Python 中执行系统命令常见的几种方法

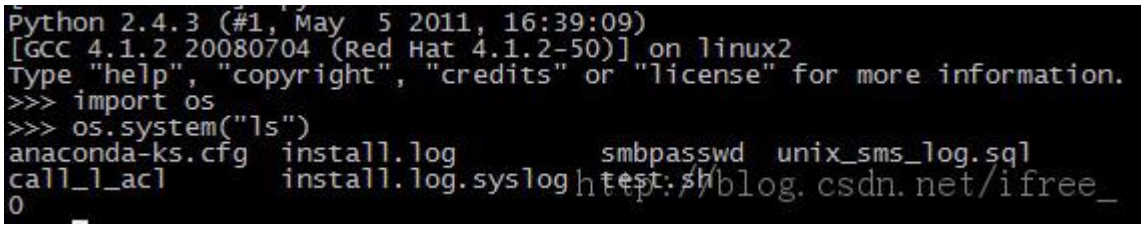
(1) os.system

这个方法是直接调用标准 C 的 system() 函数，仅仅在一个子终端运行系统命令，而不能获取命令执行后的返回信息。

os.system(command) -> exit_status

Execute the command (a string) in a subshell.

如果再命令行下执行，结果直接打印出来



(2) os.popen

该方法不但执行命令还返回执行后的信息对象，是通过一个管道文件将结果返回。

popen(command [, mode='r' [, bufsize]]) -> pipeOpen a pipe to/from a command returning a file object.

例如：

好处在于：将返回的结果赋于一变量，便于程序的处理。

(3) 使用模块 commands 模块

(status,result) = commands.getstatusoutput(cmd)

status 返回命令执行的返回值

result 返回命令执行结果

注意1：在类 unix 的系统下使用此方法返回的返回值（status）与脚本或命令执行之后的返回值不等，这是因为调用了 os.wait()的缘故，具体原因就得去了解下系统 wait()的实现了。需要正确的返回值（status），只需要对返回值进行右移8位操作就可以了。

注意2： 当执行命令的参数或者返回中包含了中文文字，那么建议使用 subprocess。

(4) 使用模块 subprocess

Subprocess 是一个功能强大的子进程管理模块，是替换 os.system ,os.spawn* 等方法的一个模块。

Class subprocess.Popen(args, bufsize=0, 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=())

有丰富的参数可以进行配置，可供我们自定义的选项多，灵活性高。之前我使用 os.system 的时候遇到文件描述符被子进程继承的问题，后来通过 close_fds = False 这个参数来解决的。