PipeSerial

Thomas Steen Rasmussen

Nov 01, 2020

CONTENTS:

CHAPTER

ONE

PIPESERIAL INTRODUCTION

PipeSerial is a Python utility to send a payload from stdin to a serial device and return the output on stdout. It is meant for sending quick commands to serial devices, for example for monitoring purposes.

Read on for usage instructions or go check out the examples!

CHAPTER

TWO

PIPESERIAL USAGE

Until I get something better written here are the argparse usage instructions:

```
usage: pipeserial [-h] [-c [EXPECTCOUNT]] [-d] [-e EXPECT]
                  [-l {DEBUG,INFO,WARNING,ERROR,CRITICAL}] [-p PAYLOAD]
                  [-q] [-s [SENDDELAY]] [-t [TIMEOUT]] [-v]
                  [--bytesize {5,6,7,8}] [--parity {N,E,O,S,M}]
                  [--stopbits {1,1.5,2}] [--rtscts] [--xonxoff] [--rts RTS]
                  [--dtr DTR]
                  serialport [baudrate]

PipeSerial version 0.4.0-dev. Sends input to a serial device, awaits (expects)
some text, and returns the output. See the manpage or ReadTheDocs for more
info.

positional arguments:
  serialport            Serial port device
  baudrate              Set baud rate, default: 115200

optional arguments:
  -h, --help            show this help message and exit
  -c [EXPECTCOUNT], --count [EXPECTCOUNT]
                        Collect output from the serial device until this many
                        regex matches, default: 1
  -d, --debug           Debug mode. Equal to setting --log-level=DEBUG.
  -e EXPECT, --expect EXPECT
                        Regular expressions to expect as end of the output.
                        Can be specified multiple times, default: [' OK ', '
                        ERROR ']
  -l {DEBUG,INFO,WARNING,ERROR,CRITICAL}, --log-level {DEBUG,INFO,WARNING,ERROR,
↪CRITICAL}
                        Logging level. One of DEBUG, INFO, WARNING, ERROR,
                        CRITICAL. Defaults to INFO.
  -p PAYLOAD, --payload PAYLOAD
                        The payload to send to the serial device, instead of
                        getting it from standard input. Default None.
  -q, --quiet           Quiet mode. No output at all if no errors are
                        encountered. Equal to setting --log-level=WARNING.
  -s [SENDDELAY], --send-delay [SENDDELAY]
                        Delay in seconds between sending each line of payload
                        to the serial device, default: 0.9
  -t [TIMEOUT], --timeout [TIMEOUT]
                        Timeout in seconds before giving up waiting for the
                        expected output, default: 30
  -v, --version         Show PipeSerial version and exit.
```

(continues on next page)

(continued from previous page)

```
Serial port options:
  --bytesize {5,6,7,8}  Set bytesize, one of {5 6 7 8}, default: 8
  --parity {N,E,O,S,M}  Set parity, one of {N E O S M}, default: N
  --stopbits {1,1.5,2}  Set stopbits, one of {1 1.5 2}, default: 1
  --rtscts              Enable RTS/CTS hardware flow control, default: off
  --xonxoff             Enable software flow control, default: off
  --rts RTS             Set initial RTS line state, one of {0, 1}, default:
                        none
  --dtr DTR             Set initial DTR line state, one of {0, 1}, default:
                        none
```

Read on for examples.

CHAPTER

THREE

PIPESERIAL EXAMPLES

3.1 Basic Use

Basic usage is just piping the input to pipeserial, telling it which output to expect, and the serial device. In these examples the serial devices are LTE modems responding to AT commands:

```
[tykling@container1 ~]$ echo "AT" | sudo pipeserial -e OK /dev/ttyU0.3
AT
OK
[tykling@container1 ~]$
```

Same thing with debug enabled:

```
[tykling@container1 ~]$ echo "AT" | sudo pipeserial -d -e OK /dev/ttyU0.3
2020-10-31 22:41:37 +0000 pipeserial DEBUG pipeserial.main():311:  Initialising the␣
↪PipeSerial class
2020-10-31 22:41:37 +0000 pipeserial DEBUG pipeserial.__init__():54:  Configuring␣
↪serial port {serialport} ...
2020-10-31 22:41:37 +0000 pipeserial DEBUG pipeserial.main():329:  Payload is 3 bytes
2020-10-31 22:41:37 +0000 pipeserial DEBUG pipeserial.main():335:  Output to expect: [
↪'OK']
2020-10-31 22:41:37 +0000 pipeserial DEBUG pipeserial.open():75:  Opening serial port.
↪..
2020-10-31 22:41:37 +0000 pipeserial DEBUG pipeserial.open():82:  Serial port opened␣
↪OK!
2020-10-31 22:41:37 +0000 pipeserial DEBUG pipeserial.run():107:  Sending payload␣
↪line: AT
2020-10-31 22:41:38 +0000 pipeserial DEBUG pipeserial.run():114:  Collecting output,␣
↪looking for one of these regular expressions: ['OK']
2020-10-31 22:41:38 +0000 pipeserial DEBUG pipeserial.run():116:  Will stop␣
↪collecting after 1 matches
2020-10-31 22:41:38 +0000 pipeserial DEBUG pipeserial.run():120:  Found match: 'OK'␣
↪(match number 1 of 1)
2020-10-31 22:41:38 +0000 pipeserial DEBUG pipeserial.run():127:  Done! Returning 9␣
↪bytes of output from serial device
2020-10-31 22:41:38 +0000 pipeserial DEBUG pipeserial.close():133:  Closing serial␣
↪port...
2020-10-31 22:41:38 +0000 pipeserial DEBUG pipeserial.close():135:  Serial port closed
2020-10-31 22:41:38 +0000 pipeserial DEBUG pipeserial.main():355:  Got 3 lines of␣
↪output from serial device /dev/ttyU0.3:
AT
OK
[tykling@container1 ~]$
```

3.2 Expecting Output

The regex matcher in pexpect is stream / single character based so ^ and $ will not work in the expect regexes. To match the start of a line add `\r\n` before the string, to match the end of a line match `\r\n` after the string.

The default is to match `\r\nOK\r\n` and `\r\nERROR\r\n` but this can be changed with `-e` / `--expect` as seen below:

```
[tykling@container1 ~]$ echo "AT" | sudo pipeserial -e OK -d /dev/ttyU0.3
2020-10-31 22:45:20 +0000 pipeserial DEBUG pipeserial.main():311:  Initialising the␣
↪PipeSerial class
2020-10-31 22:45:20 +0000 pipeserial DEBUG pipeserial.__init__():54:  Configuring␣
↪serial port {serialport} ...
2020-10-31 22:45:20 +0000 pipeserial DEBUG pipeserial.main():329:  Payload is 3 bytes
2020-10-31 22:45:20 +0000 pipeserial DEBUG pipeserial.main():335:  Output to expect: [
↪'OK']
2020-10-31 22:45:20 +0000 pipeserial DEBUG pipeserial.open():75:  Opening serial port.
↪..
2020-10-31 22:45:20 +0000 pipeserial DEBUG pipeserial.open():82:  Serial port opened␣
↪OK!
2020-10-31 22:45:20 +0000 pipeserial DEBUG pipeserial.run():107:  Sending payload␣
↪line: AT
2020-10-31 22:45:21 +0000 pipeserial DEBUG pipeserial.run():114:  Collecting output,␣
↪looking for one of these regular expressions: ['OK']
2020-10-31 22:45:21 +0000 pipeserial DEBUG pipeserial.run():116:  Will stop␣
↪collecting after 1 matches
2020-10-31 22:45:21 +0000 pipeserial DEBUG pipeserial.run():120:  Found match: 'OK'␣
↪(match number 1 of 1)
2020-10-31 22:45:21 +0000 pipeserial DEBUG pipeserial.run():127:  Done! Returning 7␣
↪bytes of output from serial device
2020-10-31 22:45:21 +0000 pipeserial DEBUG pipeserial.close():133:  Closing serial␣
↪port...
2020-10-31 22:45:21 +0000 pipeserial DEBUG pipeserial.close():135:  Serial port closed
2020-10-31 22:45:21 +0000 pipeserial DEBUG pipeserial.main():355:  Got 2 lines of␣
↪output from serial device /dev/ttyU0.3:
AT
OK
[tykling@container1 ~]$
```

Multiple lines of output from the serial device will be returned up until the expected output is encountered:

```
[tykling@container1 ~]$ echo "ATI" | sudo pipeserial /dev/ttyU0.3
ATI
Quectel
EC25
Revision: EC25EFAR06A06M4G

OK

[tykling@container1 ~]$
```

Same thing with a different LTE modem:

```
[tykling@container1 ~]$ echo "ATI" | sudo pipeserial /dev/ttyU1.2
ATI
Manufacturer: Huawei Technologies Co., Ltd.
Model: ME909s-120
```

(continues on next page)

(continued from previous page)

```
Revision: 11.617.15.00.00
IMEI: 123456789012345
+GCAP: +CGSM,+DS,+ES

OK

[tykling@container1 ~]$
```

3.3 Multiline Input

Multiple lines of payload can be sent to the serial device. Remember `-e` to make `echo` understand `\n`. PipeSerial is also told with `-c 2` to collect output until 2 expect matches has been seen:

```
[tykling@container1 ~]$ echo -ne "AT\nATI" | sudo venv/bin/python pipeserial.py -c 2 -
↪d /dev/ttyU0.3
2020-10-31 22:49:42 +0000 pipeserial DEBUG pipeserial.main():311:  Initialising the␣
↪PipeSerial class
2020-10-31 22:49:42 +0000 pipeserial DEBUG pipeserial.__init__():54:  Configuring␣
↪serial port {serialport} ...
2020-10-31 22:49:42 +0000 pipeserial DEBUG pipeserial.main():329:  Payload is 6 bytes
2020-10-31 22:49:42 +0000 pipeserial DEBUG pipeserial.main():335:  Output to expect: [
↪'\r\nOK\r\n', '\r\nERROR\r\n']
2020-10-31 22:49:42 +0000 pipeserial DEBUG pipeserial.open():75:  Opening serial port.
↪..
2020-10-31 22:49:42 +0000 pipeserial DEBUG pipeserial.open():82:  Serial port opened␣
↪OK!
2020-10-31 22:49:42 +0000 pipeserial DEBUG pipeserial.run():107:  Sending payload␣
↪line: AT
2020-10-31 22:49:43 +0000 pipeserial DEBUG pipeserial.run():107:  Sending payload␣
↪line: ATI
2020-10-31 22:49:44 +0000 pipeserial DEBUG pipeserial.run():114:  Collecting output,␣
↪looking for one of these regular expressions: ['\r\nOK\r\n', '\r\nERROR\r\n']
2020-10-31 22:49:44 +0000 pipeserial DEBUG pipeserial.run():116:  Will stop␣
↪collecting after 2 matches
2020-10-31 22:49:44 +0000 pipeserial DEBUG pipeserial.run():120:  Found match: 'OK'␣
↪(match number 1 of 2)
2020-10-31 22:49:44 +0000 pipeserial DEBUG pipeserial.run():120:  Found match: 'OK'␣
↪(match number 2 of 2)
2020-10-31 22:49:44 +0000 pipeserial DEBUG pipeserial.run():127:  Done! Returning 64␣
↪bytes of output from serial device
2020-10-31 22:49:44 +0000 pipeserial DEBUG pipeserial.close():133:  Closing serial␣
↪port...
2020-10-31 22:49:44 +0000 pipeserial DEBUG pipeserial.close():135:  Serial port closed
2020-10-31 22:49:44 +0000 pipeserial DEBUG pipeserial.main():355:  Got 9 lines of␣
↪output from serial device /dev/ttyU0.3:
AT
OK
ATI
Quectel
EC25
Revision: EC25EFAR06A06M4G

OK
```

(continues on next page)

(continued from previous page)

```
[tykling@container1 ~]$
```

The output from the serial device is sent to stdout and the logging is sent to stderr.

CHAPTER

FOUR

PIPESERIAL CHANGELOG

All notable changes to this project will be documented in this file.

This project adheres to Semantic Versioning from https://semver.org/spec/v2.0.0.html, and this changelogs format is based on Keep a Changelog from https://keepachangelog.com/en/1.0.0/.

CHAPTER

FIVE

[0.4.0] - UNRELEASED

- No changes

CHAPTER

SIX

[0.3.0] - 2020-11-01

6.1 Fixed

- Remove unused version section breaking everything

CHAPTER

SEVEN

[0.2.0] - 2020-11-01

7.1 Fixed

- Typo in debug console output prefix, was `pipeperial` instead of `pipeserial`.
- Add missing `f` for f-string in debug message in __init__ method.

CHAPTER

EIGHT

[0.1.0] - 2020-11-01

8.1 Added

- Initial pipeserial code and package
- Docs including this changelog
- The humble beginnings of a testsuite