pynotify

Release 1.5.0

Gene C

May 02, 2025

CONTENTS:

CHAPTER

ONE

PYNOTIFY

1.1 Overview

pynotify : Python inotify implementation built atop standard C-library.

1.2 Key features

- Provides a python class to monitor 1 or more file paths for inotify events.
- 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

1.3 New / Interesting

- Tidy ups: PEP-8, PEP-257, PEP-484 PEP-561
- improve reference API doc.

CHAPTER

TWO

GETTING STARTED

A simple test program is available in the examples directory. The program monitors */tmp/xxx* - this can be a file or a directory. To get a quick idea in one terminal do:

```
mkdir /tmp/xxx
./examples/test_inotify.py
```

In another terminal do something like:

```
touch /tmp/xxx/A
touch /tmp/xxx/B
rm /tmp/xxx/*
rmdir /tmp/xxx
```

What the code does is essentialy:

```
from pynotify import Inotify
inotify = Inotify()
inotify.add_watch('/tmp/xxx')

for events in inotify.get_events():
     for event in events:
         if event:
             print(..)
```

The first loop uses get_events() method which is an iterator that returns a list of events. Each even in the list provides for:

```
event.wd            # the watch descriptor
event.mask          # the event mask
event.event_types   # list of event type enums from the mask
event.path          # the path being watched (/tmp/xxx)
event.file          # Possible suppordinate file (/tmp/xxx/A)
```

Thats it in a nutshell. add_watch takes one optional argument:

```
Inotify.add_watch(*path*, mask=xxx)
```

add_watch() returns the watched descriptor, *wd*. If successful wd is >= 0. If < 0 then it means the path is non-existent.

If not provided then the watch is for all possible event types. Where mask are event types from the enum *Inotify-Mask.IN_xxx*. These event elements are the same as provided in */usr/include/sys/inotify.h*.

Before calling get_events() there is one additionl attribute that can be set on an instance of Inotify which is a select timeout which defaults to 5 seconds.

```
inotify.timeout
```

The timeout is passed down to *select()* which waits on the inotify file desriptor for events to provided. If its negative then the select will wait forever, if no events occur. Otherwise the select loop will break after the timeout. A value of zero causes select to return immediately. The default value should provide reasonable behaviour.

2.1 Mask Flags

You can get the full list of possible mask flags reading code, which has comments, or using:

```
from pynotify import InotifyMask, Inotify
for item in InotifyMask.mask_to_events(0xFFFFFFFF):
    item
```

This currently outputs the following where we have manually added comments:

```
<InotifyMask.IN_ACCESS: 1>              # File was accessed
<InotifyMask.IN_MODIFY: 2>              # File was modified.
<InotifyMask.IN_ATTRIB: 4>              # Metadata changed.
<InotifyMask.IN_CLOSE_WRITE: 8>         # Writtable file was closed.
<InotifyMask.IN_CLOSE_NOWRITE: 16>      # Unwrittable file closed.
<InotifyMask.IN_CLOSE: 24>              # File closed
<InotifyMask.IN_OPEN: 32>               # File was opened.
<InotifyMask.IN_MOVED_FROM: 64>         # File was moved from X.
<InotifyMask.IN_MOVED_TO: 128>          # File was moved to Y.
<InotifyMask.IN_MOVE: 192>              # File was moved
<InotifyMask.IN_CREATE: 256>            # Subfile was created.
<InotifyMask.IN_DELETE: 512>            # Subfile was deleted.
<InotifyMask.IN_DELETE_SELF: 1024>      # Self was deleted.
<InotifyMask.IN_MOVE_SELF: 2048>        # Self was moved.
<InotifyMask.IN_UNMOUNT: 8192>          # Backing fs was unmounted.
<InotifyMask.IN_Q_OVERFLOW: 16384>      # Event queue overflowed.
<InotifyMask.IN_IGNORED: 32768>         # File was ignored.
<InotifyMask.IN_ONLYDIR: 16777216>      # Only watch the path if it is a directory.
<InotifyMask.IN_DONT_FOLLOW: 33554432>  # Do not follow a sym link.
<InotifyMask.IN_EXCL_UNLINK: 67108864>  # Exclude events on unlinked objects.
<InotifyMask.IN_MASK_CREATE: 268435456> # Only create watches.
<InotifyMask.IN_MASK_ADD: 536870912>    # Add to the mask of an already existing watch.
<InotifyMask.IN_ISDIR: 1073741824>      # Event occurred against dir.
<InotifyMask.IN_ONESHOT: 2147483648>    # Only send event once.
<InotifyMask.IN_ALL_EVENTS: 4095>       # All events which that can be waited on.
```

CHAPTER

THREE

APPENDIX

3.1 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

3.2 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)

3.3 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[1][2].

3.4 License

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

CHAPTER

FOUR

CHANGELOG

4.1 Tags

```
1.2.1 (2024-03-29) -> 1.5.0 (2025-05-02)
22 commits.
```

4.2 Commits

- 2025-05-02 : **1.5.0**

```
              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
2024-12-31    update Docs/Changelog.rst Docs/pynotify.pdf
```

- 2024-12-31 : **1.3.0**

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

- 2024-12-22 : **1.2.9**

```
              Add API Reference to documentation
2024-10-19    update Docs/Changelog.rst Docs/pynotify.pdf
```

- 2024-10-19 : **1.2.8**

```
              remove unused requirements file
2024-09-07    update Docs/Changelog.rst Docs/pynotify.pdf
```

- 2024-09-07 : **1.2.7**

```
more rst tweaks
update Docs/Changelog.rst Docs/pynotify.pdf
```

- 2024-09-07 : **1.2.6**

```
              minor rst tweaks to readme
2024-09-05    update Docs/Changelog.rst Docs/pynotify.pdf
```

- 2024-09-05 : **1.2.5**

```
              Arch PKGBUILD : add missing dependency python-installer
                  Thanks to @Bonnietwin via https://aur.archlinux.org/packages/python-
                  pynotify#comment-989452
2024-03-29    update Docs/Changelog.rst Docs/pynotify.pdf
```

- 2024-03-29 : **1.2.4**

```
Installer takes optional package name - use in PKGBUILD
update Docs/Changelog.rst Docs/pynotify.pdf
```

- 2024-03-29 : **1.2.3**

```
typo in readme.rst
update Docs/Changelog.rst Docs/pynotify.pdf
```

- 2024-03-29 : **1.2.2**

```
Fix README.rst title level
update Docs/Changelog.rst Docs/pynotify.pdf
```

- 2024-03-29 : **1.2.1**

```
Public release
Initial commit - python Inotify class
```

CHAPTER

FIVE

MIT LICENSE

CHAPTER

SIX

HOW TO HELP WITH THIS PROJECT

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

6.1 Important resources

- Git Repo

6.2 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.

6.3 Code Changes

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

CHAPTER

SEVEN

CONTRIBUTOR COVENANT CODE OF CONDUCT

7.1 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.

7.2 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

7.3 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.

7.4 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.

7.5 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.

7.6 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

7.7 Interpretation

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

CHAPTER

EIGHT

API REFERENCE

This page contains auto-generated API reference documentation[1].

8.1 pynotify

pynotify public

8.1.1 Submodules

pynotify.class_inotify

Wrap libc inotify
see man inotify et al for details

Module Contents

`class Inotify`

Python inotify class.

- makes use of standard C-lib inotify

`add_watch`(*path: str*, *mask: int = InotifyMask.IN_ALL_EVENTS*) → int

Adds a watch to filesystem path.

Args:
path (str): The filesystem path on which to set the watch path must exist as can only use inotify existing file.

mask (InotifyMask): A bit mask from sys/inotify.h (see InotifyMask) A mask can OR together all items to watch for (m1 | m2 . . .).

Returns (int): Watch descriptor, wd. If non-existent file then returns -1

`save_errno`()

Keep the last errno

`rm_watch`(*path: str*)

Remove watch on this path.

Args:
path (str): The filesystem path to remove the watch for.

[1] Created with sphinx-autoapi

`rm_all_watches()`

Remove all current watches.

`get_events()` → Iterator[List[*pynotify.events.InotifyEvent*]]

Wait for events

Args:

Returns (Iterator[List[InotifyEvent]]):
Provides an iterater until fd is closed. fd is closed when the inotify event says so. if timeout < 0, wait forever.

pynotify.class_mask

Constants taken from sys/inotify.h

Module Contents

`class InotifyMask`

IN_xxx mask values from /usr/include/sys/inotify.h

`classmethod mask_to_events(`*mask*`)`

Return list of all events in mask

pynotify.events

Wrap libc inotify
see man inotify et al for details

Module Contents

`class InotifyEvent`

one inotify event

`mask_to_event_types(`*mask: int*`)`

From mask => return list of event types

`get_inotify_events(`*inotify*`)` → Iterator[List[*InotifyEvent*]]

wait for events

- provide iterater until fd is closed
- fd is closed when the inotify event says so.
- if timeout < 0, wait forever

PYTHON MODULE INDEX

p

INDEX