**Getting realtime output using Python Subprocess**

Posted by [Kannan Ponnusamy](https://plus.google.com/116771000414090157228) |January 28, 2015   
Tags: [bash](http://blog.endpoint.com/search/label/bash), [python](http://blog.endpoint.com/search/label/python)

**The Problem**

When I launch a long running unix process within a python script, it waits until the process is finished, and only then do I get the complete output of my program. This is annoying if I'm running a process that takes a while to finish. And I want to capture the output and display it in the nice manner with clear formatting.

**Using the subprocess and shlex library**

Python has a “batteries included” philosophy. I have used 2 standard libraries to solve this problem.

|  |  |
| --- | --- |
| 1  2 | import subprocess  import shlex |

* subprocess - Works with additional processes
* shlex - Lexical analysis of shell-style syntaxes

**subprocess.popen**

To run a process and read all of its output, set the stdout value to PIPE and call communicate().

|  |  |
| --- | --- |
| 1  2  3  4 | import subprocess  process = subprocess.Popen(['echo', '"Hello stdout"'], stdout=subprocess.PIPE)  stdout = process.communicate()[0]  print 'STDOUT:{}'.format(stdout) |

The above script will wait for the process to complete and then it will display the output. So now we are going to read the stdout line by line and display it in the console untill it completes the process.

|  |  |
| --- | --- |
| 1 | output = process.stdout.readline() |

This will read a line from the stdout.

|  |  |
| --- | --- |
| 1 | process.poll() |

The poll() method will return

* the exit code if the process is completed.
* None if the process is still running.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | while True:          output = process.stdout.readline()          if output == '' and process.poll() is not None:              break          if output:              print output.strip()      rc = process.poll() |

The above will loop and keep on reading the stdout and check for the return code and displays the output in real time.   
I had one more problem in parsing the shell commands to pass it to popen when I set the shell=False. Below is an example command:

|  |  |
| --- | --- |
| 1 | rsync -avzXH --delete --exclude=\*.swp --exclude=\*\*/drivers.ini /media/lgisos/lg.iso root@42-a:/isodevice |

To split the string using shell-like syntax I have used shlex library's split method.

**Here is the final code looks like**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | def run\_command(command):      process = subprocess.Popen(shlex.split(command), stdout=subprocess.PIPE)      while True:          output = process.stdout.readline()          if output == '' and process.poll() is not None:              break          if output:              print output.strip()      rc = process.poll()      return rc |