Event for file change #1

Edit New issue

Open eonist opened this issue 5 days ago · 68 comments

eonist commented 5 days ago

Can this be extended to support monitoring for file change events?

gurinderhans commented 5 days ago Owner

Hi **@eonist** , I understand you want to be able to do something like

```
let w = SwiftFSWatcher.createWatcher()
w.paths = ["/some/path/to/myfile.txt"]
w.watch()
```

and only want to receive events when `myfile.txt` got changed?

This is however an API limit, thus you are only allowed to watch and receive events for folders.

eonist commented 5 days ago

Thx for your reply. I went ahead and made a similar FileMonitor class in swift:
https://github.com/eonist/swift-utils/blob/master/file/FileWatcher.swift

Its not complete yet. I will add a few things to it. A way to monitor for file changes in files among them. I guess I will filter out the events that pertains to a specific file.

By the way:

Until Mac OS X 10.7, FSEvents did not "watch" the filesystem, such as Linux's inotify: the API provided no notifications for changes to individual files. An application was able to register to receive changes to a given directory, and had to determine for itself which file or files were changed.

Mac OS X 10.7 (Lion) added the ability to register for file modification notifications.

eonist commented 5 days ago

You can filter the events down to the specific file by the eventPath argument. Just supply the logic needed. Like watch(dirURL,fileName) and then return some event when an event happens on the file specified. I don't know if that made any sense to anyone els. But at leasts its possible.

gurinderhans commented 5 days ago Owner

@eonist Hey just ran that code. But there's a slight issue with the flags.

For ex. I did

```
let ss = FileWatcher(["/Users/gurinderhans/Desktop/test.txt"])
ss.start()
```

and the output I am getting is

```
eventCallback()
     eventId: 14245568 - eventFlags:  70656 - eventPath:  /Users/gurinderhans/Desktop/test.
```

```
unsupported event: 70656
```

The `eventFlags` doesn't seem to be matching the `Flags.dataChange`, which according to you is `128000` but this shows `70656`.

File detection works tho, which is awesome! Mind if I borrow it for this module?

eonist commented 5 days ago

Mi casa su casa ;)

So these eventFlags are a bit hard to interpret to be honest, i'm Still trying to decipher them. So what did you do to get the 70656 flag?

When I edit a text.txt File i get the 128000 flag.

There are 2 changes. The one I call dataChange:128000 and change:67584 The former is when you edit the content of a file. and the later is when you edit the file. Add, rename etc.

Mind you i'm on OSX El cap. Might be relevant. Im not sure.

The file detection is awesome!!! A lot of cool apps could be made with this. Hazel comes to mind: https://www.noodlesoft.com/hazel.php

gurinderhans commented 5 days ago Owner

@eonist Yes editing `test.txt` file got me `70656`. I also get the same flag when for example editing `test.md` with `.md` extension. When did you get 128000 flag? I would assume `70656` is the `dataChange` flag.

Also if you tell it to watch over a file with no extension, ex. `mytestfile`, changing the contents of file causes your code to crash at the end of the `eventCallback` method.

OS Version: 10.11.4 (El Capitan)
XCode Version : 7.3 (7D15)

eonist commented 5 days ago

```
/**
 * Helper class to differentiate between the FSEvent flag types (aka file event types)
 */
private class Flags{
    static var dataChange:UInt32 = 128000//data in the file changed
    static var change:UInt32 = 67584//add,rename, move?
    static var delete:UInt32 = 111872//the file was deleted
}
```

eonist commented 5 days ago

If you want to listen to a specific file you have to add support for that your self. What im proposing is that its not that hard to accomplish as you can derive the file path on each event. If you want to listen to a file with no extension. Well thats another monster entirely. What if you had image.jpg and image.gif for instance? Extension is a must i think in any logical sense of the mater.

eonist commented 5 days ago

What i need this class to do is monitor a folder i specify, and then have it send me events. Then ill filter these events depending on their use case. I want to keep the FileWatcher as simple as possible as its already a bit complex for something so simple.

gurinderhans commented 5 days ago Owner

Yea it seems different file types have different eventFlag numbers. And for sure! Thanks for demonstrating the method to support listening for single file changes, I will make sure to add that to this module.

eonist commented 5 days ago

Further research is needed then. Im going through some sites as we speak to try and solve this. Ill try .md. I was testing a .css

eonist commented 5 days ago

Actually. Im only getting one flag for different files as well: dataChange:UInt32 = 128000

tried .md, .css and .txt (I just edit some text inside them and then hit save)

Are you on the latest OSX? Anything different with your system? I havent updated to the latest xcode or swift. Im in XCode 7.2 (I cant update it since a lot of my work needs to not give me problems right now)

gurinderhans commented 5 days ago Owner

`.css` yeilds `70912` .

And yes!
OS Version: 10.11.4 (El Capitan), latest
XCode Version : 7.3 (7D15), latest

eonist commented 5 days ago

Seems inconsistent. Im on 10.11.3 (holding off the update)

My gut feeling says its the OSX version. Since i read somewhere that Apple is depricating things related to FSEvent. And making changes to it. Also the C call back stuff in swift isnt really that clean unsafeBitCast etc.

Ill press on to figure it out.

eonist commented 5 days ago

Here is description of what im making with this file-change-detection-utility:
http://stylekit.org/blog/2016/04/07/Live-edit/

gurinderhans commented 5 days ago Owner

Ahh, native UI design with CSS? That sounds cool!

eonist commented 5 days ago

Yepp. And not some faux JavaScript Native. This is all Native swift, and there is no reliance on any apple code other than swift it self, so no auto-layout etc. Adding Live edit to it this weekend. Think prototyping your UI design with-out recompiling your app. Ill also try to port the framework to IOS this summer so we can have one unified way of making apps for both IOS and OSX.

eonist commented 5 days ago

So if you edit: a css file named Button.css with: Button#someButton{fill:red;} to fill:blue. Then you would immediately see your app change its button color.

gurinderhans commented 5 days ago Owner

Yes makes sense, I'll definitely be sure to keep an eye on it.
Is it open source?

```
1
```

eonist commented 5 days ago

Yes, OpenSource. MIT.

Im planing to add a way to ad-hock your eventHandler to the FileWatcher instance like:

```
let fileWatcher = FileWatcher(["~/Desktop/test".tildePath])
fileWatcher.start()
func onEvent(event:Event){
    //handle switching between file event flags here and filter filePath etc
}
fileWatcher.event = onEvent
```

This way you could specify your own flags that work for you as well. Also makes the class more modular to different use cases.

What do you think of such a scheme?

gurinderhans commented 5 days ago Owner

I see nothing wrong. Should work fine!
…

eonist commented 5 days ago

Ill try it tomorrow and see how it goes. You could also pass in a method with the code you wanted to execute but having the instance call you rather than it calling something you provide it seems the better choice. I will base it on this Event system: http://stylekit.org/blog/2016/02/10/The-event-system/

eonist commented 5 days ago

By the way: The different eventFlags we are getting. Could be because its plural. Aka flags. So apple combines UInts together to form a sort of array. But we read it as one value. I never really understand how apple handles the oddities of plural UInt values. like flag1|flag2|flag3 and then thats one value and a list of values.

eonist commented 5 days ago

Here is the overview of FSEventFlags:
https://developer.apple.com/library/mac/documentation/Darwin/Reference/FSEvents_Ref/#//apple_ref/doc/constant_group/FSEventStreamEventFlags

gurinderhans commented 4 days ago Owner

Interesting about the plural thing. It's sounds very probable. I will have a look at it too once I get the chance.

eonist commented 4 days ago

Im getting the 70656 flag when the .DS_Store is changed. I can't figure out the FSEventFlags yet. But i think it has something to do that its an OR value. Pressing on to figure it out.

eonist commented 4 days ago

These are the flags according to apple:

```
enum {
      kFSEventStreamEventFlagNone = 0x00000000,
      kFSEventStreamEventFlagMustScanSubDirs = 0x00000001,
      kFSEventStreamEventFlagUserDropped = 0x00000002,
      kFSEventStreamEventFlagKernelDropped = 0x00000004,
      kFSEventStreamEventFlagEventIdsWrapped = 0x00000008,
      kFSEventStreamEventFlagHistoryDone = 0x00000010,
      kFSEventStreamEventFlagRootChanged = 0x00000020,
      kFSEventStreamEventFlagMount = 0x00000040,
      kFSEventStreamEventFlagUnmount = 0x00000080 , /* These flags are only set if you specif
      /* flags when creating the stream.*/
      kFSEventStreamEventFlagItemCreated = 0x00000100,
      kFSEventStreamEventFlagItemRemoved = 0x00000200,
      kFSEventStreamEventFlagItemInodeMetaMod = 0x00000400,
      kFSEventStreamEventFlagItemRenamed = 0x00000800,
      kFSEventStreamEventFlagItemModified = 0x00001000,
      kFSEventStreamEventFlagItemFinderInfoMod = 0x00002000,
      kFSEventStreamEventFlagItemChangeOwner = 0x00004000,
      kFSEventStreamEventFlagItemXattrMod = 0x00008000,
      kFSEventStreamEventFlagItemIsFile = 0x00010000,
      kFSEventStreamEventFlagItemIsDir = 0x00020000,
      kFSEventStreamEventFlagItemIsSymlink = 0x00040000
  };
```

Source:
https://developer.apple.com/library/mac/documentation/Darwin/Reference/FSEvents_Ref/#//apple_ref/c/tdef/FSEventStreamEventFlags

eonist commented 4 days ago

You can also watch single files with FileWatcher:

```
let fileWatcher = FileWatcher(["~/Desktop/test/text.txt".tildePath])
fileWatcher.start()
```

I didn't know it worked, I think you tried it maybe?

eonist commented 4 days ago

I see you use:

```
globalSelf
```

Im trying to make this instance based. So that one could have multiple listeners in different places. Seems other people have trouble with this aspect as well.

eonist commented 4 days ago

Found a solution to making it instance based: https://github.com/soh335/FileWatch

Ill try to make something similar to my FileWatcher class.

gurinderhans commented 4 days ago Owner

Ah, I didn't really require multiple instances and what I needed was provided by this, so didn't give it much thought. My plan is now to just rewrite this into separate objc and swift versions, in which I will also add watching single files support.

eonist commented 4 days ago

Sounds good. I found a way to do multi instance FSEvents via NotificationCenter and some trickery. But its not clean. Seemingly the proper way to do it is to extend NSThread. like this repo:
https://github.com/ooper-shlab/CocoaSlideCollection-Swift/blob/f14e89865406c650627df95a864e399074d46f09/CocoaSlideCollection/Model/AAPLFileTreeWatcherThread.swift

I just think watching only one path is kind of limiting. Also you will need to use NotificationCenter or

NSThread if you want the FSEvents to actually do something to your app. Like change a UI component etc. Since you cant really reach back onto the main thread with the current code.

eonist commented 4 days ago

Seems like you figured out how make multiple instance of the watcher. From reading your readme file. Maybe because you use a private init. Going to look into that tomorrow. How you did it. I think its a much better approach than NSNotification or NSThread. Just make sure you can reach class scoped variables in your onFileChange method. If you can, then that should be the best watcher lib i've come across.

eonist commented 4 days ago

Also be careful when you debug FSEvents. They can sometimes get stuck from the last run. Which can throw you off big time. Sometimes its best to change the path you were debugging to another path to get things working again. Threw me off a lot today.

gurinderhans commented 4 days ago Owner

Oh haha, and I do have the stop stream method to close the stream. Is that what you mean? Or just through some bug the stream gets stuck? In that case there's a dealloc method. I also should write tests for this, helps keep code clean. And do you mean `onFileChange` or `onFileChange d` ? I should prolly change the private inner class method name to avoid ambiguity.

gurinderhans commented 3 days ago Owner

How are you planning on doing

```
let fileWatcher = FileWatcher(["~/Desktop/test".tildePath])
fileWatcher.start()
func onEvent(event:Event){
    //handle switching between file event flags here and filter filePath etc
}
fileWatcher.event = onEvent
```

Inside the callback you use `unsafeBitCast` to get instance of the `FileWatcher` but it's not the exact copy returned. So the `fileWatcher.event = onEvent` has no effect since inside `handleEvent` you won't be able to access `event` variable.

eonist commented 3 days ago

Im just not able to deinit in swift. It just deinits right after you init. So i commented the deinit away. This is probably what causes the ghost debugging problems.

I moved away from the event scheme you described above. I use NSNotification now. But i dont like it. Im going to press on today and figure it out.

gurinderhans commented 3 days ago Owner

Interesting... what are you doing with NSNotification? Are you using it in place of a callback? That seems a bit overkill?

eonist commented 3 days ago

I am. Then I pass on the context in the userInfo in the Notification to differentiate the different notifications. Hey, at least it works. The only drawback is that every listener has to check if its the right context. Which is not good code conduct in my book. Have enough of these and performance could slow down. you can check out my current build here: https://github.com/eonist/swift-utils/blob/master/file/FileWatcher.swift

This is my Notification system:

```
NSNotificationCenter.defaultCenter().addObserver(self, selector: "someObserver:", name: "Sc

func someObserver(notification: NSNotification) {//remember to place this in a class scope
        //Swift.print("someObserver " + "\(notification.userInfo!["data"]!)")

        //Swift.print("\(fileWatcher!.contextInfoCopy!)")


        if((notification.userInfo!["data"]! as! String) == "\(fileWatcher!.contextInfoCopy!
            Swift.print("correct fileWatcher")
        }
    }
```

Then you add this to the callBack closure in the FileWatcher class:

```
NSNotificationCenter.defaultCenter().postNotificationName("SomeNotification", object:nil,us
```

This works. But its my backup solution. If I cant find a better way at least I have a way. But like you said its OverKill deluxe.

eonist commented 3 days ago

A question about your code:

```
watcher.onFileChange = {numEvents, changedPaths in
      println("recieved: \(numEvents) events")
      println("changedPaths: \(changedPaths)")
  }
```

are you able to reach class scoped variables inside that closure? like if I did:

```
let someVar:String = "testing"
watcher.onFileChange = {numEvents, changedPaths in
      println("recieved: \(numEvents) events")
      println("changedPaths: \(changedPaths)")
      print(someVar)//<----would this be able to print?
  }
```

The reason i'm using Notifications is that its an easy way to pass information from the thread that the FileWatcher is on and on to the main thread that the rest of your app is on.

eonist commented 3 days ago

Still awake in Canada? I solved it. So easy!

```
var temp:String = "123"
fileWatcher!.onEvent = { [unowned self] eventId, eventPath, eventFlags in
            Swift.print("onFileChange() " + "\(self.temp)")//<---this prints out 123
        }
```

So its able to work on the main thread by including that [unowned self] variable

gurinderhans commented 3 days ago Owner

I can't access class level variables inside the closure. And wow looks interesting! Is it with the `[unowned self]` ? How come this works?

eonist commented 3 days ago

If you cant reach class level variables a FileWatcher is pretty useless. No offence, this stuff is pretty undocumented. Other people solve this with Grand Central Dispatch. Ive solved a similar case with performSelectorOnMainThread in my Animation Kit to get 60fps frame animation working in OSX. But this approach didnt work with the FSEvent unfortunately.

by including the [unowned self] you create a ref inside the closure. [weak self] is probably more appropriate.

The general rule is:

1. If self could be nil in the closure use [weak self].
2. If self will never be nil in the closure use [unowned self].

By using [weak self] the original ref can be removed and the FileWatcher would still work.

Here is a bit more info: http://blog.xebia.com/swift-self-reference-in-inner-closure/ It mentions the a_sync aspect. But its not a complete explanation why this works.

gurinderhans commented 3 days ago Owner

Wait wait wait.... I take that back. Yes you can access `class` level variables. I assume you mean something like....

```
class MyClass: NSObject {
    var memberVar: Int = 3
    func someFunc() {
        var temp:String = "123"
        fileWatcher!.onEvent = { [unowned self] eventId, eventPath, eventFlags in
            print("onFileChange() " + "\(self.temp)")//<---this prints out 123
            print("memberVar: \(memberVar)")
            // both would work
        }
    }
}
```

Stupid of me, but previously I thought, when you referenced to class that you were saying access class level variables from `SwiftFSWatcher.class` , which is ridiculously hilarious.

1

eonist commented 3 days ago

Right. Both works.

Class scoped variables. I guess Instance scoped variables would be a better name?, feel free to suggest a better phrasing here.... Static class variables, and class type variables is another monster. Which is not related here :D

....

Also figured out our Flag problems. Just drop this in your evenhandler:

```
if (eventFlags & FSEventStreamEventFlags(kFSEventStreamEventFlagItemModified)) != 0 {
                Swift.print("File modified: \(eventPath) - \(eventId)")
            }
```

the other cases is on my blog: http://stylekit.org/blog/2016/04/07/Live-edit/ or from apples own FSEvent ref docs

FUN-FACT: Dropbox also uses FSEvents to watch the change inside the Dropbox folder.

I think thats it. Should be able to make awesome FileWatch'er kits now.

1

eonist commented 3 days ago

The only flag that doesn't work is the "file removed" flag. Oh well, probably eligible for a apple bug report.

Maybe it works for you?

eonist commented 3 days ago

This is someone doing FileWatching with NSThread: https://github.com/ooper-shlab/CocoaSlideCollection-Swift/blob/f14e89865406c650627df95a864e399074d46f09/CocoaSlideCollection/Model/AAPLFileTreeWatcherThread.swift

These FileWatcher kits use GDC:
https://github.com/Eonil/FileSystemEvents &
https://github.com/nvzqz/FileKit/blob/develop/FileKit/Core/FileSystemWatcher.swift

Pretty complicated stuff. So I think the [unowned self] approach is a much simpler way to do it.

gurinderhans commented 3 days ago Owner

I still don't understand this line https://github.com/eonist/swift-utils/blob/master/file/FileWatcher.swift#L63
I just tested and it doesn't work since `event` is `nil`. Then what you said about `[unowned self]` doesn't work out either.

eonist commented 3 days ago

event is nil yes. you have to set it

before you call fileWatcher.start()
Like:

```
fileWatcher = FileWatcher(["~/Desktop/test/".tildePath],FSEventStreamEventId(kFSEventStrean

        fileWatcher!.event = { [weak self] eventId, eventPath, eventFlags in

            print(self?.temp)
            Swift.print("\t eventId: \(eventId) - eventFlags:  \(eventFlags) - eventPath:
}

        fileWatcher!.start()
```

eonist commented 3 days ago

Here is someone doing FileWatching with delegation: Although they use: **@objc** public func. Which is a no-no in my book. Isn't that obj-c bridging?

https://github.com/seorenn/SRPath/blob/a8cdcae445a3bab13ed9355f0fd379e47cc3ccc6/SRPath/SRPath/Sources/SRPathMonitor.swift

Here is a way you could make methods instead of closures with weak self. But I think it looks to complicated and isn't worth it : http://blog.xebia.com/function-references-in-swift-and-retain-cycles/ I rather use a closure. Although i try to always favour methods over closures.

eonist commented 3 days ago

Im happy with my FileWatcher at this point: Also made a cool FileWatcherEvent to simplify the event handling:

https://github.com/eonist/swift-utils/blob/master/file/filewatcher/FileWatcherEvent.swift

and here is the final FileWatcher class:

https://github.com/eonist/swift-utils/blob/master/file/FileWatcher.swift

And here is the final example code:

https://github.com/eonist/swift-utils#filewatcher

It does what I need it to do: Alert me if there is a file change. However it does not handle removing a file and adding a file, then it alerts you that a file has been renamed. As mentioned I think this is an Apple bug. Ill try to compile on another computer as it may be the xcode version i'm working on that has a bug etc.

Would be awesome if this worked for you. As I can recall we did get different flags a while back there.

eonist commented 3 days ago

FileWatcher at work: https://vimeo.com/162258482 Live Editing now works. Will make the process of making apps so much faster!

gurinderhans commented 3 days ago Owner

OH man that demo looks awesome!

But I still can't get your class to work. It crashes at line 66 [https://github.com/eonist/swift-utils/blob/master/file/FileWatcher.swift#L66], becasue `fileSystemWatcher` is `nil`.

eonist commented 3 days ago

```
var fileWatcher = FileWatcher(["~/Desktop/test/".tildePath])/*<---the fileWatcher instance

fileWatcher!.event = { [weak self] event in//<--The weak self part enables you to interact
    Swift.print(self?.someVariable)//Outputs: a variable in your current class
    Swift.print(event.description)//Outputs: a description of the file change
}
```

Notice the comment in the first line there?

eonist commented 3 days ago

Trending on github today. https://github.com/trending/swift :D

1 1

gurinderhans commented 3 days ago Owner

Ahh that worked! Although it's a limitation we shouldn't have.

eonist commented 3 days ago

What do you mean?

gurinderhans commented 3 days ago Owner

Like the variables should be able to be locally scoped and it still receives events.

eonist commented 3 days ago

The thing is that when your dealing with multiple threads. The system has to have a way to reference things. Variables inside methods are unreachable for the system. So you have to scope it to the class. Same thing with my Animation kit. If I scope it to a method it doesnt work.

1

eonist commented 3 days ago

Just the way it is Im afraid. I think it is related to refrence counting, ARC etc, stumbled on similar cases throught he building of my UI framework.

eonist commented 3 days ago

Scoping things to the class scope isnt auch a bad thing though. if you want to gracefully close a class for isntance its noce to be able to reach filewatcher. I agree that it would be best to be optional. I dont like that closure either but doing it with a method was way more complex.

eonist commented 3 days ago

Got to go and sleep. Its late in Norway. Good luck with your kit. Grab any code you want. And i hope you solve the file event flags. The only flag that worked for me was file modification. That was the only one i needed though.

1

gurinderhans commented 2 days ago Owner

I sorta came up with this

```
class FileEvents {

    /// MARK: - file events
    static var fileCreated = (kFSEventStreamEventFlagItemIsFile + kFSEventStreamEventFlagIt
    static var fileRenamed = (kFSEventStreamEventFlagItemIsFile + kFSEventStreamEventFlagIt
    static var fileDeleted = (kFSEventStreamEventFlagItemIsFile + kFSEventStreamEventFlagIt
    static var fileModified = (kFSEventStreamEventFlagItemIsFile + kFSEventStreamEventFlagI

    /// MARK: - folder events
    static var folderCreated = (kFSEventStreamEventFlagItemIsDir + kFSEventStreamEventFlagI
    static var folderRenamed = (kFSEventStreamEventFlagItemIsDir + kFSEventStreamEventFlagI
    static var folderDeleted = (kFSEventStreamEventFlagItemIsDir + kFSEventStreamEventFlagI
}
```

It's not perfect since, `fileDeleted` is sometimes `kFSEventStreamEventFlagItemIsFile + kFSEventStreamEventFlagItemRemoved+kFSEventStreamEventFlagItemCreated`, which makes no sense to me.

eonist commented 2 days ago

Yeh, the concept of renaming doesnt exist to a computer i guess. It rather deletes it and recreates it in a new name.

So your way of doing flags may be the best way actually.

1

eonist commented 2 days ago

good job on your filewatcher, how did you solve the weak self closure?

gurinderhans commented a day ago Owner

I didn't solve it. It just works without!

1

eonist commented 3 hours ago

That sounds strange. Never seen a FileWatcher kit that doesn't somehow have to deal with the concurrency of threads. As the FSEvent machine definitely doesn't run on the main thread. Im going to build a tiny app soon that utilises the FSEvent machine. I want to make an app that clones your file-structure from one folder to the other. Maybe ill know more about this threading later.

Ive been digging around on stackoverflow for the Event type inconsistencies we had to deal with and it seems that its something that plagues others as well. There are ways to mitigate this. Like asserting the file index before and after an event. Or making a hybrid system with KQueues (kernelQueues) which apparently also has its inconsistencies.

Labels

None yet

Milestone

No milestone

Assignee

No one assigned

Notifications

Unsubs

You're receiving notifi
because you were me

2 participants

eonist commented just now

Actually. Maybe just moving the FileWatcher instance from the scope of a method to the scope of the class instance could have solved why you don't need the "weak self". Ive experienced something similar with my Animation kit before. I tried to remove the weak self part and it also works for me.

The reason behind this is totally speculating but as Ive mentioned before I think it has something to do with being reachable by the app it self. A variable inside a method is sort of isolated from the app as opposed to a class scoped variable which would be reachable by the app.

One thing to think about is that it may need to be reachable all the way back to the appDelegate class. So if you isolate the class instance that holds your fileWatcher instance it may stop to work. Im speculating here.

Just something to keep in mind if you get an error down the line and cant figure out why.

Write Preview

Leave a comment

Attach files by dragging & dropping, selecting them, or pasting from the clipboard.

Styling with Markdown is supported

Close issue Comment