Oct. 11, 2013

 OS (/os/) subprocess (/subprocess/) System & OS (/systems-programming/)

Subprocess and Shell Commands in Python

Subprocess Overview

```
For a long time I have been using os.system() when dealing with system
administration tasks in Python.

The main reason for that, was that I thought that was the simplest way of
running Linux commands.
```

```
In the official python documentation we can read that subprocess should be used
for accessing system commands.

The subprocess module allows us to spawn processes, connect to their
input/output/error pipes, and obtain their return codes.

Subprocess intends to replace several other, older modules and functions,
like: os.system, os.spawn*, os.popen*, popen2.* commands. [Source (http://www.python.org/doc//current/library/subprocess.html)]
```

```
Let's start looking into the different functions of subprocess.
```

subprocess.call()

```
Run the command described by "args".

We can run the command line with the arguments passed as a list of strings
(example 1)  or by setting the shell argument to a True value (example 2)

Note, the default value of the shell argument is False.
```

```
Let's look at two examples where we show the summary of disk usage using
subprocess.call()
```

```
subprocess.call(['df', '-h'])
```

```
This time we set the shell argument to True
```

```
subprocess.call('du -hs $HOME', shell=True)
```

```
Note, the official Python documentation states a warning about using the
shell=True argument.

"Invoking the system shell with shell=True can be a security hazard if combined
with untrusted input" [source (http://docs.python.org/2/library/subprocess.html)]
```

```
Now, let's move on and look at the Input / Output.
```

Input and Output

```
With subprocess you can suppress the output, which is very handy when you
want to run a system call but are not interested about the standard output.

It also gives you a way to cleanly integrate shell commands into your scripts
while managing input/output in a standard way.
```

Return Codes

```
You can use subprocess.call return codes to determine the success of the command.

Every process will return an exit code and you can do something with your script
based on that code.

If the return code is anything else than zero, it means that an error occurred.

If you want to do system administration in Python, I recommend reading
Python for Unix and Linux System Administration (http://www.amazon.com/Python-Unix-Linux-System-Administration/dp/0596515820)
```

stdin, stdout and stderr

```
One of the trickiest part I had with subprocess was how to work with pipes
and to pipe commands together.

PIPE indicates that a new pipe to the child should be created.

The default setting is "None", which means that no redirection will occur.

The standard error (or stderr) can be STDOUT, which indicates that the stderr
data from the child process should be captured into the same file handle as
for stdout.
```

subprocess.Popen()

```
The underlying process creation and management in the subprocess module is
handled by the Popen class. subprocess.popen is replacing os.popen.
```

```
Let's get started with some real examples.

subprocess.Popen takes a list of arguments
```

```
import subprocess

p = subprocess.Popen(["echo", "hello world"], stdout=subprocess.PIPE)

print p.communicate()

>>>('hello world
', None)
```

```
Note, even though you could have used "shell=True", it is not the recommended
way of doing it.
```

```
If you know that you will only work with specific subprocess functions,
such as Popen and PIPE, then it is enough to only import those.
```

```
from subprocess import Popen, PIPE

p1 = Popen(["dmesg"], stdout=PIPE)

print p1.communicate()
```

Popen.communicate()

```
The communicate() method returns a tuple (stdoutdata, stderrdata).

Popen.communicate() interacts with process: Send data to stdin.

Read data from stdout and stderr, until end-of-file is reached.

Wait for process to terminate.

The optional input argument should be a string to be sent to the
child process, or None, if no data should be sent to the child.

Basically, when you use communicate() it means that you want to
execute the command
```

Ping program using subprocess

```
In the "More Reading" section below, you can find links to read more
about the subprocess module, but also examples.
```

```
Let's write our own ping program where we first ask the user for input,
and then perform the ping request to that host.
```

```
# Import the module
import subprocess

# Ask the user for input
host = raw_input("Enter a host to ping: ")

# Set up the echo command and direct the output to a pipe
p1 = subprocess.Popen(['ping', '-c 2', host], stdout=subprocess.PIPE)

# Run the command
output = p1.communicate()[0]

print output
```

```
Let's show one more example. This time we use the host command.
```

```
target = raw_input("Enter an IP or Host to ping:
")

host = subprocess.Popen(['host', target], stdout = subprocess.PIPE).communicate()[0]

print host
```

```
I recommend that you read the links below to gain more knowledge about the
subprocess module in Python.

If you have any questions or comments, please use the comment field below.
```

More Reading

```
http://docs.python.org/2/library/subprocess.html (http://docs.python.org/2/library/subprocess.html)
The ever useful and neat subprocess module (http://sharats.me/the-ever-useful-and-neat-subprocess-module.html)
http://pymotw.com/2/subprocess/ (http://pymotw.com/2/subprocess/)
http://www.bogotobogo.com/python/python_subprocess_module.php (http://www.bogotobogo.com/python/python_subprocess_module.php)
```

Recommended Python Training – DataCamp (https://www.datacamp.com?tap_a=5644-dce66f&tap_s=75426-9cf8ad)

For Python training (https://www.datacamp.com?tap_a=5644-dce66f&tap_s=75426-9cf8ad), our top recommendation is DataCamp.

Datacamp (https://www.datacamp.com?tap_a=5644-dce66f&tap_s=75426-9cf8ad) provides online interactive courses that combine interactive coding challenges with videos from top instructors in the field.

Datacamp has beginner to advanced Python training that programmers of all levels benefit from.

Tweet

Like Share 6

Read more about:

- OS (/os/)
- subprocess (/subprocess/)
- System & OS (/systems-programming/)

Did You Know?

Unleash the Techy in You

Loïc Le Meur

GETTING HOME SAFELY IS JUST A CLICK AWAY
NHTSA
Child Car Safety
SaferCar.gov/TheRightSeat

5 Comments Pythonforbeginners.com Aronno Jibon

Recommend 3 Share Sort by Best

Pythonforbeginners.com requires you to verify your email address before posting. Send verification email to tebunia@gmail.com

Join the discussion…

Kamil • 3 years ago

I think your explanation of
# Run the command
output = p1.communicate()[0]
is incorrect.

The command is actually run right after your Popen() call, only the output is redirected to a pipe because you specified it. If you set stdout=None, you will see your command runs regardless of communicate being called.

communicate() is meant as a means to send/receive input/output to/from a process, not start the process.

2 • Reply • Share ›

disqus_IgLvZXCjcf • 3 years ago

i cant find the command that gives you information of the ram !

• Reply • Share ›

Raju K • 3 years ago

how to do piping with subprocess?

• Reply • Share ›

Andrew Ippoliti → Raju K • 3 years ago

You specify that stdin is the stdout of another command. For example if you wanted to run `echo 'Hello World' | grep -i -o 'hello'`:

```
# print 'Hello World' to stdout
command1 = ['echo']
command1.append('Hello World')
process1 = subprocess.Popen(command1,stdout=subprocess.PIPE)

# Find 'hello' in the input and print that match to stdout
command2 = ['grep']
command2.append('-o')
command2.append('-i')
command2.append('hello')
process2 = subprocess.Popen(command2,stdin=process1.stdout,stdout=subprocess.PIPE)
```

5 • Reply • Share ›

E3D3 • 3 years ago

Thanks for writing this nice tutorial.

IMHO are the first 2 examples better with commands
where the return-code is relevant,
for example a move- or copy-operation
(although I'm system administration is not my job)

• Reply • Share ›

Subscribe Add Disqus to your site Add Disqus Add Privacy

Sponsored Links

Improve Your Phone's Camera Quality Easily Thanks to This Brilliant Device
HDZoom360

Now You Can Track Your Car Using Your Smartphone
Smart Device Trends

Get Cheaper Prices for Your Flights.
tripsinsider.com

End Your Nightly Snoring Nightmare With This Simple Solution
My Snoring Solution

Turn Your Smartphone into a Super Telephoto in 30 Seconds
HDZoom360

Every Smartphone Owner Needs This New Device
HDZoom360

Disclosure of Material Connection: Some of the links in the post above are "affiliate links." This means if you click on the link and purchase the item, I will receive an affiliate commission. Regardless, PythonForBeginners.com only recommend products or services that we try personally and believe will add value to our readers.

Search SEARCH

No Experience Jobs

Latest Technology News

Find Houses For Rent

Web Templates Free

Python For Beginners

Free Python Extensions

Python Programming Tutorials

Learn Python Programming

Graphic Logo Designs

Python Training Courses

Python Programming Help

Categories

Basics (/basics/)