
pynotify
Release 1.7.1

Gene C

Jan 06, 2026

CONTENTS

1	pynotify	1
1.1	Overview	1
1.2	Key features	1
1.3	New / Interesting	1
2	Getting Started	3
2.1	Mask Flags	4
3	Appendix	5
3.1	Installation	5
3.2	Dependencies	5
3.3	Philosophy	6
3.4	License	6
4	License	7
5	API Reference	9
5.1	pynotify	9
	Python Module Index	13
	Index	15

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

1.7.0

- Switch packaging from hatch to uv

Older

- License switch to GPL-2.0-or-later.
- Improve API reference documentation
- 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 essentially:

```
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 subordinate 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 descriptor 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>            # Writable file was closed.
<InotifyMask.IN CLOSE_NOWRITE: 16>          # Unwritable 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.
```

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 (3.13 or later)

Building Package :

- git
- uv
- uv-build (aka python-uv-build)
- rsync
- pytest (aka python-pytest)
- pytest-asyncio (aka python-pytest-asyncio)

Optional for building docs :

- sphinx
- texlive-latexextra (archlinux packaguing of texlive tools)

3.3 Philosophy

We follow the *live at head commit* philosophy as recommended by Google's Abseil team¹. This means we recommend using the latest commit on git master branch.

3.4 License

Created by Gene C. and licensed under the terms of the GPL-2.0-or-later license.

- SPDX-License-Identifier: GPL-2.0-or-later
- SPDX-FileCopyrightText: © 2023-present Gene C <arch@sapience.com>

¹ <https://abseil.io/about/philosophy#upgrade-support>

**CHAPTER
FOUR**

LICENSE

pynotify module provides a python Inotify class.

Copyright © 2022-present Gene C <arch@sapience.com>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

API REFERENCE

This page contains auto-generated API reference documentation¹.

5.1 pynotify

pynotify public

5.1.1 Submodules

pynotify.class_inotify

Wrap around the standard libc library inotify.

See also standard man pages for inotify for more detail.

Module Contents

class Inotify

Python inotify class.

Uses *inotify* from the standard C-library.

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):

The mask specifies which events to watch for.

A mask is made by bitwise OR of one or more masks. Each mask item is taken from *InotifyMask*.

For example: * mask = (InotifyMask.OPEN | InotifyMask.CLOSE). * See InotifyMask for the full list.

Returns:

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

save_errno ()

Keep the last errno

¹ Created with sphinx-autoapi

`rm_watch(path: str)`

Remove watch on this path.

Args:

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

`rm_all_watches()`

Remove all current watches.

`get_events() → collections.abc.Iterator[list[pynotify.events.InotifyEvent]]`

Wait for events.

Args:

Returns:

Iterator[list[InotifyEvent]]: Provides an iterator 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`

Mask values from /usr/include/sys/inotify.h.

Masks can be bitwise *OR* together. For example IN_CLOSE can be written as (IN_CLOSE_WRITE | IN_CLOSE_NOWRITE).

Available masks:

- **IN_ACCESS : File was accessed**
- **IN_MODIFY : File was modified**
- **IN_ATTRIB : Metadata changed**
- **IN_CLOSE_WRITE : Writable file was closed**
- **IN_CLOSE_NOWRITE : Unwritable file closed**
- **IN_CLOSE : Closed write or nowrite**
- **IN_OPEN : File was opened**
- **IN_MOVED_FROM : File was moved from X**
- **IN_MOVED_TO : File was moved to Y**
- **IN_MOVE : Moved from or to**
- **IN_CREATE : Subfile was created**
- **IN_DELETE : Subfile was deleted**
- **IN_DELETE_SELF : Self was deleted**
- **IN_MOVE_SELF : Self was moved**

Events sent by the kernel.

- **IN_UNMOUNT : Backing fs was unmounted**
- **IN_Q_OVERFLOW : Event queued overflowed**

- IN_IGNORED : **File was ignored**

Special flags.

- IN_ONLYDIR : **Only watch the path if it is a directory**
- IN_DONT_FOLLOW : **Do not follow a sym link**
- IN_EXCL_UNLINK : **Exclude events on unlinked objects**
- IN_MASK_CREATE : **Only create watches**
- IN_MASK_ADD : **Add to mask of an existing watch**
- IN_ISDIR : **Event occurred against dir**
- IN_ONESHOT : **Only send event once**

All events which a program can wait on.

- IN_ALL_EVENTS : **All events**

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) → collections.abc.Iterator[list[InotifyEvent]]

wait for events

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

PYTHON MODULE INDEX

p

`pynotify`, 9
`pynotify.class_inotify`, 9
`pynotify.class_mask`, 10
`pynotify.events`, 11

INDEX

A

`add_watch()` (*Inotify method*), 9

G

`get_events()` (*Inotify method*), 10

`get_inotify_events()` (*in module pynotify.events*), 11

I

`Inotify` (*class in pynotify.class_inotify*), 9

`InotifyEvent` (*class in pynotify.events*), 11

`InotifyMask` (*class in pynotify.class_mask*), 10

M

`mask_to_event_types()` (*in module pynotify.events*),
11

`mask_to_events()` (*InotifyMask class method*), 11

`module`

`pynotify`, 9

`pynotify.class_inotify`, 9

`pynotify.class_mask`, 10

`pynotify.events`, 11

P

`pynotify`

`module`, 9

`pynotify.class_inotify`

`module`, 9

`pynotify.class_mask`

`module`, 10

`pynotify.events`

`module`, 11

R

`rm_all_watches()` (*Inotify method*), 10

`rm_watch()` (*Inotify method*), 9

S

`save_errno()` (*Inotify method*), 9