Python: Capture standard output, standard error, and the exit code of a subprocess

subprocess (/search/subprocess) stderr (/search/stderr) stdout (/search/stdout)
exit (/search/exit) returncode (/search/returncode)
PYTHONUNBUFFERED (/search/PYTHONUNBUFFERED) tee (/search/tee)

❮ Prev (/python-seek) Next ❯ (/python-iterate-list-of-tuples)

I might be missing the obvious, but I don't think the `subprocess` module has a method that will capture the standard output, standard error, and the exit code of a subprocess in a single call. It is not complex to write one and can be useful.

The code that captures theresults

examples/python/capture.py

```
 1. import subprocess
 2. import sys
 3.
 4.
 5. def run(cmd):
 6.     proc = subprocess.Popen(cmd,
 7.         stdout = subprocess.PIPE,
 8.         stderr = subprocess.PIPE,
 9.     )
10.     stdout, stderr = proc.communicate()
11.
12.     return proc.returncode, stdout, stderr
13.
14. code, out, err = run([sys.executable, 'examples/python/run.py'])
15.
16. print("out: '{}'".format(out))
17. print("err: '{}'".format(err))
18. print("exit: {}".format(code))
19.
```

A sample external program

examples/python/run.py

```
1. import sys
2.
3. sys.stdout.write("STDOUT: Hello World!\n")
4. sys.stderr.write("STDERR: Welcome to the dark side!\n")
5. sys.stdout.write("STDOUT: Second line\n")
6. sys.stderr.write("STDERR: Warning\n")
7. exit(42)
8.
```

Running the external program directly

```
$ python examples/python/run.py

STDOUT: Hello World!
STDERR: Welcome to the dark side!
STDOUT: Second line
STDERR: Warning
```

```
echo $?
42
```

The results

```
out: 'STDOUT: Hello World!
STDOUT: Second line
'
err: 'STDERR: Welcome to the dark side!
STDERR: Warning
'
exit: 42
```

As you can see in this case the standard output and standard error are separated. You can't tell the exact order.

Capture STDOUT and STDERR together

Sometimes it is better to be able to capture the two together:

examples/python/capture_together.py

```
 1. import subprocess
 2. import sys
 3. import os
 4.
 5.
 6. def run(cmd):
 7.     os.environ['PYTHONUNBUFFERED'] = "1"
 8.     proc = subprocess.Popen(cmd,
 9.         stdout = subprocess.PIPE,
10.         stderr = subprocess.STDOUT,
11.     )
12.     stdout, stderr = proc.communicate()
13.
14.     return proc.returncode, stdout, stderr
15.
16. code, out, err = run([sys.executable, 'examples/python/run.py'])
17.
18. print("out: '{}'".format(out))
19. print("err: '{}'".format(err))
20. print("exit: {}".format(code))
21.
22.
```

Here we had stderr = subprocess.STDOUT instead of stderr = subprocess.PIPE .

```
out: 'STDOUT: Hello World!
STDERR: Welcome to the dark side!
STDOUT: Second line
STDERR: Warning
'
err: 'None'
exit: 42
```

In order for this to work properly on a Python script we'll need to turn off output buffering for the child process. This can be done by setting the PYTHONUNBUFFERED environment variable.

Tee: Capture and also print

Finally in this example we both collect the out and at the same time keep printing to the screen. Just to show off we captur STDOUT and STDERR both individually and mixed together.

You'll probably use some subset of these features.

examples/python/capture_tee.py

```
 1. from __future__ import print_function
 2. import os
 3. import subprocess
 4. import sys
 5.
 6. def run(cmd):
 7.     os.environ['PYTHONUNBUFFERED'] = "1"
 8.     proc = subprocess.Popen(cmd,
 9.                             stdout = subprocess.PIPE,
10.                             stderr = subprocess.PIPE,
11.                             universal_newlines = True,

12.                             )
13.     stdout = []
14.     stderr = []
15.     mix = []
16.     while proc.poll() is None:
17.         line = proc.stdout.readline()
18.         if line != "":
19.             stdout.append(line)
20.             mix.append(line)
21.             print(line, end='')
22.
23.         line = proc.stderr.readline()
24.         if line != "":
25.             stderr.append(line)
26.             mix.append(line)
27.             print(line, end='')
28.
29.     return proc.returncode, stdout, stderr, mix
30.
31. code, out, err, mix = run([sys.executable, 'examples/python/run.py'])
32.
33. print("out: '{}'".format(out))
34. print("err: '{}'".format(err))
35. print("err: '{}'".format(mix))
36. print("exit: {}".format(code))
37.
```

```
python examples/python/capture_tee.py
```

```
STDOUT: Hello World!
STDERR: Welcome to the dark side!
STDOUT: Second line
STDERR: Warning
out: '['STDOUT: Hello World!\n', 'STDOUT: Second line\n']'
err: '['STDERR: Welcome to the dark side!\n', 'STDERR: Warning\n']'
err: '['STDOUT: Hello World!\n', 'STDERR: Welcome to the dark side!\n', '
exit: 42
```

❮ Prev (/python-seek) Next ❯ (/python-iterate-list-of-tuples)

Gabor Szabo
(https://plus.google.com/102810219707784087582?rel=author)

Comments

In the comments, please wrap your code snippets within <pre> </pre> tags and use spaces for indentation.

Author: Gabor Szabo

Gabor who runs the Code Maven site helps companies set up **test automation**, CI/CD **Continuous Integration** and **Continuous Deployment** and other **DevOps** related systems. Gabor can help your team improve the development speed and reduce the risk of bugs.

He is also the author of a number of eBooks (https://leanpub.com/u/szabgab).

Contact Gabor (https://szabgab.com/contact.html) if you'd like to hire his services.

If you would like to support his freely available work, you can do it via Patreon (https://www.patreon.com/szabgab).