在Python3中， Subprocess用于代替os.system以及os.spawn\*等模块。在大多数情况下，我们可以使用subprocess预先创建的方法和系统交互。在复杂的任务下，我们可以使用subprocess.Popen加入不同的参数，来自定义与系统的交互。

注意：

If encoding or errors are specified, or universal\_newlines is true, file objects for stdin, stdout and stderr are opened in text mode using the specified encoding and errors or the [io.TextIOWrapper](https://docs.python.org/release/3.6.6/library/io.html" \l "io.TextIOWrapper" \o "io.TextIOWrapper) default. Otherwise, file objects are opened in binary mode.

在我们使用subprocess模块的时候，除非指定了universal\_newlines为true或者在subprocess方法中指定了encoding参数，否则返回的是bytes类型。Python3中bytes类型无法于str直接进行字符链接操作。

***Subprocess.run(…)***

**[可返回结果]**

在Python3.5才加入。执行命令，并且堵塞进行，命令完成后返回CompletedProcess对象实例，该实例用于获取当初执行命令的参数，返回的结果等等。

可选参数shell=True时， 如果shell没有指定为True, 则传入需要执行的命令时候，要以数组的形式传入，如果指定shell=True, 则传入的命令以字符串传入。

可选参数check等于True时，当执行状态不是0时，会抛出CalledProcessError异常提示

Subprocess.run获取返回结果时候，需要加入可选参数stdout=subprocess.PIPE. 然后从CompletedProcess对象的.stdout获取。

***subprocess.getstatusoutput(cmd)***

**[可返回结果，以默认编码decode]**

执行命令，返回一个tuple, 第一个元素为命令执行状态码(0为正常，1为非正常)，第二个元素为命令返回的结果.

没有shell参数，命令行以字符串的形式传入。

返回的结果自动以当前运行环境来进行decode，返回为str对象。

***subprocess.call(cmd)***

执行命令，只返回状态码，0为顺利执行。并在终端打印运行结果，无法将运行结果存在变量中。

***subprocess.check\_call()以及subprocess.call()***

check\_call基本和call功能一样，只是增加了返回状态码校验，如果执行状态码是0，则返回0，否则抛出异常

shell=True 如果shell没有指定为True, 则传入需要执行的命令时候，要以数组的形式传入，如果指定shell=True, 则传入的命令以字符串传入。

### *subprocess.check\_output()*

**[可返回结果，结果默认为bytes, 可用unversal\_newlines,或者encoding自动变为str对象]**

执行命令，如果状态码是0，则返回执行结果，否则抛出异常

shell=True 如果shell没有指定为True, 则传入需要执行的命令时候，要以数组的形式传入，如果指定shell=True, 则传入的命令以字符串传入。

### *subprocess.Popen()*

参数说明：

args： 要执行的shell命令，可以是字符串，也可以是命令各个参数组成的序列。当该参数的值是一个字符串时，该命令的解释过程是与平台相关的，因此通常建议将args参数作为一个序列传递。

bufsize： 指定缓存策略，0表示不缓冲，1表示行缓冲，其他大于1的数字表示缓冲区大小，负数 表示使用系统默认缓冲策略。

stdin, stdout, stderr： 分别表示程序标准输入、输出、错误句柄。

preexec\_fn： 用于指定一个将在子进程运行之前被调用的可执行对象，只在Unix平台下有效。

close\_fds： 如果该参数的值为True，则除了0,1和2之外的所有文件描述符都将会在子进程执行之前被关闭。

shell： 该参数用于标识是否使用shell作为要执行的程序，如果shell值为True，则建议将args参数作为一个字符串传递而不要作为一个序列传递。

cwd： 如果该参数值不是None，则该函数将会在执行这个子进程之前改变当前工作目录。

env： 用于指定子进程的环境变量，如果env=None，那么子进程的环境变量将从父进程中继承。如果env!=None，它的值必须是一个映射对象。

universal\_newlines： 如果该参数值为True，则该文件对象的stdin，stdout和stderr将会作为文本流被打开，否则他们将会被作为二进制流被打开。

startupinfo和creationflags： 这两个参数只在Windows下有效，它们将被传递给底层的CreateProcess()函数，用于设置子进程的一些属性，如主窗口的外观，进程优先级等。

### 注意： subprocess.Popen执行后指令便执行，不需要进行communicate. communicate只是和进程交流。例如mkdir在popen中，执行后不需要communicate便会新建文件夹。

### *subprocess.Popen类的实例可调用的方法*

|  |  |
| --- | --- |
| Popen.poll() | 用于检查子进程（命令）是否已经执行结束，没结束返回None，结束后返回状态码。 |
| Popen.wait(timeout=None) | 等待子进程结束，并返回状态码；如果在timeout指定的秒数之后进程还没有结束，将会抛出一个TimeoutExpired异常。 |
| Popen.communicate(input=None, timeout=None) | 该方法可用来与进程进行交互，比如发送数据到stdin，从stdout和stderr读取数据，直到到达文件末尾。 |
| Popen.send\_signal(signal) | 发送指定的信号给这个子进程。 |
| Popen.terminate() | 停止该子进程。 |
| Popen.kill() | 杀死该子进程。 |

关于communicate()方法的说明：

该方法中的可选参数 input 应该是将被发送给子进程的数据，或者如没有数据发送给子进程，该参数应该是None。input参数的数据类型必须是字节串，如果universal\_newlines参数值为True，则input参数的数据类型必须是字符串。

该方法返回一个元组(stdout\_data, stderr\_data)，这些数据将会是字节穿或字符串（如果universal\_newlines的值为True）。

如果在timeout指定的秒数后该进程还没有结束，将会抛出一个TimeoutExpired异常。捕获这个异常，然后重新尝试通信不会丢失任何输出的数据。但是超时之后子进程并没有被杀死，为了合理的清除相应的内容，一个好的应用应该手动杀死这个子进程来结束通信。