:mod:`subprocess` --- Subprocess management

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

Unknown directive type "module".

```
.. module:: subprocess
   :synopsis: Subprocess management.
```

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

Unknown directive type "moduleauthor".

```
.. moduleauthor:: Peter Ã…strand <astrand@lysator.liu.se>
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 8)

Unknown directive type "sectionauthor".

```
.. sectionauthor:: Peter Ã…strand <astrand@lysator.liu.se>
```

Source code: :source:`Lib/subprocess.py`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 10); *backlink*

Unknown interpreted text role "source".

The :mod:`subprocess` module allows you to spawn new processes, connect to their input/output/error pipes, and obtain their return codes. This module intends to replace several older modules and functions:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 14); *backlink*

Unknown interpreted text role "mod".

```
os.system
os.spawn*
```

Information about how the :mod:`subprocess` module can be used to replace these modules and functions can be found in the following sections.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 21); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 24)

Unknown directive type "seealso".

```
.. seealso::

   :pep:`324` -- PEP proposing the subprocess module
```

Using the :mod:`subprocess` Module

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

Unknown interpreted text role "mod".

The recommended approach to invoking subprocesses is to use the :func:`run` function for all use cases it can handle. For more advanced use cases, the underlying :class:`Popen` interface can be used directly.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 32); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 32); *backlink*

Unknown interpreted text role "class".

The :func:`run` function was added in Python 3.5; if you need to retain compatibility with older versions, see the :ref:`call-function-trio` section.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 36); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 36); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 40)

Unknown directive type "function".

```
.. function:: run(args, *, stdin=None, input=None, stdout=None, stderr=None,\
                  capture_output=False, shell=False, cwd=None, timeout=None, \
                  check=False, encoding=None, errors=None, text=None, env=None, \
                  universal_newlines=None, **other_popen_kwargs)

   Run the command described by *args*.  Wait for command to complete, then
   return a :class:`CompletedProcess` instance.

   The arguments shown above are merely the most common ones, described below
   in :ref:`frequently-used-arguments` (hence the use of keyword-only notation
   in the abbreviated signature). The full function signature is largely the
   same as that of the :class:`Popen` constructor - most of the arguments to
   this function are passed through to that interface. (*timeout*,  *input*,
   *check*, and *capture_output* are not.)

   If *capture_output* is true, stdout and stderr will be captured.
   When used, the internal :class:`Popen` object is automatically created with
   ``stdout=PIPE`` and ``stderr=PIPE``. The *stdout* and *stderr* arguments may
   not be supplied at the same time as *capture_output*.  If you wish to capture
   and combine both streams into one, use ``stdout=PIPE`` and ``stderr=STDOUT``
   instead of *capture_output*.

   The *timeout* argument is passed to :meth:`Popen.communicate`. If the timeout
   expires, the child process will be killed and waited for.  The
   :exc:`TimeoutExpired` exception will be re-raised after the child process
   has terminated.

   The *input* argument is passed to :meth:`Popen.communicate` and thus to the
   subprocess's stdin.  If used it must be a byte sequence, or a string if
   *encoding* or *errors* is specified or *text* is true.  When
   used, the internal :class:`Popen` object is automatically created with
   ``stdin=PIPE``, and the *stdin* argument may not be used as well.

   If *check* is true, and the process exits with a non-zero exit code, a
   :exc:`CalledProcessError` exception will be raised. Attributes of that
   exception hold the arguments, the exit code, and stdout and stderr if they
   were captured.

   If *encoding* or *errors* are specified, or *text* is true,
   file objects for stdin, stdout and stderr are opened in text mode using the
   specified *encoding* and *errors* or the :class:`io.TextIOWrapper` default.
   The *universal_newlines* argument is equivalent  to *text* and is provided
   for backwards compatibility. By default, file objects are opened in binary mode.

   If *env* is not ``None``, it must be a mapping that defines the environment
   variables for the new process; these are used instead of the default
   behavior of inheriting the current process' environment. It is passed directly
   to :class:`Popen`.

   Examples::

      >>> subprocess.run(["ls", "-l"])  # doesn't capture output
      CompletedProcess(args=['ls', '-l'], returncode=0)

      >>> subprocess.run("exit 1", shell=True, check=True)
      Traceback (most recent call last):
        ...
      subprocess.CalledProcessError: Command 'exit 1' returned non-zero exit status 1

      >>> subprocess.run(["ls", "-l", "/dev/null"], capture_output=True)
      CompletedProcess(args=['ls', '-l', '/dev/null'], returncode=0,
      stdout=b'crw-rw-rw- 1 root root 1, 3 Jan 23 16:23 /dev/null\n', stderr=b'')
```

```
   .. versionadded:: 3.5

   .. versionchanged:: 3.6

      Added *encoding* and *errors* parameters

   .. versionchanged:: 3.7

      Added the *text* parameter, as a more understandable alias of *universal_newlines*.
      Added the *capture_output* parameter.
```

The return value from :func:`run`, representing a process that has finished.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 116); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 118)

Unknown directive type "attribute".

```
.. attribute:: args

   The arguments used to launch the process. This may be a list or a string.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 122)

Unknown directive type "attribute".

```
.. attribute:: returncode

   Exit status of the child process. Typically, an exit status of 0 indicates
   that it ran successfully.

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

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 130)

Unknown directive type "attribute".

```
.. attribute:: stdout

   Captured stdout from the child process. A bytes sequence, or a string if
   :func:`run` was called with an encoding, errors, or text=True.
   ``None`` if stdout was not captured.

   If you ran the process with ``stderr=subprocess.STDOUT``, stdout and
   stderr will be combined in this attribute, and :attr:`stderr` will be
   ``None``.
```

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

Unknown directive type "attribute".

```
.. attribute:: stderr

   Captured stderr from the child process. A bytes sequence, or a string if
   :func:`run` was called with an encoding, errors, or text=True.
   ``None`` if stderr was not captured.
```

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

Unknown directive type "method".

```
.. method:: check_returncode()

   If :attr:`returncode` is non-zero, raise a :exc:`CalledProcessError`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 150)

Unknown directive type "versionadded".

```
.. versionadded:: 3.5
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 152)

Unknown directive type "data".

```
.. data:: DEVNULL

   Special value that can be used as the *stdin*, *stdout* or *stderr* argument
   to :class:`Popen` and indicates that the special file :data:`os.devnull`
   will be used.

   .. versionadded:: 3.3
```

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

Unknown directive type "data".

```
.. data:: PIPE

   Special value that can be used as the *stdin*, *stdout* or *stderr* argument
   to :class:`Popen` and indicates that a pipe to the standard stream should be
   opened.  Most useful with :meth:`Popen.communicate`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 168)

Unknown directive type "data".

```
.. data:: STDOUT

   Special value that can be used as the *stderr* argument to :class:`Popen` and
   indicates that standard error should go into the same handle as standard
   output.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 175)

Unknown directive type "exception".

```
.. exception:: SubprocessError

    Base class for all other exceptions from this module.

    .. versionadded:: 3.3
```

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

Unknown directive type "exception".

```
.. exception:: TimeoutExpired

    Subclass of :exc:`SubprocessError`, raised when a timeout expires
    while waiting for a child process.

    .. attribute:: cmd

        Command that was used to spawn the child process.

    .. attribute:: timeout

        Timeout in seconds.

    .. attribute:: output

        Output of the child process if it was captured by :func:`run` or
        :func:`check_output`.  Otherwise, ``None``.

    .. attribute:: stdout

        Alias for output, for symmetry with :attr:`stderr`.

    .. attribute:: stderr

        Stderr output of the child process if it was captured by :func:`run`.
        Otherwise, ``None``.

    .. versionadded:: 3.3

    .. versionchanged:: 3.5
        *stdout* and *stderr* attributes added
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 214)

Unknown directive type "exception".

```
.. exception:: CalledProcessError

    Subclass of :exc:`SubprocessError`, raised when a process run by
    :func:`check_call` or :func:`check_output` returns a non-zero exit status.

    .. attribute:: returncode

        Exit status of the child process.  If the process exited due to a
        signal, this will be the negative signal number.

    .. attribute:: cmd

        Command that was used to spawn the child process.

    .. attribute:: output

        Output of the child process if it was captured by :func:`run` or
        :func:`check_output`.  Otherwise, ``None``.

    .. attribute:: stdout

        Alias for output, for symmetry with :attr:`stderr`.

    .. attribute:: stderr

        Stderr output of the child process if it was captured by :func:`run`.
        Otherwise, ``None``.

    .. versionchanged:: 3.5
        *stdout* and *stderr* attributes added
```

Frequently Used Arguments

To support a wide variety of use cases, the :class:`Popen` constructor (and the convenience functions) accept a large number of optional arguments. For most typical use cases, many of these arguments can be safely left at their default values. The arguments that are most commonly needed are:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 251); *backlink*

Unknown interpreted text role "class".

args is required for all calls and should be a string, or a sequence of program arguments. Providing a sequence of arguments is generally preferred, as it allows the module to take care of any required escaping and quoting of arguments (e.g. to permit spaces in file names). If passing a single string, either *shell* must be :const:`True` (see below) or else the string must simply name the program to be executed without specifying any arguments.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 256); *backlink*

Unknown interpreted text role "const".

stdin, *stdout* and *stderr* specify the executed program's standard input, standard output and standard error file handles, respectively. Valid values are :data:`PIPE`, :data:`DEVNULL`, an existing file descriptor (a positive integer), an existing file object with a valid file descriptor, and `None`. :data:`PIPE` indicates that a new pipe to the child should be created. :data:`DEVNULL` indicates that the special file :data:`os.devnull` will be used. With the default settings of `None`, no redirection will occur; the child's file handles will be inherited from the parent. Additionally, *stderr* can be :data:`STDOUT`, which indicates that the stderr data from the child process should be captured into the same file handle as for *stdout*.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 264); *backlink*

Unknown interpreted text role "data".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 264); *backlink*

Unknown interpreted text role "data".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 264); *backlink*

Unknown interpreted text role "data".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 264); *backlink*

Unknown interpreted text role "data".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 264); *backlink*

Unknown interpreted text role "data".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 264); *backlink*

Unknown interpreted text role "data".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 276)

Unknown directive type "index".

```
.. index::
   single: universal newlines; subprocess module
```

If *encoding* or *errors* are specified, or *text* (also known as *universal_newlines*) is true, the file objects *stdin*, *stdout* and *stderr* will be opened in text mode using the *encoding* and *errors* specified in the call or the defaults for :class:`io.TextIOWrapper`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 279); *backlink*

Unknown interpreted text role "class".

For *stdin*, line ending characters `'\n'` in the input will be converted to the default line separator :data:`os.linesep`. For *stdout* and *stderr*, all line endings in the output will be converted to `'\n'`. For more information see the documentation of the :class:`io.TextIOWrapper` class when the *newline* argument to its constructor is `None`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 285); *backlink*

Unknown interpreted text role "data".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 285); *backlink*

Unknown interpreted text role "class".

If text mode is not used, *stdin*, *stdout* and *stderr* will be opened as binary streams. No encoding or line ending conversion is performed.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 294)

Unknown directive type "versionadded".

```
.. versionadded:: 3.6
   Added *encoding* and *errors* parameters.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 297)

Unknown directive type "versionadded".

```
.. versionadded:: 3.7
   Added the *text* parameter as an alias for *universal_newlines*.
```

Note

The newlines attribute of the file objects :attr:`Popen.stdin`, :attr:`Popen.stdout` and :attr:`Popen.stderr` are not updated by the :meth:`Popen.communicate` method.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 302); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 302); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 302); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 302); *backlink*

Unknown interpreted text role "meth".

If *shell* is `True`, the specified command will be executed through the shell. This can be useful if you are using Python primarily for the enhanced control flow it offers over most system shells and still want convenient access to other shell features such as shell pipes, filename wildcards, environment variable expansion, and expansion of ~ to a user's home directory. However, note that Python itself offers implementations of many shell-like features (in particular, :mod:`glob`, :mod:`fnmatch`, :func:`os.walk`, :func:`os.path.expandvars`, :func:`os.path.expanduser`, and :mod:`shutil`).

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 306); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 306); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 306); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 306); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 306); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 306); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 316)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.3
   When *universal_newlines* is ``True``, the class uses the encoding
   :func:`locale.getpreferredencoding(False) <locale.getpreferredencoding>`
   instead of ``locale.getpreferredencoding()``.  See the
   :class:`io.TextIOWrapper` class for more information on this change.
```

Note

Read the Security Considerations section before using `shell=True`.

These options, along with all of the other options, are described in more detail in the :class:`Popen` constructor documentation.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 326); *backlink*

Unknown interpreted text role "class".

Popen Constructor

The underlying process creation and management in this module is handled by the :class:`Popen` class. It offers a lot of flexibility so that developers are able to handle the less common cases not covered by the convenience functions.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 333); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 339)

Invalid class attribute value for "class" directive: "Popen(args, bufsize=-1, executable=None, stdin=None, stdout=None, \ stderr=None, preexec_fn=None, close_fds=True, shell=False, \ cwd=None, env=None, universal_newlines=None, \ startupinfo=None, creationflags=0, restore_signals=True, \ start_new_session=False, pass_fds=(), *, group=None, \ extra_groups=None, user=None, umask=-1, \ encoding=None, errors=None, text=None, pipesize=-1)".

```
.. class:: Popen(args, bufsize=-1, executable=None, stdin=None, stdout=None, \
                 stderr=None, preexec_fn=None, close_fds=True, shell=False, \
                 cwd=None, env=None, universal_newlines=None, \
                 startupinfo=None, creationflags=0, restore_signals=True, \
                 start_new_session=False, pass_fds=(), *, group=None, \
                 extra_groups=None, user=None, umask=-1, \
                 encoding=None, errors=None, text=None, pipesize=-1)

   Execute a child program in a new process.  On POSIX, the class uses
   :meth:`os.execvpe`-like behavior to execute the child program.  On Windows,
   the class uses the Windows ``CreateProcess()`` function.  The arguments to
   :class:`Popen` are as follows.

   *args* should be a sequence of program arguments or else a single string
   or :term:`path-like object`.
   By default, the program to execute is the first item in *args* if *args* is
   a sequence.  If *args* is a string, the interpretation is
   platform-dependent and described below.  See the *shell* and *executable*
   arguments for additional differences from the default behavior.  Unless
   otherwise stated, it is recommended to pass *args* as a sequence.

   .. warning::

      For maximum reliability, use a fully-qualified path for the executable.
      To search for an unqualified name on :envvar:`PATH`, use
      :meth:`shutil.which`. On all platforms, passing :data:`sys.executable`
      is the recommended way to launch the current Python interpreter again,
      and use the ``-m`` command-line format to launch an installed module.

      Resolving the path of *executable* (or the first item of *args*) is
      platform dependent. For POSIX, see :meth:`os.execvpe`, and note that
      when resolving or searching for the executable path, *cwd* overrides the
      current working directory and *env* can override the ``PATH``
      environment variable. For Windows, see the documentation of the
      ``lpApplicationName`` and ``lpCommandLine`` parameters of WinAPI
      ``CreateProcess``, and note that when resolving or searching for the
      executable path with ``shell=False``, *cwd* does not override the
      current working directory and *env* cannot override the ``PATH``
      environment variable. Using a full path avoids all of these variations.

   An example of passing some arguments to an external program
   as a sequence is::

     Popen(["/usr/bin/git", "commit", "-m", "Fixes a bug."])

   On POSIX, if *args* is a string, the string is interpreted as the name or
   path of the program to execute.  However, this can only be done if not
   passing arguments to the program.

   .. note::

      It may not be obvious how to break a shell command into a sequence of arguments,
      especially in complex cases. :meth:`shlex.split` can illustrate how to
      determine the correct tokenization for *args*::

         >>> import shlex, subprocess
         >>> command_line = input()
         /bin/vikings -input eggs.txt -output "spam spam.txt" -cmd "echo '$MONEY'"
         >>> args = shlex.split(command_line)
```

```
      >>> print(args)
      ['/bin/vikings', '-input', 'eggs.txt', '-output', 'spam spam.txt', '-cmd', "echo '$MONEY'"]
      >>> p = subprocess.Popen(args) # Success!

   Note in particular that options (such as *-input*) and arguments (such
   as *eggs.txt*) that are separated by whitespace in the shell go in separate
   list elements, while arguments that need quoting or backslash escaping when
   used in the shell (such as filenames containing spaces or the *echo* command
   shown above) are single list elements.

On Windows, if *args* is a sequence, it will be converted to a string in a
manner described in :ref:`converting-argument-sequence`.  This is because
the underlying ``CreateProcess()`` operates on strings.

.. versionchanged:: 3.6
   *args* parameter accepts a :term:`path-like object` if *shell* is
   ``False`` and a sequence containing path-like objects on POSIX.

.. versionchanged:: 3.8
   *args* parameter accepts a :term:`path-like object` if *shell* is
   ``False`` and a sequence containing bytes and path-like objects
   on Windows.

The *shell* argument (which defaults to ``False``) specifies whether to use
the shell as the program to execute.  If *shell* is ``True``, it is
recommended to pass *args* as a string rather than as a sequence.

On POSIX with ``shell=True``, the shell defaults to :file:`/bin/sh`.  If
*args* is a string, the string specifies the command
to execute through the shell.  This means that the string must be
formatted exactly as it would be when typed at the shell prompt.  This
includes, for example, quoting or backslash escaping filenames with spaces in
them.  If *args* is a sequence, the first item specifies the command string, and
any additional items will be treated as additional arguments to the shell
itself.  That is to say, :class:`Popen` does the equivalent of::

   Popen(['/bin/sh', '-c', args[0], args[1], ...])

On Windows with ``shell=True``, the :envvar:`COMSPEC` environment variable
specifies the default shell.  The only time you need to specify
``shell=True`` on Windows is when the command you wish to execute is built
into the shell (e.g. :command:`dir` or :command:`copy`).  You do not need
``shell=True`` to run a batch file or console-based executable.

.. note::

   Read the `Security Considerations`_ section before using ``shell=True``.

*bufsize* will be supplied as the corresponding argument to the
:func:`open` function when creating the stdin/stdout/stderr pipe
file objects:

- :const:`0` means unbuffered (read and write are one
  system call and can return short)
- :const:`1` means line buffered
  (only usable if ``universal_newlines=True`` i.e., in a text mode)
- any other positive value means use a buffer of approximately that
  size
- negative bufsize (the default) means the system default of
  io.DEFAULT_BUFFER_SIZE will be used.

.. versionchanged:: 3.3.1
   *bufsize* now defaults to -1 to enable buffering by default to match the
   behavior that most code expects.  In versions prior to Python 3.2.4 and
   3.3.1 it incorrectly defaulted to :const:`0` which was unbuffered
   and allowed short reads.  This was unintentional and did not match the
   behavior of Python 2 as most code expected.

The *executable* argument specifies a replacement program to execute.   It
is very seldom needed.  When ``shell=False``, *executable* replaces the
program to execute specified by *args*.  However, the original *args* is
still passed to the program.  Most programs treat the program specified
by *args* as the command name, which can then be different from the program
actually executed.  On POSIX, the *args* name
becomes the display name for the executable in utilities such as
:program:`ps`.  If ``shell=True``, on POSIX the *executable* argument
specifies a replacement shell for the default :file:`/bin/sh`.

.. versionchanged:: 3.6
   *executable* parameter accepts a :term:`path-like object` on POSIX.

.. versionchanged:: 3.8
   *executable* parameter accepts a bytes and :term:`path-like object`
   on Windows.

*stdin*, *stdout* and *stderr* specify the executed program's standard input,
standard output and standard error file handles, respectively.  Valid values
are :data:`PIPE`, :data:`DEVNULL`, an existing file descriptor (a positive
integer), an existing :term:`file object` with a valid file descriptor,
and ``None``.  :data:`PIPE` indicates that a new pipe to the child should
be created.  :data:`DEVNULL` indicates that the special file
:data:`os.devnull` will be used. With the default settings of ``None``,
no redirection will occur; the child's file handles will be inherited from
the parent.  Additionally, *stderr* can be :data:`STDOUT`, which indicates
that the stderr data from the applications should be captured into the same
file handle as for stdout.
```

```
If *preexec_fn* is set to a callable object, this object will be called in the
child process just before the child is executed.
(POSIX only)

.. warning::

   The *preexec_fn* parameter is not safe to use in the presence of threads
   in your application.  The child process could deadlock before exec is
   called.
   If you must use it, keep it trivial!  Minimize the number of libraries
   you call into.

.. note::

   If you need to modify the environment for the child use the *env*
   parameter rather than doing it in a *preexec_fn*.
   The *start_new_session* parameter can take the place of a previously
   common use of *preexec_fn* to call os.setsid() in the child.

.. versionchanged:: 3.8

   The *preexec_fn* parameter is no longer supported in subinterpreters.
   The use of the parameter in a subinterpreter raises
   :exc:`RuntimeError`. The new restriction may affect applications that
   are deployed in mod_wsgi, uWSGI, and other embedded environments.

If *close_fds* is true, all file descriptors except :const:`0`, :const:`1` and
:const:`2` will be closed before the child process is executed.  Otherwise
when *close_fds* is false, file descriptors obey their inheritable flag
as described in :ref:`fd_inheritance`.

On Windows, if *close_fds* is true then no handles will be inherited by the
child process unless explicitly passed in the ``handle_list`` element of
:attr:`STARTUPINFO.lpAttributeList`, or by standard handle redirection.

.. versionchanged:: 3.2
   The default for *close_fds* was changed from :const:`False` to
   what is described above.

.. versionchanged:: 3.7
   On Windows the default for *close_fds* was changed from :const:`False` to
   :const:`True` when redirecting the standard handles. It's now possible to
   set *close_fds* to :const:`True` when redirecting the standard handles.

*pass_fds* is an optional sequence of file descriptors to keep open
between the parent and child.  Providing any *pass_fds* forces
*close_fds* to be :const:`True`.  (POSIX only)

.. versionchanged:: 3.2
   The *pass_fds* parameter was added.

If *cwd* is not ``None``, the function changes the working directory to
*cwd* before executing the child.  *cwd* can be a string, bytes or
:term:`path-like <path-like object>` object.  On POSIX, the function
looks for *executable* (or for the first item in *args*) relative to *cwd*
if the executable path is a relative path.

.. versionchanged:: 3.6
   *cwd* parameter accepts a :term:`path-like object` on POSIX.

.. versionchanged:: 3.7
   *cwd* parameter accepts a :term:`path-like object` on Windows.

.. versionchanged:: 3.8
   *cwd* parameter accepts a bytes object on Windows.

If *restore_signals* is true (the default) all signals that Python has set to
SIG_IGN are restored to SIG_DFL in the child process before the exec.
Currently this includes the SIGPIPE, SIGXFZ and SIGXFSZ signals.
(POSIX only)

.. versionchanged:: 3.2
   *restore_signals* was added.

If *start_new_session* is true the setsid() system call will be made in the
child process prior to the execution of the subprocess.  (POSIX only)

.. versionchanged:: 3.2
   *start_new_session* was added.

If *group* is not ``None``, the setregid() system call will be made in the
child process prior to the execution of the subprocess. If the provided
value is a string, it will be looked up via :func:`grp.getgrnam()` and
the value in ``gr_gid`` will be used. If the value is an integer, it
will be passed verbatim. (POSIX only)

.. availability:: POSIX
.. versionadded:: 3.9

If *extra_groups* is not ``None``, the setgroups() system call will be
made in the child process prior to the execution of the subprocess.
Strings provided in *extra_groups* will be looked up via
:func:`grp.getgrnam()` and the values in ``gr_gid`` will be used.
Integer values will be passed verbatim. (POSIX only)

.. availability:: POSIX
.. versionadded:: 3.9
```

```
If *user* is not ``None``, the setreuid() system call will be made in the
child process prior to the execution of the subprocess. If the provided
value is a string, it will be looked up via :func:`pwd.getpwnam()` and
the value in ``pw_uid`` will be used. If the value is an integer, it will
be passed verbatim. (POSIX only)

.. availability:: POSIX
.. versionadded:: 3.9

If *umask* is not negative, the umask() system call will be made in the
child process prior to the execution of the subprocess.

.. availability:: POSIX
.. versionadded:: 3.9

If *env* is not ``None``, it must be a mapping that defines the environment
variables for the new process; these are used instead of the default
behavior of inheriting the current process' environment.

.. note::

   If specified, *env* must provide any variables required for the program to
   execute.  On Windows, in order to run a `side-by-side assembly`_ the
   specified *env* **must** include a valid :envvar:`SystemRoot`.

.. _side-by-side assembly: https://en.wikipedia.org/wiki/Side-by-Side_Assembly

If *encoding* or *errors* are specified, or *text* is true, the file objects
*stdin*, *stdout* and *stderr* are opened in text mode with the specified
encoding and *errors*, as described above in :ref:`frequently-used-arguments`.
The *universal_newlines* argument is equivalent  to *text* and is provided
for backwards compatibility. By default, file objects are opened in binary mode.

.. versionadded:: 3.6
   *encoding* and *errors* were added.

.. versionadded:: 3.7
   *text* was added as a more readable alias for *universal_newlines*.

If given, *startupinfo* will be a :class:`STARTUPINFO` object, which is
passed to the underlying ``CreateProcess`` function.
*creationflags*, if given, can be one or more of the following flags:

   * :data:`CREATE_NEW_CONSOLE`
   * :data:`CREATE_NEW_PROCESS_GROUP`
   * :data:`ABOVE_NORMAL_PRIORITY_CLASS`
   * :data:`BELOW_NORMAL_PRIORITY_CLASS`
   * :data:`HIGH_PRIORITY_CLASS`
   * :data:`IDLE_PRIORITY_CLASS`
   * :data:`NORMAL_PRIORITY_CLASS`
   * :data:`REALTIME_PRIORITY_CLASS`
   * :data:`CREATE_NO_WINDOW`
   * :data:`DETACHED_PROCESS`
   * :data:`CREATE_DEFAULT_ERROR_MODE`
   * :data:`CREATE_BREAKAWAY_FROM_JOB`

*pipesize* can be used to change the size of the pipe when
:data:`PIPE` is used for *stdin*, *stdout* or *stderr*. The size of the pipe
is only changed on platforms that support this (only Linux at this time of
writing). Other platforms will ignore this parameter.

.. versionadded:: 3.10
   The ``pipesize`` parameter was added.

Popen objects are supported as context managers via the :keyword:`with` statement:
on exit, standard file descriptors are closed, and the process is waited for.
::

   with Popen(["ifconfig"], stdout=PIPE) as proc:
       log.write(proc.stdout.read())

.. audit-event:: subprocess.Popen executable,args,cwd,env subprocess.Popen

   Popen and the other functions in this module that use it raise an
   :ref:`auditing event <auditing>` ``subprocess.Popen`` with arguments
   ``executable``, ``args``, ``cwd``, and ``env``. The value for ``args``
   may be a single string or a list of strings, depending on platform.

.. versionchanged:: 3.2
   Added context manager support.

.. versionchanged:: 3.6
   Popen destructor now emits a :exc:`ResourceWarning` warning if the child
   process is still running.

.. versionchanged:: 3.8
   Popen can use :func:`os.posix_spawn` in some cases for better
   performance. On Windows Subsystem for Linux and QEMU User Emulation,
   Popen constructor using :func:`os.posix_spawn` no longer raise an
   exception on errors like missing program, but the child process fails
   with a non-zero :attr:`~Popen.returncode`.
```

Exceptions

Exceptions raised in the child process, before the new program has started to execute, will be re-raised in the parent.

The most common exception raised is :exc:`OSError`. This occurs, for example, when trying to execute a non-existent file. Applications should prepare for :exc:`OSError` exceptions. Note that, when `shell=True`, :exc:`OSError` will be raised by the child only if the selected shell itself was not found. To determine if the shell failed to find the requested application, it is necessary to check the return code or output from the subprocess.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst`, line 692); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst`, line 692); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst`, line 692); *backlink*

Unknown interpreted text role "exc".

A :exc:`ValueError` will be raised if :class:`Popen` is called with invalid arguments.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst`, line 699); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst`, line 699); *backlink*

Unknown interpreted text role "class".

:func:`check_call` and :func:`check_output` will raise :exc:`CalledProcessError` if the called process returns a non-zero return code.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst`, line 702); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst`, line 702); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst`, line 702); *backlink*

Unknown interpreted text role "exc".

All of the functions and methods that accept a *timeout* parameter, such as :func:`call` and :meth:`Popen.communicate` will raise :exc:`TimeoutExpired` if the timeout expires before the process exits.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst`, line 706); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst`, line 706); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst`, line 706); *backlink*

Unknown interpreted text role "exc".

Exceptions defined in this module all inherit from :exc:`SubprocessError`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst`, line 710); *backlink*

Unknown interpreted text role "exc".

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

resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 712)

Unknown directive type "versionadded".

```
.. versionadded:: 3.3
   The :exc:`SubprocessError` base class was added.
```

Security Considerations

Unlike some other popen functions, this implementation will never implicitly call a system shell. This means that all characters, including shell metacharacters, can safely be passed to child processes. If the shell is invoked explicitly, via `shell=True`, it is the application's responsibility to ensure that all whitespace and metacharacters are quoted appropriately to avoid shell injection vulnerabilities. On :ref:`some platforms <shlex-quote-warning>`, it is possible to use :func:`shlex.quote` for this escaping.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 720); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 720); *backlink*

Unknown interpreted text role "func".

Popen Objects

Instances of the :class:`Popen` class have the following methods:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 734); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 737)

Unknown directive type "method".

```
.. method:: Popen.poll()

   Check if child process has terminated.  Set and return
   :attr:`~Popen.returncode` attribute. Otherwise, returns ``None``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 743)

Unknown directive type "method".

```
.. method:: Popen.wait(timeout=None)

   Wait for child process to terminate.  Set and return
   :attr:`~Popen.returncode` attribute.

   If the process does not terminate after *timeout* seconds, raise a
   :exc:`TimeoutExpired` exception.  It is safe to catch this exception and
   retry the wait.

   .. note::

      This will deadlock when using ``stdout=PIPE`` or ``stderr=PIPE``
      and the child process generates enough output to a pipe such that
      it blocks waiting for the OS pipe buffer to accept more data.
      Use :meth:`Popen.communicate` when using pipes to avoid that.

   .. note::

      The function is implemented using a busy loop (non-blocking call and
      short sleeps). Use the :mod:`asyncio` module for an asynchronous wait:
      see :class:`asyncio.create_subprocess_exec`.

   .. versionchanged:: 3.3
      *timeout* was added.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 768)

Unknown directive type "method".

```
.. method:: Popen.communicate(input=None, timeout=None)

   Interact with process: Send data to stdin.  Read data from stdout and stderr,
   until end-of-file is reached.  Wait for process to terminate and set the
```

```
:attr:`~Popen.returncode` attribute.  The optional *input* argument should be
data to be sent to the child process, or ``None``, if no data should be sent
to the child.  If streams were opened in text mode, *input* must be a string.
Otherwise, it must be bytes.

:meth:`communicate` returns a tuple ``(stdout_data, stderr_data)``.
The data will be strings if streams were opened in text mode; otherwise,
bytes.

Note that if you want to send data to the process's stdin, you need to create
the Popen object with ``stdin=PIPE``.  Similarly, to get anything other than
``None`` in the result tuple, you need to give ``stdout=PIPE`` and/or
``stderr=PIPE`` too.

If the process does not terminate after *timeout* seconds, a
:exc:`TimeoutExpired` exception will be raised.  Catching this exception and
retrying communication will not lose any output.

The child process is not killed if the timeout expires, so in order to
cleanup properly a well-behaved application should kill the child process and
finish communication::

   proc = subprocess.Popen(...)
   try:
       outs, errs = proc.communicate(timeout=15)
   except TimeoutExpired:
       proc.kill()
       outs, errs = proc.communicate()

.. note::

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

.. versionchanged:: 3.3
   *timeout* was added.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 810)

Unknown directive type "method".

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

   Sends the signal *signal* to the child.

   Do nothing if the process completed.

   .. note::

      On Windows, 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]subprocess.rst, line 823)

Unknown directive type "method".

```
.. method:: Popen.terminate()

   Stop the child. On POSIX OSs the method sends SIGTERM to the
   child. On Windows the Win32 API function :c:func:`TerminateProcess` is called
   to stop the child.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 830)

Unknown directive type "method".

```
.. method:: Popen.kill()

   Kills the child. On POSIX OSs the function sends SIGKILL to the child.
   On Windows :meth:`kill` is an alias for :meth:`terminate`.
```

The following attributes are also available:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 838)

Unknown directive type "attribute".

```
.. attribute:: Popen.args

   The *args* argument as it was passed to :class:`Popen` -- a
```

```
   sequence of program arguments or else a single string.

   .. versionadded:: 3.3
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 845)

Unknown directive type "attribute".

```
.. attribute:: Popen.stdin

   If the *stdin* argument was :data:`PIPE`, this attribute is a writeable
   stream object as returned by :func:`open`. If the *encoding* or *errors*
   arguments were specified or the *universal_newlines* argument was ``True``,
   the stream is a text stream, otherwise it is a byte stream. If the *stdin*
   argument was not :data:`PIPE`, this attribute is ``None``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 854)

Unknown directive type "attribute".

```
.. attribute:: Popen.stdout

   If the *stdout* argument was :data:`PIPE`, this attribute is a readable
   stream object as returned by :func:`open`. Reading from the stream provides
   output from the child process. If the *encoding* or *errors* arguments were
   specified or the *universal_newlines* argument was ``True``, the stream is a
   text stream, otherwise it is a byte stream. If the *stdout* argument was not
   :data:`PIPE`, this attribute is ``None``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 864)

Unknown directive type "attribute".

```
.. attribute:: Popen.stderr

   If the *stderr* argument was :data:`PIPE`, this attribute is a readable
   stream object as returned by :func:`open`. Reading from the stream provides
   error output from the child process. If the *encoding* or *errors* arguments
   were specified or the *universal_newlines* argument was ``True``, the stream
   is a text stream, otherwise it is a byte stream. If the *stderr* argument was
   not :data:`PIPE`, this attribute is ``None``.
```

Warning

Use :meth:`~Popen.communicate` rather than :attr:`.stdin.write <Popen.stdin>`, :attr:`.stdout.read <Popen.stdout>` or :attr:`.stderr.read <Popen.stderr>` to avoid deadlocks due to any of the other OS pipe buffers filling up and blocking the child process.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 875); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 875); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 875); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 875); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 881)

Unknown directive type "attribute".

```
.. attribute:: Popen.pid

   The process ID of the child process.

   Note that if you set the *shell* argument to ``True``, this is the process ID
   of the spawned shell.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 889)

Unknown directive type "attribute".

```
.. attribute:: Popen.returncode

   The child return code, set by :meth:`poll` and :meth:`wait` (and indirectly
   by :meth:`communicate`).  A ``None`` value indicates that the process
   hasn't terminated yet.

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

Windows Popen Helpers

The :class:`STARTUPINFO` class and following constants are only available on Windows.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 902); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 905)

Invalid class attribute value for "class" directive: "STARTUPINFO(*, dwFlags=0, hStdInput=None, hStdOutput=None, \ hStdError=None, wShowWindow=0, lpAttributeList=None)".

```
.. class:: STARTUPINFO(*, dwFlags=0, hStdInput=None, hStdOutput=None, \
                       hStdError=None, wShowWindow=0, lpAttributeList=None)

   Partial support of the Windows
   `STARTUPINFO <https://msdn.microsoft.com/en-us/library/ms686331(v=vs.85).aspx>`__
   structure is used for :class:`Popen` creation.  The following attributes can
   be set by passing them as keyword-only arguments.

   .. versionchanged:: 3.7
      Keyword-only argument support was added.

   .. attribute:: dwFlags

      A bit field that determines whether certain :class:`STARTUPINFO`
      attributes are used when the process creates a window. ::

         si = subprocess.STARTUPINFO()
         si.dwFlags = subprocess.STARTF_USESTDHANDLES | subprocess.STARTF_USESHOWWINDOW

   .. attribute:: hStdInput

      If :attr:`dwFlags` specifies :data:`STARTF_USESTDHANDLES`, this attribute
      is the standard input handle for the process. If
      :data:`STARTF_USESTDHANDLES` is not specified, the default for standard
      input is the keyboard buffer.

   .. attribute:: hStdOutput

      If :attr:`dwFlags` specifies :data:`STARTF_USESTDHANDLES`, this attribute
      is the standard output handle for the process. Otherwise, this attribute
      is ignored and the default for standard output is the console window's
      buffer.

   .. attribute:: hStdError

      If :attr:`dwFlags` specifies :data:`STARTF_USESTDHANDLES`, this attribute
      is the standard error handle for the process. Otherwise, this attribute is
      ignored and the default for standard error is the console window's buffer.

   .. attribute:: wShowWindow

      If :attr:`dwFlags` specifies :data:`STARTF_USESHOWWINDOW`, this attribute
      can be any of the values that can be specified in the ``nCmdShow``
      parameter for the
      `ShowWindow <https://msdn.microsoft.com/en-us/library/ms633548(v=vs.85).aspx>`__
      function, except for ``SW_SHOWDEFAULT``. Otherwise, this attribute is
      ignored.

      :data:`SW_HIDE` is provided for this attribute. It is used when
      :class:`Popen` is called with ``shell=True``.
```

```
   .. attribute:: lpAttributeList

      A dictionary of additional attributes for process creation as given in
      ``STARTUPINFOEX``, see
      `UpdateProcThreadAttribute <https://msdn.microsoft.com/en-us/library/windows/desktop/ms686880(v=vs.85)

      Supported attributes:

      **handle_list**
         Sequence of handles that will be inherited. *close_fds* must be true if
         non-empty.

         The handles must be temporarily made inheritable by
         :func:`os.set_handle_inheritable` when passed to the :class:`Popen`
         constructor, else :class:`OSError` will be raised with Windows error
         ``ERROR_INVALID_PARAMETER`` (87).

         .. warning::

            In a multithreaded process, use caution to avoid leaking handles
            that are marked inheritable when combining this feature with
            concurrent calls to other process creation functions that inherit
            all handles such as :func:`os.system`.  This also applies to
            standard handle redirection, which temporarily creates inheritable
            handles.

      .. versionadded:: 3.7
```

Windows Constants

The :mod:`subprocess` module exposes the following constants.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 987); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 989)

Unknown directive type "data".

```
.. data:: STD_INPUT_HANDLE

   The standard input device. Initially, this is the console input buffer,
   ``CONIN$``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 994)

Unknown directive type "data".

```
.. data:: STD_OUTPUT_HANDLE

   The standard output device. Initially, this is the active console screen
   buffer, ``CONOUT$``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 999)

Unknown directive type "data".

```
.. data:: STD_ERROR_HANDLE

   The standard error device. Initially, this is the active console screen
   buffer, ``CONOUT$``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1004)

Unknown directive type "data".

```
.. data:: SW_HIDE

   Hides the window. Another window will be activated.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1008)

Unknown directive type "data".

```
.. data:: STARTF_USESTDHANDLES

   Specifies that the :attr:`STARTUPINFO.hStdInput`,
```

```
   :attr:`STARTUPINFO.hStdOutput`, and :attr:`STARTUPINFO.hStdError` attributes
   contain additional information.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1014)

Unknown directive type "data".

```
.. data:: STARTF_USESHOWWINDOW

   Specifies that the :attr:`STARTUPINFO.wShowWindow` attribute contains
   additional information.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1019)

Unknown directive type "data".

```
.. data:: CREATE_NEW_CONSOLE

   The new process has a new console, instead of inheriting its parent's
   console (the default).
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1024)

Unknown directive type "data".

```
.. data:: CREATE_NEW_PROCESS_GROUP

   A :class:`Popen` ``creationflags`` parameter to specify that a new process
   group will be created. This flag is necessary for using :func:`os.kill`
   on the subprocess.

   This flag is ignored if :data:`CREATE_NEW_CONSOLE` is specified.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1032)

Unknown directive type "data".

```
.. data:: ABOVE_NORMAL_PRIORITY_CLASS

   A :class:`Popen` ``creationflags`` parameter to specify that a new process
   will have an above average priority.

   .. versionadded:: 3.7
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1039)

Unknown directive type "data".

```
.. data:: BELOW_NORMAL_PRIORITY_CLASS

   A :class:`Popen` ``creationflags`` parameter to specify that a new process
   will have a below average priority.

   .. versionadded:: 3.7
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1046)

Unknown directive type "data".

```
.. data:: HIGH_PRIORITY_CLASS

   A :class:`Popen` ``creationflags`` parameter to specify that a new process
   will have a high priority.

   .. versionadded:: 3.7
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1053)

Unknown directive type "data".

```
.. data:: IDLE_PRIORITY_CLASS

   A :class:`Popen` ``creationflags`` parameter to specify that a new process
   will have an idle (lowest) priority.

   .. versionadded:: 3.7
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1060)

Unknown directive type "data".

```
.. data:: NORMAL_PRIORITY_CLASS

   A :class:`Popen` ``creationflags`` parameter to specify that a new process
   will have an normal priority. (default)

   .. versionadded:: 3.7
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1067)

Unknown directive type "data".

```
.. data:: REALTIME_PRIORITY_CLASS

   A :class:`Popen` ``creationflags`` parameter to specify that a new process
   will have realtime priority.
   You should almost never use REALTIME_PRIORITY_CLASS, because this interrupts
   system threads that manage mouse input, keyboard input, and background disk
   flushing. This class can be appropriate for applications that "talk" directly
   to hardware or that perform brief tasks that should have limited interruptions.

   .. versionadded:: 3.7
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1078)

Unknown directive type "data".

```
.. data:: CREATE_NO_WINDOW

   A :class:`Popen` ``creationflags`` parameter to specify that a new process
   will not create a window.

   .. versionadded:: 3.7
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1085)

Unknown directive type "data".

```
.. data:: DETACHED_PROCESS

   A :class:`Popen` ``creationflags`` parameter to specify that a new process
   will not inherit its parent's console.
   This value cannot be used with CREATE_NEW_CONSOLE.

   .. versionadded:: 3.7
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1093)

Unknown directive type "data".

```
.. data:: CREATE_DEFAULT_ERROR_MODE

   A :class:`Popen` ``creationflags`` parameter to specify that a new process
   does not inherit the error mode of the calling process. Instead, the new
   process gets the default error mode.
   This feature is particularly useful for multithreaded shell applications
   that run with hard errors disabled.

   .. versionadded:: 3.7
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1103)

Unknown directive type "data".

```
.. data:: CREATE_BREAKAWAY_FROM_JOB

   A :class:`Popen` ``creationflags`` parameter to specify that a new process
   is not associated with the job.

   .. versionadded:: 3.7
```

Older high-level API

Prior to Python 3.5, these three functions comprised the high level API to subprocess. You can now use :func:`run` in many cases, but lots of existing code calls these functions.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1115); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1119)

Unknown directive type "function".

```
.. function:: call(args, *, stdin=None, stdout=None, stderr=None, \
                   shell=False, cwd=None, timeout=None, **other_popen_kwargs)

   Run the command described by *args*.  Wait for command to complete, then
   return the :attr:`~Popen.returncode` attribute.

   Code needing to capture stdout or stderr should use :func:`run` instead::

       run(...).returncode

   To suppress stdout or stderr, supply a value of :data:`DEVNULL`.

   The arguments shown above are merely some common ones.
   The full function signature is the
   same as that of the :class:`Popen` constructor - this function passes all
   supplied arguments other than *timeout* directly through to that interface.

   .. note::

      Do not use ``stdout=PIPE`` or ``stderr=PIPE`` with this
      function.  The child process will block if it generates enough
      output to a pipe to fill up the OS pipe buffer as the pipes are
      not being read from.

   .. versionchanged:: 3.3
      *timeout* was added.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1146)

Unknown directive type "function".

```
.. function:: check_call(args, *, stdin=None, stdout=None, stderr=None, \
                         shell=False, cwd=None, timeout=None, \
                         **other_popen_kwargs)

   Run command with arguments.  Wait for command to complete. If the return
   code was zero then return, otherwise raise :exc:`CalledProcessError`. The
   :exc:`CalledProcessError` object will have the return code in the
   :attr:`~CalledProcessError.returncode` attribute.
   If :func:`check_call` was unable to start the process it will propagate the exception
   that was raised.

   Code needing to capture stdout or stderr should use :func:`run` instead::

       run(..., check=True)

   To suppress stdout or stderr, supply a value of :data:`DEVNULL`.

   The arguments shown above are merely some common ones.
   The full function signature is the
   same as that of the :class:`Popen` constructor - this function passes all
   supplied arguments other than *timeout* directly through to that interface.

   .. note::

      Do not use ``stdout=PIPE`` or ``stderr=PIPE`` with this
      function.  The child process will block if it generates enough
      output to a pipe to fill up the OS pipe buffer as the pipes are
      not being read from.

   .. versionchanged:: 3.3
      *timeout* was added.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1179)

Unknown directive type "function".

```
.. function:: check_output(args, *, stdin=None, stderr=None, shell=False, \
                           cwd=None, encoding=None, errors=None, \
                           universal_newlines=None, timeout=None, text=None, \
                           **other_popen_kwargs)

   Run command with arguments and return its output.
```

```
If the return code was non-zero it raises a :exc:`CalledProcessError`. The
:exc:`CalledProcessError` object will have the return code in the
:attr:`~CalledProcessError.returncode` attribute and any output in the
:attr:`~CalledProcessError.output` attribute.

This is equivalent to::

    run(..., check=True, stdout=PIPE).stdout

The arguments shown above are merely some common ones.
The full function signature is largely the same as that of :func:`run` -
most arguments are passed directly through to that interface.
One API deviation from :func:`run` behavior exists: passing ``input=None``
will behave the same as ``input=b''`` (or ``input=''``, depending on other
arguments) rather than using the parent's standard input file handle.

By default, this function will return the data as encoded bytes. The actual
encoding of the output data may depend on the command being invoked, so the
decoding to text will often need to be handled at the application level.

This behaviour may be overridden by setting *text*, *encoding*, *errors*,
or *universal_newlines* to ``True`` as described in
:ref:`frequently-used-arguments` and :func:`run`.

To also capture standard error in the result, use
``stderr=subprocess.STDOUT``::

   >>> subprocess.check_output(
   ...     "ls non_existent_file; exit 0",
   ...     stderr=subprocess.STDOUT,
   ...     shell=True)
   'ls: non_existent_file: No such file or directory\n'

.. versionadded:: 3.1

.. versionchanged:: 3.3
   *timeout* was added.

.. versionchanged:: 3.4
   Support for the *input* keyword argument was added.

.. versionchanged:: 3.6
   *encoding* and *errors* were added.  See :func:`run` for details.

.. versionadded:: 3.7
   *text* was added as a more readable alias for *universal_newlines*.
```

Replacing Older Functions with the :mod:`subprocess` Module

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1236); *backlink*

Unknown interpreted text role "mod".

In this section, "a becomes b" means that b can be used as a replacement for a.

Note

All "a" functions in this section fail (more or less) silently if the executed program cannot be found; the "b" replacements raise :exc:`OSError` instead.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1243); *backlink*

Unknown interpreted text role "exc".

In addition, the replacements using :func:`check_output` will fail with a :exc:`CalledProcessError` if the requested operation produces a non-zero return code. The output is still available as the :attr:`~CalledProcessError.output` attribute of the raised exception.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1247); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1247); *backlink*

Unknown interpreted text role "exc".

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]subprocess.rst, line 1247); *backlink*

Unknown interpreted text role "attr".

In the following examples, we assume that the relevant functions have already been imported from the :mod:`subprocess` module.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1252); *backlink*

Unknown interpreted text role "mod".

Replacing :program:`/bin/sh` shell command substitution

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1256); *backlink*

Unknown interpreted text role "program".

```
output=$(mycmd myarg)
```

becomes:

```
output = check_output(["mycmd", "myarg"])
```

Replacing shell pipeline

```
output=$(dmesg | grep hda)
```

becomes:

```
p1 = Popen(["dmesg"], stdout=PIPE)
p2 = Popen(["grep", "hda"], stdin=p1.stdout, stdout=PIPE)
p1.stdout.close()  # Allow p1 to receive a SIGPIPE if p2 exits.
output = p2.communicate()[0]
```

The `p1.stdout.close()` call after starting the p2 is important in order for p1 to receive a SIGPIPE if p2 exits before p1.

Alternatively, for trusted input, the shell's own pipeline support may still be used directly:

```
output=$(dmesg | grep hda)
```

becomes:

```
output = check_output("dmesg | grep hda", shell=True)
```

Replacing :func:`os.system`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1296); *backlink*

Unknown interpreted text role "func".

```
sts = os.system("mycmd" + " myarg")
# becomes
retcode = call("mycmd" + " myarg", shell=True)
```

Notes:

- Calling the program through the shell is usually not required.
- The :func:`call` return value is encoded differently to that of :func:`os.system`.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1308); *backlink*

  Unknown interpreted text role "func".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1308); *backlink*

  Unknown interpreted text role "func".

- The :func:`os.system` function ignores SIGINT and SIGQUIT signals while the command is running, but the caller must do this separately when using the :mod:`subprocess` module.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1311); *backlink*

  Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1311); *backlink*

Unknown interpreted text role "mod".

A more realistic example would look like this:

```
try:
    retcode = call("mycmd" + " myarg", shell=True)
    if retcode < 0:
        print("Child was terminated by signal", -retcode, file=sys.stderr)
    else:
        print("Child returned", retcode, file=sys.stderr)
except OSError as e:
    print("Execution failed:", e, file=sys.stderr)
```

Replacing the :func:`os.spawn <os.spawnl>` family

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1327); *backlink*

Unknown interpreted text role "func".

P_NOWAIT example:

```
pid = os.spawnlp(os.P_NOWAIT, "/bin/mycmd", "mycmd", "myarg")
==>
pid = Popen(["/bin/mycmd", "myarg"]).pid
```

P_WAIT example:

```
retcode = os.spawnlp(os.P_WAIT, "/bin/mycmd", "mycmd", "myarg")
==>
retcode = call(["/bin/mycmd", "myarg"])
```

Vector example:

```
os.spawnvp(os.P_NOWAIT, path, args)
==>
Popen([path] + args[1:])
```

Environment example:

```
os.spawnlpe(os.P_NOWAIT, "/bin/mycmd", "mycmd", "myarg", env)
==>
Popen(["/bin/mycmd", "myarg"], env={"PATH": "/usr/bin"})
```

Replacing :func:`os.popen`, :func:`os.popen2`, :func:`os.popen3`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1356); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1356); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1356); *backlink*

Unknown interpreted text role "func".

```
(child_stdin, child_stdout) = os.popen2(cmd, mode, bufsize)
==>
p = Popen(cmd, shell=True, bufsize=bufsize,
          stdin=PIPE, stdout=PIPE, close_fds=True)
(child_stdin, child_stdout) = (p.stdin, p.stdout)

(child_stdin,
 child_stdout,
 child_stderr) = os.popen3(cmd, mode, bufsize)
==>
p = Popen(cmd, shell=True, bufsize=bufsize,
          stdin=PIPE, stdout=PIPE, stderr=PIPE, close_fds=True)
(child_stdin,
 child_stdout,
 child_stderr) = (p.stdin, p.stdout, p.stderr)

(child_stdin, child_stdout_and_stderr) = os.popen4(cmd, mode, bufsize)
==>
p = Popen(cmd, shell=True, bufsize=bufsize,
          stdin=PIPE, stdout=PIPE, stderr=STDOUT, close_fds=True)
(child_stdin, child_stdout_and_stderr) = (p.stdin, p.stdout)
```

Return code handling translates as follows:

```
pipe = os.popen(cmd, 'w')
...
rc = pipe.close()
if rc is not None and rc >> 8:
    print("There were some errors")
==>
process = Popen(cmd, stdin=PIPE)
...
process.stdin.close()
if process.wait() != 0:
    print("There were some errors")
```

Replacing functions from the :mod:`popen2` module

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1402); *backlink*

Unknown interpreted text role "mod".

Note

If the cmd argument to popen2 functions is a string, the command is executed through /bin/sh. If it is a list, the command is directly executed.

```
(child_stdout, child_stdin) = popen2.popen2("somestring", bufsize, mode)
==>
p = Popen("somestring", shell=True, bufsize=bufsize,
          stdin=PIPE, stdout=PIPE, close_fds=True)
(child_stdout, child_stdin) = (p.stdout, p.stdin)

(child_stdout, child_stdin) = popen2.popen2(["mycmd", "myarg"], bufsize, mode)
==>
p = Popen(["mycmd", "myarg"], bufsize=bufsize,
          stdin=PIPE, stdout=PIPE, close_fds=True)
(child_stdout, child_stdin) = (p.stdout, p.stdin)
```

:class:`popen2.Popen3` and :class:`popen2.Popen4` basically work as :class:`subprocess.Popen`, except that:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1426); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1426); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1426); *backlink*

Unknown interpreted text role "class".

- :class:`Popen` raises an exception if the execution fails.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1429); *backlink*

  Unknown interpreted text role "class".

- The *capturestderr* argument is replaced with the *stderr* argument.
- `stdin=PIPE` and `stdout=PIPE` must be specified.
- popen2 closes all file descriptors by default, but you have to specify `close_fds=True` with :class:`Popen` to guarantee this behavior on all platforms or past Python versions.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1435); *backlink*

  Unknown interpreted text role "class".

Legacy Shell Invocation Functions

This module also provides the following legacy functions from the 2.x `commands` module. These operations implicitly invoke the system shell and none of the guarantees described above regarding security and exception handling consistency are valid for these functions.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1448)

Unknown directive type "function".

```
.. function:: getstatusoutput(cmd)

   Return ``(exitcode, output)`` of executing *cmd* in a shell.

   Execute the string *cmd* in a shell with :meth:`Popen.check_output` and
   return a 2-tuple ``(exitcode, output)``. The locale encoding is used;
   see the notes on :ref:`frequently-used-arguments` for more details.

   A trailing newline is stripped from the output.
   The exit code for the command can be interpreted as the return code
   of subprocess.  Example::

      >>> subprocess.getstatusoutput('ls /bin/ls')
      (0, '/bin/ls')
      >>> subprocess.getstatusoutput('cat /bin/junk')
      (1, 'cat: /bin/junk: No such file or directory')
      >>> subprocess.getstatusoutput('/bin/junk')
      (127, 'sh: /bin/junk: not found')
      >>> subprocess.getstatusoutput('/bin/kill $$')
      (-15, '')

   .. availability:: POSIX & Windows.

   .. versionchanged:: 3.3.4
      Windows support was added.

      The function now returns (exitcode, output) instead of (status, output)
      as it did in Python 3.3.3 and earlier.  exitcode has the same value as
      :attr:`~Popen.returncode`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1479)

Unknown directive type "function".

```
.. function:: getoutput(cmd)

   Return output (stdout and stderr) of executing *cmd* in a shell.

   Like :func:`getstatusoutput`, except the exit code is ignored and the return
   value is a string containing the command's output.  Example::

      >>> subprocess.getoutput('ls /bin/ls')
      '/bin/ls'

   .. availability:: POSIX & Windows.

   .. versionchanged:: 3.3.4
      Windows support added
```

Notes

Converting an argument sequence to a string on Windows

On Windows, an *args* sequence is converted to a string that can be parsed using the following rules (which correspond to the rules used by the MS C runtime):

1. Arguments are delimited by white space, which is either a space or a tab.
2. A string surrounded by double quotation marks is interpreted as a single argument, regardless of white space contained within. A quoted string can be embedded in an argument.
3. A double quotation mark preceded by a backslash is interpreted as a literal double quotation mark.
4. Backslashes are interpreted literally, unless they immediately precede a double quotation mark.
5. If backslashes immediately precede a double quotation mark, every pair of backslashes is interpreted as a literal backslash. If the number of backslashes is odd, the last backslash escapes the next double quotation mark as described in rule 3.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]subprocess.rst, line 1528)

Unknown directive type "seealso".

```
.. seealso::

   :mod:`shlex`
      Module which provides function to parse and escape command lines.
```