

---

# ladang Documentation

*Release 0.9.0*

**E A Faisal <eafaisal at gmail dot com>**

January 15, 2013



# CONTENTS

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	What is ladang? . . . . .	3
1.2	Installation . . . . .	3
<b>2</b>	<b>Tutorial: ladang - inotify Made Easy</b>	<b>5</b>
2.1	Step One . . . . .	5
2.2	Step Two . . . . .	5
2.3	Step Three . . . . .	5
2.4	Step Four . . . . .	6
2.5	Step Five . . . . .	6
2.6	Step Six . . . . .	6
<b>3</b>	<b>ladang - High Level API</b>	<b>7</b>
3.1	Inotify event mask constants . . . . .	7
<b>4</b>	<b>_ladang - Low Level API</b>	<b>9</b>
4.1	add_watch() flag constants . . . . .	9
<b>5</b>	<b>Indices and tables</b>	<b>11</b>



Contents:



# INTRODUCTION

## 1.1 What is ladang?

A Python module which provides a very thin layer binding to inotify API supporting both blocking and non-blocking operation.

## 1.2 Installation

Create virtualenv environment. This step is optional and requires virtualenv installed.

```
$ python virtualenv ENV
$ cd ENV
$ source bin/activate
```

Fork/clone ladang repository locally from GitHub. This step requires git installed.

```
$ git clone https://github.com/efaisal/ladang.git
```

Runs setup.py.

```
$ cd ladang
$ python setup install
```



# LICENSE

ladang is licensed under the MIT license.

## 2.1 The MIT License

Copyright (c) 2013 E A Faisal

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.



# TUTORIAL: LADANG - INOTIFY MADE EASY

You are only 6 steps away to fully utilize ladang. In this tutorial we will monitor a directory for all possible inotify events.

## 3.1 Step One

Import the module.

```
import ladang
```

## 3.2 Step Two

Instantiate `ladang.Ladang()` object.

```
inotify = ladang.Ladang() # This call is for blocking get_event()
```

or

```
inotify = ladang.Ladang(ladang.NONBLOCK|ladang.CLOEXEC) # This call for non-blocking get_event()
```

## 3.3 Step Three

Next we register a file or a directory which we are interested to monitor and event which we wish the kernel to notify us. By default all events will be reported.

```
inotify.watch('/path/to/mydir', ladang.IN_ALL_EVENTS)
```

The bit mask `ladang.IN_ALL_EVENTS` indicates we are interested in all events. There are many other masks available which can be bitwise ORed. See *Inotify event mask constants* for other possible values.

## 3.4 Step Four

If any of the events we are interested in occur, `inotify` will put it into an event queue. All we have to do is fetch the occurred events from the queue. First create a file in `/path/to/mydir` so that there is a filesystem event happen. Execute from the shell:

```
$ touch /path/to/mydir/myfile
```

Then we can pull the events from the queue by doing:

```
events = inotify.get_event()
```

If there is no `inotify` event, `inotify.get_event()` returns an empty tuple. Else we can do this to print out the events:

```
for event in events:
    print("Watch description: %d" % evt['wd'])
    print("Mask: %d" % evt['mask'])
    print("Mask descr: %s => %s" % ladang.INOTIFY_MASKS[evt['mask']])
    print("Cookie: %d" % evt['cookie'])
    print("Name: %s" % evt['name'].strip("\0"))
```

A word of cautious, if you are employing a multithreaded strategy, it is important to employ a proper locking. This is because internally, the watched file descriptor will be shared across all thread.

## 3.5 Step Five

When you are no longer interested to monitor any event, just do:

```
inotify.unwatch('/path/to/dir')
```

## 3.6 Step Six

And when you're done, simply call the `inotify.close()` method to close the controlling `inotify` file descriptor.

```
inotify.close(notify_fd)
```

# LADANG - HIGH LEVEL API

## 4.1 Inotify event mask constants

### **IN\_ACCESS**

File was accessed.

### **IN\_MODIFY**

File was modified.

### **IN\_ATTRIB**

Metadata changed.

### **IN\_CLOSE\_WRITE**

Writtable file was closed.

### **IN\_CLOSE\_NOWRITE**

Unwrittable file closed.

### **IN\_CLOSE**

Equivalent to `ladang.IN_CLOSE_WRITE | ladang.IN_CLOSE_NOWRITE`

### **IN\_OPEN**

File was opened.

### **IN\_MOVED\_FROM**

File was moved from X.

### **IN\_MOVED\_TO**

File was moved to Y.

### **IN\_MOVE**

Equivalent to `ladang.IN_MOVED_FROM | ladang.IN_MOVED_TO`

### **IN\_CREATE**

Subfile was created.

### **IN\_DELETE**

Subfile was deleted.

### **IN\_DELETE\_SELF**

Self was deleted.

### **IN\_MOVE\_SELF**

Self was moved.

### **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\_ADD**

Add to the mask of an already existing watch.

**IN\_ISDIR**

Event occurred against dir.

**IN\_ONESHOT**

Only send event once.

**IN\_ALL\_EVENTS**

All events which can be waited on.

Equivalent to:

ladang.IN\_ACCESS | ladang.IN\_MODIFY | ladang.IN\_ATTRIB | ladang.IN\_CLOSE\_WRITE |  
ladang.IN\_CLOSE\_NOWRITE | ladang.IN\_OPEN | ladang.IN\_MOVED\_FROM | ladang.IN\_MOVED\_TO |  
ladang.IN\_CREATE | ladang.IN\_DELETE | ladang.IN\_DELETE\_SELF | ladang.IN\_MOVE\_SELF

# **\_LADANG - LOW LEVEL API**

## **5.1 add\_watch() flag constants**

**IN\_ACCESS**

File was accessed.

**IN\_MODIFY**

File was modified.

**IN\_ATTRIB**

Metadata changed.

**IN\_CLOSE\_WRITE**

Writtable file was closed.

**IN\_CLOSE\_NOWRITE**

Unwritable file closed.

**IN\_CLOSE**

Equivalent to `ladang.IN_CLOSE_WRITE | ladang.IN_CLOSE_NOWRITE`

**IN\_OPEN**

File was opened.

**IN\_MOVED\_FROM**

File was moved from X.

**IN\_MOVED\_TO**

File was moved to Y.

**IN\_MOVE**

Equivalent to `ladang.IN_MOVED_FROM | ladang.IN_MOVED_TO`

**IN\_CREATE**

Subfile was created.

**IN\_DELETE**

Subfile was deleted.

**IN\_DELETE\_SELF**

Self was deleted.

**IN\_MOVE\_SELF**

Self was moved.

**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\_ADD**

Add to the mask of an already existing watch.

**IN\_ISDIR**

Event occurred against dir.

**IN\_ONESHOT**

Only send event once.

**IN\_ALL\_EVENTS**

All events which can be waited on.

Equivalent to:

ladang.IN\_ACCESS | ladang.IN\_MODIFY | ladang.IN\_ATTRIB | ladang.IN\_CLOSE\_WRITE |  
ladang.IN\_CLOSE\_NOWRITE | ladang.IN\_OPEN | ladang.IN\_MOVED\_FROM | ladang.IN\_MOVED\_TO |  
ladang.IN\_CREATE | ladang.IN\_DELETE | ladang.IN\_DELETE\_SELF | ladang.IN\_MOVE\_SELF

# INDICES AND TABLES

- *genindex*
- *modindex*
- *search*