# Модуль subprocess

Модуль subprocess отвечает за выполнение следующих действий: порождение новых процессов, соединение c потоками стандартного ввода, стандартного вывода, стандартного вывода сообщений об ошибках и получение кодов возврата от этих процессов.

Рекомендуемым подходом к работе с подпроцессами является использование следующих вспомогательных функций для всех случаев, где они могут справиться. Для более сложных случаев может быть использован непосредственно интерфейс Popen.

**subprocess.call**(*args, \*, stdin=None, stdout=None, stderr=None, shell=False, timeout=None*) - выполняет команду, описанную args. Ожидает завершения команды, а затем возвращает код возврата.

Аргументы, приведенные выше, являются лишь наиболее распространенными из них. Полная сигнатура функция в значительной степени такая же, как конструктор Popen. Аргумент timeout передается Popen.wait(). Если тайм-аут истекает, дочерний процесс будет убит, а затем будет поднято исключение TimeoutExpired.

>>>

**>>>** subprocess.call(["ls", "-l"])

0

**>>>** subprocess.call("exit 1", shell=True)

1

**subprocess.check\_call**(*args, \*, stdin=None, stdout=None, stderr=None, shell=False, timeout=None*) - выполняет команду, описанную args. Ожидает завершения команды, а затем завершается, если код возврата 0, или поднимает исключение CalledProcessError, объект которого возвращает код завершения атрибутом returncode.

>>>

**>>>** subprocess.check\_call(["ls", "-l"])

0

**>>>** subprocess.check\_call("exit 1", shell=True)

Traceback (most recent call last):

*...*

subprocess.CalledProcessError: Command 'exit 1' returned non-zero exit status 1

**subprocess.check\_output**(*args, \*, input=None, stdin=None, stderr=None, shell=False, universal\_newlines=False, timeout=None*) - выполняет команду и возвращает её вывод. Поднимает исключение CalledProcessError, если код возврата ненулевой.

>>>

**>>>** subprocess.check\_output(["echo", "Hello World!"])

b'Hello World!\n'

**>>>** subprocess.check\_output(["echo", "Hello World!"], universal\_newlines=True)

'Hello World!\n'

**>>>** subprocess.check\_output(["sed", "-e", "s/foo/bar/"],

**...**  input=b"when in the course of fooman events**\n**")

b'when in the course of barman events\n'

**>>>** subprocess.check\_output("exit 1", shell=True)

Traceback (most recent call last):

*...*

subprocess.CalledProcessError: Command 'exit 1' returned non-zero exit status 1

>>>

**>>>** subprocess.check\_output(

**...**  "ls non\_existent\_file; exit 0",

**...**  stderr=subprocess.STDOUT,

**...**  shell=True)

**...**

'ls: non\_existent\_file: No such file or directory\n'

Создание новых процессов и управление ими в данном модуле обрабатывается классом Popen. Он предлагает большую гибкость, так что разработчики могут справиться с менее распространенными случаями, не охваченными удобными функциями.

**subprocess.DEVNULL** - значение, которое может использоваться в качестве аргумента stdin, stdout или stderr. Означает, что будет использован специальный файл devnull.

**subprocess.PIPE** - значение, которое может использоваться в качестве аргумента stdin, stdout или stderr. Означает, что для дочернего процесса будет создан пайп.

**subprocess.STDOUT** - значение, которое может использоваться в качестве аргумента stderr. Означает, что поток ошибок будет перенаправлен в поток вывода.

Класс **subprocess.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=False, startupinfo=None, creationflags=0, restore\_signals=True, start\_new\_session=False, pass\_fds=()*) - Выполняет программу в новом процессе. args – строка или последовательность аргументов программы. Обычно первым указывают исполняемую программу, а затем аргументы, но также ее можно указать в параметре executable.

Также Popen поддерживает [менеджеры контекста](https://pythonworld.ru/osnovy/with-as-menedzhery-konteksta.html):

**with** Popen(["ifconfig"], stdout=PIPE) **as** proc:

log.write(proc.stdout.read())

Методы класса Popen:

**Popen.poll**() - если процесс завершил работу - вернёт код возврата, в ином случае None.

**Popen.wait**(*timeout=None*) - ожидает завершения работы процесса и возвращает код возврата. Если в течение timeout процесс не завершился, поднимется исключение TimeoutExpired (которое можно перехватить, после чего сделать ещё раз wait).

Этот метод может вызвать блокировку (зависание), если установлено stdout=PIPE или stderr=PIPE, и дочерний процесс генерирует большое количество данных в stdout и stderr. Использование communicate() позволит избежать этого.

**Popen.communicate**(*input=None, timeout=None*) - взаимодействовует с процессом: посылает данные, содержащиеся в input в stdin процесса, ожидает завершения работы процесса, возвращает кортеж данных потока вывода и ошибок. При этом в Popen необходимо задать значение PIPE для stdin (если вы хотите посылать в stdin), stdout, stderr (если вы хотите прочитать вывод дочернего процесса).

Если в течение timeout процесс не завершился, поднимется исключение TimeoutExpired (которое можно перехватить, после чего сделать ещё раз communicate, либо убить дочерний процесс).

Прочитанные данные буферизируются в память, поэтому не стоит применять этот метод в случае огромных выходных данных.

**Popen.send\_signal**(*signal*) - посылает сигнал *signal*.

**Popen.terminate**() - останавливает дочерний процесс.

**Popen.kill**() - убивает дочерний процесс.