:: stderr

   Standard error stream (:class:`StreamReader`) or ``None``
   if the process was created with ``stderr=None``.
```

Warning

Use the :meth:`communicate` method rather than :attr:`process.stdin.write() <stdin>`, :attr:`await process.stdout.read() <stdout>` or :attr:`await process.stderr.read <stderr>`. This avoids deadlocks due to streams pausing reading or writing and blocking the child process.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-subprocess.rst, line 275); *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-subprocess.rst, line 275); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-subprocess.rst, line 275); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-subprocess.rst, line 275); *backlink*

Unknown interpreted text role "attr".

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

Unknown directive type "attribute".

```
.. attribute:: pid

   Process identification number (PID).

   Note that for processes created by the :func:`create_subprocess_shell`
   function, this attribute is the PID of the spawned shell.
```

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

Unknown directive type "attribute".

```
.. attribute:: returncode

   Return code of the process when it exits.

   A ``None`` value indicates that the process has not terminated yet.

   A negative value ``-N`` indicates that the child was terminated
   by signal ``N`` (POSIX only).
```

Subprocess and Threads

Standard asyncio event loop supports running subprocesses from different threads by default.

On Windows subprocesses are provided by :class:`ProactorEventLoop` only (default), :class:`SelectorEventLoop` has no subprocess support.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-subprocess.rst, line 307); *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-subprocess.rst, line 307); *backlink*

Unknown interpreted text role "class".

On UNIX *child watchers* are used for subprocess finish waiting, see :ref:`asyncio-watchers` for more info.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-subprocess.rst, line 310); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-subprocess.rst, line 314)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.8

   UNIX switched to use :class:`ThreadedChildWatcher` for spawning subprocesses from
   different threads without any limitation.

   Spawning a subprocess with *inactive* current child watcher raises
   :exc:`RuntimeError`.
```

Note that alternative event loop implementations might have own limitations; please refer to their documentation.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-subprocess.rst, line 325)

Unknown directive type "seealso".

```
.. seealso::

   The :ref:`Concurrency and multithreading in asyncio
   <asyncio-multithreading>` section.
```

Examples

An example using the :class:`~asyncio.subprocess.Process` class to control a subprocess and the :class:`StreamReader` class to read from its standard output.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-subprocess.rst, line 334); *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-subprocess.rst, line 334); *backlink*

Unknown interpreted text role "class".

The subprocess is created by the :func:`create_subprocess_exec` function:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-subprocess.rst, line 340); *backlink*

Unknown interpreted text role "func".

```
import asyncio
import sys

async def get_date():
    code = 'import datetime; print(datetime.datetime.now())'

    # Create the subprocess; redirect the standard output
    # into a pipe.
    proc = await asyncio.create_subprocess_exec(
        sys.executable, '-c', code,
        stdout=asyncio.subprocess.PIPE)

    # Read one line of output.
    data = await proc.stdout.readline()
    line = data.decode('ascii').rstrip()

    # Wait for the subprocess exit.
    await proc.wait()
```

```
        return line

    date = asyncio.run(get_date())
    print(f"Current date: {date}")
```

See also the :ref:`same example <asyncio_example_subprocess_proto>` written using low-level APIs.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)asyncio-subprocess.rst, line 367); *backlink*

Unknown interpreted text role "ref".