
lockmgr

Release 1.7.0

Gene C

May 22, 2025

CONTENTS:

1	lockmgr	1
1.1	Overview	1
1.2	New / Interesting	1
2	Using LockMgr Class	3
3	Research on file locking	5
3.1	Linux: C Implementations	5
3.2	Python	5
3.3	Examples	6
3.4	C-code	6
3.5	Tests: c_lock_test	6
3.6	Python : lock_fcntl	7
3.7	Python : lock_flock	7
4	Appendix	9
4.1	Installation	9
4.2	Dependencies	9
4.3	Philosophy	10
4.4	License	10
5	Changelog	11
5.1	Tags	11
5.2	Commits	11
6	MIT License	13
7	How to help with this project	15
7.1	Important resources	15
7.2	Reporting Bugs or feature requests	15
7.3	Code Changes	15
8	Contributor Covenant Code of Conduct	17
8.1	Our Pledge	17
8.2	Our Standards	17
8.3	Our Responsibilities	17
8.4	Scope	18
8.5	Enforcement	18
8.6	Attribution	18
8.7	Interpretation	18

9	API Reference	19
9.1	lockmgr	19
	Python Module Index	21
	Index	23

LOCKMGR

1.1 Overview

lockmgr : Python Class implementing file locking

All git tags are signed with arch@sapience.com key which is available via WKD or download from <https://www.sapience.com/tech>. Add the key to your package builder gpg keyring. The key is included in the Arch package and the source= line with *?signed* at the end can be used to verify the git tag. You can also manually verify the signature

1.2 New / Interesting

- Tidy ups: PEP-8, PEP-257, PEP-484 PEP-561
- improve reference API doc.
- Add py.typed so type checkers like mypy can be used with the module.

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
    ...
```


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 description' locks came to be via `F_OFD_SETLK` (See¹ and²). 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.

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.

¹ File private locks <https://lwn.net/Articles/586904/>

² Open File Description <https://lwn.net/Articles/640404/>

The python 3.12 docs have examples³, 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 codes⁴

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 program see the *Tests: `c_lock_test`* section below.

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 Python `fcntl` docs: <https://docs.python.org/3/library/fcntl.html>

⁴ Python `struct` module: <https://docs.python.org/3/library/struct.html>

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.

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 Google⁵⁶.

4.4 License

Created by Gene C. and licensed under the terms of the MIT license.

- SPDX-License-Identifier: MIT
- Copyright (c) 2023 Gene C

⁵ <https://github.com/google/googletest>

⁶ <https://abseil.io/about/philosophy#upgrade-support>

CHANGELOG

5.1 Tags

1.3.0 (2024-03-29) -> 1.7.0 (2025-05-21)
21 commits.

5.2 Commits

- 2025-05-21 : 1.7.0

2025-05-02 Small **type** hint change
update Docs/Changelog.rst Docs/lockmgr.pdf

- 2025-05-02 : 1.6.1

2025-02-26 Tidy ups: PEP-8, PEP-257, PEP-484 PEP-561
improve reference API doc.
Add py.typed so **type** checkers like mypy can be used **with** the module.
update Docs/Changelog.rst Docs/lockmgr.pdf

- 2025-02-26 : 1.5.4

2024-12-31 PKGBUILD: Add missing makedepends on python-installer (per aur comment by
@piater)
update Docs/Changelog.rst Docs/lockmgr.pdf

- 2024-12-31 : 1.5.3

2024-12-22 Git tags are now signed.
Update SPDX tags
Add git signing key to Arch Package
update Docs/Changelog.rst Docs/lockmgr.pdf

- 2024-12-22 : 1.5.2

Use autodoc style parameter descriptions **for** API docs
update Docs/Changelog.rst Docs/lockmgr.pdf

- 2024-12-22 : 1.5.1

2024-10-19 Add API reference to documentation
update Docs/Changelog.rst Docs/lockmgr.pdf

- 2024-10-19 : 1.5.0

2024-03-29 remove unused requirements file
update Docs/Changelog.rst Docs/lockmgr.pdf

- 2024-03-29 : 1.4.0

Change arch PKGBUILD dependency to python-pynotify
update Docs/Changelog.rst Docs/lockmgr.pdf

- 2024-03-29 : 1.3.2

Fix typo **in** PKGBUILD depends
update Docs/Changelog.rst Docs/lockmgr.pdf

- 2024-03-29 : 1.3.1

README fix rst title level

- 2024-03-29 : 1.3.0

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

MIT LICENSE

Copyright © 2023 Gene C

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

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.

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.

API REFERENCE

This page contains auto-generated API reference documentation¹.

9.1 lockmgr

locking

9.1.1 Submodules

lockmgr.class_lock

File locks

Module Contents

class **LockMgr**(*lockfile*)

Robust file locking manager.

acquire_lock(*wait: bool = False, timeout: int = 30*) → bool

Try to acquire a lock.

Args:

wait (bool):

If True and timeout > 0: wait until lock is acquired (up to 10 attempts). This can be racy from the time inotify returns till we acquire lock will fail and we will try again. If False, then do not retry if unable to acquire lock on first attempt.

timeout (int): Number of seconds > 0 to wait between attempts to acquire the lock Will retry up to 10 times.

Returns: bool: True if lock acquired

release_lock()

Release an Acquired Lock

Drop the acquired lock. No-op if there is no acquired lock.

¹ Created with sphinx-autoapi

PYTHON MODULE INDEX

|

`lockmgr`, [19](#)

`lockmgr.class_lock`, [19](#)

INDEX

A

`acquire_lock()` (*LockMgr* method), 19

L

`lockmgr`

 module, 19

`LockMgr` (class in `lockmgr.class_lock`), 19

`lockmgr.class_lock`

 module, 19

M

module

`lockmgr`, 19

`lockmgr.class_lock`, 19

R

`release_lock()` (*LockMgr* method), 19