Iadang Documentation

Release 0.9.0

E A Faisal <eafaisal at gmail dot com>

January 15, 2013

CONTENTS

Contents:

CHAPTER

ONE

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
```

CHAPTER

TWO

LICENSE

ladang is licensed under the MIT license.

2.1 The MIT License

CHAPTER

THREE

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 occured 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)
```

CHAPTER

FOUR

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

CHAPTER

FIVE

_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
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

CHAPTER

SIX

INDICES AND TABLES

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