Temp Monitor Documentation

Release 0.4.0

Goncalo Magno

Feb 22, 2020

Contents:

CHAPTER 1

Temp Monitor

A Python CLI utility to monitor cpu temperature

- Free software: MIT license
- Documentation: https://tmon.readthedocs.io.

1.1 Install

```
pip install tmonpy
```

or

```
$ # download the AppImage file from the releases page
$ wget https://github.com/gmagno/tmon/releases/latest/download/tmon-???????-x86_64.
↪AppImage
$ chmod +x tmon-*-x86_64.AppImage
```

1.2 Usage

```
tmon -h
usage: tmon [-h] [-v] [-y YSIZE] [-x XSIZE] [-l MIN MAX] ...

Temperature Monitor (tmon v0.3.6) -- executes a program while
```

(continues on next page)

(continued from previous page)

```
monitoring CPU temperature, reporting the min, max and mean
temperatures and plotting an ascii chart at the end to stderr.
All signals are redirected to the program.
If no program is passed, tmon runs as expected returning on
SIGINT (Ctrl-C).
For full documentation check the repo: https://github.com/gmagno/tmon

positional arguments:
CMD                   Command args to run.

optional arguments:
-h, --help            show this help message and exit
-v, --version         Shows tmon version.
-y YSIZE, --ysize YSIZE
                        Y-axis size in number terminal characters
-x XSIZE, --xsize XSIZE
                        X-axis size in number terminal characters
-l MIN MAX, --ylim MIN MAX
                        Y-axis view limits with min and max values. It is
                        ignored if the measured temperatures fall outside the
                        specified range.

return:
    tmon returns when the child program exits, stops, or is
    terminated by a signal. The return value of tmon is the return
    value of the program it executed.

examples:
    $ tmon echo How can a clam cram in a clean cream can
    How can a clam cram in a clean cream can

    ==================
    Temp Monitor Report:

    Temp (°C) for a period of 0:00:00
    >> 53.0 °C <<

    /tmp/tmon-YYYYMMDD@HHhMMmSS-XXXXXXXX.txt
    ==================

    $ tmon bash -c 'sleep 6; stress -c 4 -t 3; sleep 6'
    stress: info: [17832] dispatching hogs: 4 cpu, 0 io, 0 vm, 0 hdd
    stress: info: [17832] successful run completed in 3s

    ==================
    Temp Monitor Report:

    Temp (°C) for a period of 0:00:15
    59.00
    58.67
    58.33
    58.00
    57.67        _
    57.33        | |
    57.00        | |
    56.67
    56.33       |   |
```

(continues on next page)

(continued from previous page)

```
    56.00        |    |
    55.67  -     |    |
    55.33    |   |    |
    55.00    ___    ____
    54.67                |
    54.33                |
    54.00

    /tmp/tmon-YYYYMMDD@HHhMMmSS-XXXXXXXX.txt
    ====================

    $ tmon -y 5 -x 5 bash -c 'sleep 6; stress -c 4 -t 3; sleep 6'
    stress: info: [17181] dispatching hogs: 4 cpu, 0 io, 0 vm, 0 hdd
    stress: info: [17181] successful run completed in 3s

    ====================
    Temp Monitor Report:

    Temp (°C) for a period of 0:00:15
    60.00
    59.00
    58.00  |
    57.00   |
    56.00     -

    /tmp/tmon-YYYYMMDD@HHhMMmSS-XXXXXXXX.txt
    ====================

    $ tmon -l 40 70 -x 10 -y 10 bash -c 'stress -c 4 -t 3; sleep 6'
    stress: info: [19677] dispatching hogs: 4 cpu, 0 io, 0 vm, 0 hdd
    stress: info: [19677] successful run completed in 3s

    ====================
    Temp Monitor Report:

    Temp (°C) for a period of 0:00:09
    70.00
    66.67
    63.33
    60.00
    56.67  ___
    53.33     _____
    50.00
    46.67
    43.33
    40.00

    /tmp/tmon-YYYYMMDD@HHhMMmSS-XXXXXXXX.txt
    ====================

copyright:
    Copyright © 2020 Gonçalo Magno <goncalo@gmagno.dev>
    This software is licensed under the MIT License.
```

CHAPTER 2

Installation

2.1 Stable release

To install Temp Monitor, run this command in your terminal:

```
$ pip install tmonpy
```

This is the preferred method to install Temp Monitor, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

Alternatively a *tmon* binary (AppImage) may be downloaded from the releases page:

```
$ # replace ??????? by the actual hash
$ wget https://github.com/gmagno/tmon/releases/latest/download/tmon-???????-x86_64.
↪AppImage
$ chmod +x tmon-*-x86_64.AppImage
```

2.2 From sources

The sources for Temp Monitor can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/gmagno/tmon
```

Or download the tarball:

```
$ curl -OJL https://github.com/gmagno/tmon/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 3

Usage

To use Temp Monitor in a project:

```
tmon -h
usage: tmon [-h] [-v] [-y YSIZE] [-x XSIZE] [-l MIN MAX] ...

Temperature Monitor (tmon v0.3.6) -- executes a program while
monitoring CPU temperature, reporting the min, max and mean
temperatures and plotting an ascii chart at the end to stderr.
All signals are redirected to the program.
If no program is passed, tmon runs as expected returning on
SIGINT (Ctrl-C).
For full documentation check the repo: https://github.com/gmagno/tmon

positional arguments:
CMD                   Command args to run.

optional arguments:
-h, --help            show this help message and exit
-v, --version         Shows tmon version.
-y YSIZE, --ysize YSIZE
                        Y-axis size in number terminal characters
-x XSIZE, --xsize XSIZE
                        X-axis size in number terminal characters
-l MIN MAX, --ylim MIN MAX
                        Y-axis view limits with min and max values. It is
                        ignored if the measured temperatures fall outside the
                        specified range.

return:
    tmon returns when the child program exits, stops, or is
    terminated by a signal. The return value of tmon is the return
    value of the program it executed.

examples:
```

(continues on next page)

(continued from previous page)

```
$ tmon echo How can a clam cram in a clean cream can
How can a clam cram in a clean cream can

===================
Temp Monitor Report:

Temp (°C) for a period of 0:00:00
>> 53.0 °C <<

/tmp/tmon-YYYYMMDD@HHhMMmSS-XXXXXXXX.txt
===================

$ tmon bash -c 'sleep 6; stress -c 4 -t 3; sleep 6'
stress: info: [17832] dispatching hogs: 4 cpu, 0 io, 0 vm, 0 hdd
stress: info: [17832] successful run completed in 3s

===================
Temp Monitor Report:

Temp (°C) for a period of 0:00:15
59.00
58.67
58.33
58.00
57.67        _
57.33       | |
57.00       | |
56.67
56.33      |   |
56.00      |   |
55.67  _   |   |
55.33   |  |   |
55.00   ___  ____
54.67                |
54.33                |
54.00

/tmp/tmon-YYYYMMDD@HHhMMmSS-XXXXXXXX.txt
===================

$ tmon -y 5 -x 5 bash -c 'sleep 6; stress -c 4 -t 3; sleep 6'
stress: info: [17181] dispatching hogs: 4 cpu, 0 io, 0 vm, 0 hdd
stress: info: [17181] successful run completed in 3s

===================
Temp Monitor Report:

Temp (°C) for a period of 0:00:15
60.00
59.00
58.00  |
57.00   |
56.00     _

/tmp/tmon-YYYYMMDD@HHhMMmSS-XXXXXXXX.txt
===================
```

(continues on next page)

(continued from previous page)

```
    $ tmon -l 40 70 -x 10 -y 10 bash -c 'stress -c 4 -t 3; sleep 6'
    stress: info: [19677] dispatching hogs: 4 cpu, 0 io, 0 vm, 0 hdd
    stress: info: [19677] successful run completed in 3s

    ====================
    Temp Monitor Report:

    Temp (°C) for a period of 0:00:09
    70.00
    66.67
    63.33
    60.00
    56.67  ──╮
    53.33    ╰─────
    50.00
    46.67
    43.33
    40.00

    /tmp/tmon-YYYYMMDD@HHhMMmSS-XXXXXXXX.txt
    ====================

copyright:
    Copyright © 2020 Gonçalo Magno <goncalo@gmagno.dev>
    This software is licensed under the MIT License.
```

CHAPTER 4

tmon

4.1 tmon package

4.1.1 Submodules

4.1.2 tmon.asciichart module

This file was copied from https://github.com/kroitor/asciichart and slightly changed. For details please check the repo. It was copied because asciichartpy is missig from the anaconda channels/repos and I did not care to upload it...

tmon.asciichart.**plot**(*series*, *cfg=None*)

Possible cfg parameters are 'minimum', 'maximum', 'offset', 'height' and 'format'. cfg is a dictionary, thus dictionary syntax has to be used. Example: print(plot(series, { 'height' :10 }))

4.1.3 tmon.cli module

CLI script for tmon.

class tmon.cli.**AxisSizeAction**(*option_strings*, *dest*, *nargs=None*, *const=None*, *default=None*, *type=None*, *choices=None*, *required=False*, *help=None*, *metavar=None*)

Bases: argparse.Action

tmon.cli.**main**()

tmon.cli.**parse_args**()

4.1.4 tmon.tmon module

Main module.

class tmon.tmon.**Monitor**

Bases: object

start(*args*)

class tmon.tmon.**Report**(*ds*, *tfname*, *period*, *fahrenheit=False*)
Bases: object

Temperature Reporter

chart(*xsize*, *ysize*, *ylim*)

footer()

header()

report(*xsize*, *ysize*, *ylim*)

stats()

tmon.tmon.**run**(*cmd=None*, *xsize=70*, *ysize=15*, *ylim=None*, *fahrenheit=False*, *stats_only=False*, *chart_only=False*, *path_only=False*)

4.1.5 Module contents

Top-level package for Temp Monitor.

CHAPTER 5

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/gmagno/tmon/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

5.1.4 Write Documentation

Temp Monitor could always use more documentation, whether as part of the official Temp Monitor docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/gmagno/tmon/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.2 Get Started!

Ready to contribute? Here's how to set up *tmon* for local development.

1. Fork the *tmon* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/tmon.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv tmon
$ cd tmon/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 tmon tests
$ python setup.py test or pytest
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.5, 3.6, 3.7 and 3.8, and for PyPy. Check https://travis-ci.org/gmagno/tmon/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ pytest tests.test_tmon
```

5.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bump2version patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER 6

Credits

6.1 Development Lead

- Goncalo Magno <goncalo@gmagno.dev>

6.2 Contributors

None yet. Why not be the first?

CHAPTER 7

History

7.1 0.1.0 (2020-01-28)

- First release on PyPI.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

t

Index