Flockcontext Documentation

Release 0.3.1

Antoine Cezar

December 28, 2015

Contents

Contents:

CHAPTER 1

Flockcontext

Improves fcntl.flock usage.

`flock` is a Unix command for file locking, the mecanism that controls access restrictions of files.

1.1 Usage

Exclusive blocking lock:

```
from flockcontext import FlockOpen

with FlockOpen('/tmp/my.lock', 'w') as lock:
    lock.fd.write('Locked\n')
```

Exclusive non-blocking lock:

```
from flockcontext import FlockOpen

try:
    with FlockOpen('/tmp/my.lock', 'w', blocking=False) as lock:
        lock.fd.write('Locked\n')
except IOError as e:
    print('Can not acquire lock')
```

Shared blocking lock:

```
from flockcontext import Flock

with FlockOpen('/tmp/my.lock', 'w', exclusive=False) as lock:
    lock.fd.write('Locked\n')
```

Acquire and release within context:

```
from flockcontext import FlockOpen

with FlockOpen('/tmp/my.lock', 'w') as lock:
    print('Lock acquired')
    lock.fd.write('Locked\n')

    lock.release()
    print('Lock released')

    lock.acquire()
```

```
    print('Lock acquired')
    lock.fd.write('Locked\n')
```

Locking alredy opened file:

```
from flockcontext import Flock

with open('/tmp/my.lock', 'w') as fd:
    with Flock(fd):
        fd.write('Locked\n')
```

1.2 License

- Free software: BSD license

CHAPTER 2

Installation

At the command line:

```
$ pip install flockcontext
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv flockcontext
$ pip install flockcontext
```

CHAPTER 3

Usage

flockcontext provide two context manager for fcntl.flock. `Flock` locks an opened file while `FlockOpen` does the same job but opens the file for you.

3.1 FlockOpen exemples

Exclusive blocking lock:: python

    from flockcontext import FlockOpen

    with FlockOpen('/tmp/my.lock', 'w') as lock: lock.fd.write('Lockedn')

Exclusive non-blocking lock:: python

    from flockcontext import FlockOpen

    try:

        with FlockOpen('/tmp/my.lock', 'w', blocking=False) as lock: lock.fd.write('Lockedn')

    except IOError as e: print('Can not acquire lock')

Shared blocking lock:: python

    from flockcontext import Flock

    with FlockOpen('/tmp/my.lock', 'w', exclusive=False) as lock: lock.fd.write('Lockedn')

Acquire and release within context:: python

    from flockcontext import FlockOpen

    with FlockOpen('/tmp/my.lock', 'w') as lock: print('Lock acquired') lock.fd.write('Lockedn')

        lock.release() print('Lock released')

        lock.acquire() print('Lock acquired') lock.fd.write('Lockedn')

3.2 Flock exemples

Blocking lock:: python

    from flockcontext import Flock

    with open('/tmp/my.lock', 'w') as fd:

with Flock(fd): fd.write('Lockedn')

Non blocking lock:: python

from flockcontext import Flock

with open('/tmp/my.lock', 'w') as fd:

try:

with Flock(fd, blocking=False): fd.write('Lockedn')

except IOError as e: print('Can not acquire lock')

Shared lock:: python

from flockcontext import Flock

with open('/tmp/my.lock', 'w') as fd:

with Flock(fd, exclusive=False): fd.write('Lockedn')

Acquire and release within context:: python

from flockcontext import Flock

with open('/tmp/my.lock', 'w') as fd:

with Flock(fd) as lock: print('Lock acquired') fd.write('Lockedn')

lock.release() print('Lock released')

lock.acquire() print('Lock acquired') fd.write('Lockedn')

CHAPTER 4

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:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/AntoineCezar/flockcontext/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.

4.1.2 Fix Bugs

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

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

4.1.4 Write Documentation

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

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/AntoineCezar/flockcontext/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 :)

4.2 Get Started!

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

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

```
$ git clone git@github.com:your_name_here/flockcontext.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 flockcontext
$ cd flockcontext/
$ 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 flockcontext tests
$ python setup.py test
$ 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.

4.3 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 2.6, 2.7, 3.3, and 3.4, and for PyPy. Check https://travis-ci.org/AntoineCezar/flockcontext/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_flock
```

CHAPTER 5

Credits

5.1 Development Lead

- Antoine Cezar <antoine@cezar.fr>

5.2 Contributors

- Mathieu Leplatre <contact@mathieu-leplatre.info>

CHAPTER 6

History

CHAPTER 7

0.3.1 (2015-08-24)

- Add syntax highlighting for code exemples
- Add Flock manager exemple in README

CHAPTER 8

0.3.0 (2015-08-21)

- Add FlockOpen context manager.

CHAPTER 9

0.2.0 (2015-08-20)

- Add Flock relase and acquire capability withing context.

CHAPTER 10

0.1.0 (2015-08-19)

- Add Flock context manager.

CHAPTER 11

Indices and tables

- genindex
- modindex
- search