term Documentation

Release 2.5

Stefan H. Holek

Sep 14, 2023

CONTENTS

The *term* module is an enhanced version of the standard library's tty module. It provides a set of functions and context managers for POSIX-style terminal programming.

See also:

Module termios
Low-level terminal control interface.

Xterm Control Sequences
Detailed list of escape sequences accepted by xterm.

CHAPTER

ONE

API DOCUMENTATION

CHAPTER

TWO

CONSTANTS

term.**IFLAG = 0**

Input modes. Index into list returned by tcgetattr.

term.**OFLAG = 1**

Output modes. Index into list returned by tcgetattr.

term.**CFLAG = 2**

Control modes. Index into list returned by tcgetattr.

term.**LFLAG = 3**

Local modes. Index into list returned by tcgetattr.

term.**ISPEED = 4**

Input speed. Index into list returned by tcgetattr.

term.**OSPEED = 5**

Output speed. Index into list returned by tcgetattr.

term.**CC = 6**

Control characters. Index into list returned by tcgetattr.

term.**TIMEOUT = 2**

The default read timeout in 1/10ths of a second.

CHAPTER

THREE

TERMINAL CONTROL

term.**setraw**(*fd*, *when=TCSAFLUSH*, *min=1*, *time=0*)

Put the terminal in raw mode.

Wait until at least *min* bytes or characters have been read. If *min* is 0, give up after *time* (in 1/10ths of a second) without data becoming available.

term.**setcbreak**(*fd*, *when=TCSAFLUSH*, *min=1*, *time=0*)

Put the terminal in cbreak mode.

Wait until at least *min* bytes or characters have been read. If *min* is 0, give up after *time* (in 1/10ths of a second) without data becoming available.

term.**rawmode**(*fd*, *when=TCSAFLUSH*, *min=1*, *time=0*)

Context manager to put the terminal in raw mode.

The current mode is saved and restored on exit.

term.**cbreakmode**(*fd*, *when=TCSAFLUSH*, *min=1*, *time=0*)

Context manager to put the terminal in cbreak mode.

The current mode is saved and restored on exit.

CHAPTER

FOUR

TERMINAL I/O

term.**opentty**(*bufsize=-1*, *mode='r+b'*)

Context manager returning a new rw stream connected to /dev/tty.

The stream is None if the device cannot be opened. The *mode* argument must be 'r+b' (default) or 'r+'.

term.**readto**(*stream*, *endswith*)

Read bytes or characters from *stream* until buffer.endswith(*endswith*) is true.

The *endswith* argument may be a single suffix or a tuple of suffixes to try. Suffixes must be bytes or str depending on the stream. Empty suffixes are ignored.

CHAPTER

FIVE

HIGH-LEVEL FUNCTIONS

These functions are implemented using the low-level facilities above and should probably live in a different package; yet here we are.

All functions may time out if the terminal does not respond. Set *term.TIMEOUT* to increase the timeout.

High-level functions are not included in `from term import *`.

term.**getyx**()

Return the cursor position as 1-based (line, col) tuple.

Line and col are 0 if the device cannot be opened or does not support DSR 6.

term.**getfgcolor**()

Return the terminal foreground color as (r, g, b) tuple.

All values are -1 if the device cannot be opened or does not support OSC 10.

term.**getbgcolor**()

Return the terminal background color as (r, g, b) tuple.

All values are -1 if the device cannot be opened or does not support OSC 11.

term.**islightmode**()

Return true if the background color is lighter than the foreground color.

May return None if the device cannot be opened or does not support OSC 10 & 11.

term.**isdarkmode**()

Return true if the background color is darker than the foreground color.

May return None if the device cannot be opened or does not support OSC 10 & 11.

CHAPTER

SIX

EXAMPLES

The getyx function may be implemented like this:

```
from re import search
from term import opentty, cbreakmode, readto

def getyx():
    with opentty() as tty:
        if tty is not None:
            with cbreakmode(tty, min=0, time=2):  # 0.2 secs
                tty.write(b'\033[6n')  # DSR 6
                p = readto(tty, b'R')  # expect b'\033[24;1R'
                if p:
                    m = search(b'(\\d+);(\\d+)R$', p)
                    if m is not None:
                        return int(m.group(1)), int(m.group(2))
    return 0, 0
```

Or with stdin/stdout and text I/O:

```
import sys

from re import search
from term import cbreakmode, readto

def getyx():
    with cbreakmode(sys.stdin, min=0, time=2):  # 0.2 secs
        sys.stdout.write('\033[6n')
        sys.stdout.flush()
        p = readto(sys.stdin, 'R')  # expect '\033[24;1R'
        if p:
            m = search(r'(\d+);(\d+)R$', p)
            if m is not None:
                return int(m.group(1)), int(m.group(2))
    return 0, 0
```

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX