lockmgr

Release 1.3.1

Gene C

Mar 29, 2024

CONTENTS:

CHAPTER

ONE

LOCKMGR

1.1 Overview

lockmgr : Python Class implementing file locking

1.2 New / Interesting

Fun new info goes here

CHAPTER

TWO

USING LOCKMGR CLASS

Using the class is straightforward. Choose a file to use as the lockfile - avoid NFS mounts. I find using */tmp* works well since its a *tmpfs* file system.

An sample application is available in the examples directory. To see it in action, simply run *test_lock.py* in 2 different terminals. One process will acquire the lock while the other will wait until its released.

Typical usage:

```
import os
from lockmgr import LockMgr

    lockdir = '/tmp/xxx'
    os.makedirs(lockdir, exist_ok=True)
    lockfile = os.path.join(lockdir, 'foo.lock')
    lockmgr = LockMgr(lockfile)

    ...

    # Acquire lock
    if lockmgr.acquire_lock(wait=True, timeout=30):
        # do stuff
        ...

        # release lock
        lockmgr.release_lock()
   else:
        # failed to get lock
        ...
```

CHAPTER

THREE

RESEARCH ON FILE LOCKING

I share my notes and research on file locking. All code samples are available in the examples/research directory.

3.1 Linux: C Implementations

File Locking has linux kernel support via standard C library using fcntl(). The mechanism uses 'struct flock' as the communication mechanism with (posix) fcntl().

The standard locking mechanism uses F_SETLK. This lock lives at the process level and is the original locking in linux. Around 2015 or so 'open file desription' locks came to be via F_OFD_SETLK (See[1] and[2]). These are at the open file level. So while F_SETLK is not passed to child processes, OFD locks are. And OFD locks remain attached to the open file handle. This can be enormously useful and also surprising.

Posix locks are attached to (inode, pid) pair which means they work at the process level. If thread opens same file, even tho has different file handle, when that thread closes the fd, the lock is released for all threads in same process.

OFD locking was introduced to deal with this. Locking is attached to the "open file" and not the PID.

linux also provides "lockf" which is a wrapper around fcntl() locks - should only be used in simple cases and will interact with "normal" fcntl() locks - caveat emptor.

Summary:

- Posix Lock: fcntl() - F_SETLK
- OFD Lock: fcntl() - F_OFD_SETLK
- lockf - bah

NB. The flock struct contains l_pid - this MUST be 0 for OFD locks and PID for posix locks.

Its also worth noting that locking can be file system dependent. In particular NFS should probably be avoided. Since my dominant use case is single machine, multiple process, I use /tmp which is a TMPFS file system and works well.

[1] File private locks https://lwn.net/Articles/586904/
[2] Open File Description https://lwn.net/Articles/640404/

3.2 Python

Python provides for same locking mechanisms - I recommend only 1 way for file locks in python. Python library provides for:

- fcntl.fcntl => do not use

As with C there is support for F_SETLK and F_OFD_SETLK. While these work fine, they require using 'struct' module to 'pack' and 'unpack' the C flock struct. To make this work the caller must provide the sizes (coded with letters as per the python struct module) of each element being packed.

The python 3.12 docs have examples[3], and while they may well work for a Sun workstation or similar, if you have one, the struct element sizes dont seem correct for X86_64.

I provide a little C-program to print out the correct byte sizes which you can then map to the python struct letter codes[4]

This approach is brittle - its one thing when you are coding with your own C structures, its another entirely when using system ones - these sizes should be compiled into python - while these routines work I strongly recommend not using them for this reason.

- fcntl.lockf => do not use

Wrapper around fcntl() - in spite of name this is NOT C lockf() function.

- fcntl.flock => *use this one*

Wrapper around fcntl with OFD support. i.e. this lock is associated with open file descriptor. This is what I use and recommend.

3.3 Examples

3.4 C-code

Sample code for F_SETLK and F_OFD_SETLK To compile:

```
make
```

Builds 2 programs - *flock_sizes* and *c_lock_test*.

flock_sizes is used To print size of struct flock elements which provide the correct sizes to use in python fcntl.fcntl approach.

```
./flock_sizes
```

The test program demonstrates locking with and without OFD. To run the test progrm see the *Tests: c_lock_test* section below.

[3] Python fcntl docs: https://docs.python.org/3/library/fcntl.html
[4] Python struct module: https://docs.python.org/3/library/struct.html

3.5 Tests: c_lock_test

To run locking tests, use 2 terminals. Run c_lock_test in both. The first will acquire lock while second will fail until first exits or is interrupted.

3.5.1 Test 1 : Using F_SETLK

```
./c_lock_test
```

3.5.2 Test 2 : Using F_OFD_SETLK

Repeat test but with argument to turn on OFD

```
./c_lock_test ofd
```

Test (1) and (2) both work.

3.6 Python : lock_fcntl

F_SETLK and F_OFD_SETLK tests in python. Run test in 2 terminals as above:

3.6.1 Test 3 : Using F_SETLK

```
./lock_fcntl.py
```

3.6.2 Test 4 : Using F_OFD_SETLK

```
./lock_fcntl.py ofd
```

Test (3) and (4) both work.

3.7 Python : lock_flock

This is what I am using. As above, run test in 2 terminals.

3.7.1 Test 5 :

```
./lock_fcntl.py
```

Test (5) works.

CHAPTER

FOUR

APPENDIX

4.1 Installation

Available on

- Github
- Archlinux AUR

On Arch you can build using the provided PKGBUILD in the packaging directory or from the AUR. To build manually, clone the repo and :

```
rm -f dist/*
/usr/bin/python -m build --wheel --no-isolation
root_dest="/"
./scripts/do-install $root_dest
```

When running as non-root then set root_dest a user writable directory

4.2 Dependencies

- Run Time :
  - python (3.11 or later)
- Building Package:
  - git
  - hatch (aka python-hatch)
  - wheel (aka python-wheel)
  - build (aka python-build)
  - installer (aka python-installer)
  - rsync
- Optional for building docs:
  - sphinx
  - texlive-latexextra (archlinux packaguing of texlive tools)

4.3 Philosophy

We follow the *live at head commit* philosophy. This means we recommend using the latest commit on git master branch. We also provide git tags.

This approach is also taken by Google[5][6].

4.4 License

[5] https://github.com/google/googletest
[6] https://abseil.io/about/philosophy#upgrade-support

CHAPTER

FIVE

CHANGELOG

[1.3.1] —— 2024-03-29

```
README fix rst title level
```

[1.3.0] —— 2024-03-29

```
Public release along with lockfile research and code
Initial commit
```

CHAPTER

SIX

MIT LICENSE

CHAPTER

SEVEN

HOW TO HELP WITH THIS PROJECT

Thank you for your interest in improving this project. This project is open-source under the MIT license.

7.1 Important resources

- Git Repo

7.2 Reporting Bugs or feature requests

Please report bugs on the issue tracker in the git repo. To make the report as useful as possible, please include

- operating system used
- version of python
- explanation of the problem or enhancement request.

7.3 Code Changes

If you make code changes, please update the documentation if it's appropriate.

CHAPTER

EIGHT

CONTRIBUTOR COVENANT CODE OF CONDUCT

8.1 Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, sex characteristics, gender identity and expression, level of experience, education, socio-economic status, nationality, personal appearance, race, religion, or sexual identity and orientation.

8.2 Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Trolling, insulting/derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

8.3 Our Responsibilities

Maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

8.4 Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

8.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the project team at <arch@sapience.com>. All complaints will be reviewed and investigated and will result in a response that is deemed necessary and appropriate to the circumstances. The Code of Conduct Committee is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

8.6 Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 1.4, available at https://www.contributor-covenant.org/version/1/4/code-of-conduct.html

8.7 Interpretation

The interpretation of this document is at the discretion of the project team.

CHAPTER

NINE

INDICES AND TABLES

- genindex
- modindex
- search