Python的subprocess模块是从Python2.4以后加入Python的，用于代替os.system, os.popen等方法。

首先subprocess提供了一些已经包装的的快捷的方法，在不需要复杂处理的时候，可以马上调用对外部执行命令。

1. subprocess.call(*args*, *\**, *stdin=None*, *stdout=None*, *stderr=None*, *shell=False*)

语句会堵塞执行，当完成后在执行下面Python语句。返回的结果为0,1代码执行的结果状态码。

1. subprocess.check\_call(args, \*, stdin=None, stdout=None, stderr=None, shell=False)

语句会堵塞执行，当完成后在执行下面Python语句。若顺利执行则返回结果状态码0,如果不为0则会报CalledProcessError错误。

1. subprocess.check\_output(args, \*, stdin=None, stderr=None, shell=False, universal\_newlines=False)

语句会堵塞执行，当完成后在执行下面Python语句。若顺利执行则返回结果，而非状态码,如果不为0则会报CalledProcessError错误。

Note: 以上的方法在不要指定stdout,stderr的值为subprocess.PIPE将会引起死锁，需要执行时使用subprocess.Popen方法。

1. subprocess.run(args, \*, stdin=None, input=None, stdout=None, stderr=None, shell=False, cwd=None, timeout=None, check=False, encoding=None, errors=None)

此方法是从Python 3.5引入的，语句会堵塞执行，当完成后在执行下面Python语句。并且返回一个CompletedProcess的实例，我们通过这个返回的实例，可以获得当初执行语句的参数，结果，状态码等。

1. subprocess.Popen(*args*, *bufsize=0*, *executable=None*, *stdin=None*, *stdout=None*, *stderr=None*, *preexec\_fn=None*, *close\_fds=False*, *shell=False*, *cwd=None*, *env=None*, *universal\_newlines=False*, *startupinfo=None*, *creationflags=0*)

Popen的用处在与，当无法使用快捷的包装方法时候，需要有灵活自定义设置时，可以使用该方法。

1. *Subprocess.Popen*默认不堵塞，除非使用*wait()*方法。

Example:

example = subprocess.Popen(“tail -50f test.txt”,shell=True)

print(“Could you see me ?”)

没有example.wait()方法将可以输出Could you see me, 如果有则不会。

1. Subprocess.Popen方法中的stdin,stdout,stderr如果不指定默认为父进程的对应流。父进程即是执行这个Python文件的进程。

subprocess.Popen(“ls –l”,shell=True)时，stdout会默认为父进程stdout直接打印在屏幕上

subprocess.Popen(“ls –l”,stdout=subprocess.PIPE, shell=True)时，输出会暂存在PIPE里面，从而可以从里面取出内容(Popen.stdin.read()，或者转移到其他进程作为输入。)

1. example1 = subprocess.Popen("ls -l", shell=True,stdout = subprocess.PIPE)

example2 = subprocess.Popen("grep Test",shell=True, stdin = example1.stdout, stdout= subprocess.PIPE)

print(example2.communicate())

这里我们让example1返回的结果放入PIPE中进行缓存，然后当example2从PIPE中取出作为输入，然后example2将输出结果放入PIPE中。多个进程时，使用p.communicate()，将阻塞父进程，然后返回(stdout\_\_data,stderr\_\_data)作为结果。