File system notifications for Go

GO reference go report A+

fsnotify utilizes golang.org/x/sys rather than `syscall` from the standard library. Ensure you have the latest version installed by running:

```
go get -u golang.org/x/sys/...
```

Cross platform: Windows, Linux, BSD and macOS.

Adapter	OS	Status
inotify	Linux 2.6.27 or later, Android*	build passing Supported
kqueue	BSD, macOS, iOS*	build passing Supported
ReadDirectoryChangesW	Windows	build passing Supported
FSEvents	macOS	Planned
FEN	Solaris 11	In Progress
fanotify	Linux 2.6.37+	Planned
USN Journals	Windows	Maybe
Polling	*All*	Maybe

* Android and iOS are untested.

Please see the documentation and consult the FAQ for usage information.

API stability

fsnotify is a fork of howeyc/fsnotify with a new API as of v1.0. The API is based on this design document.

All releases are tagged based on Semantic Versioning. Further API changes are planned, and will be tagged with a new major revision number.

Go 1.6 supports dependencies located in the `vendor/` folder. Unless you are creating a library, it is recommended that you copy fsnotify into `vendor/github.com/fsnotify/fsnotify` within your project, and likewise for `golang.org/x/sys`.

Usage

```go
package main

import (
    "log"

    "github.com/fsnotify/fsnotify"
)

func main() {
    watcher, err := fsnotify.NewWatcher()
    if err != nil {
        log.Fatal(err)
    }
    defer watcher.Close()

    done := make(chan bool)
    go func() {
        for {
            select {
            case event, ok := <-watcher.Events:
                if !ok {
                    return
                }
                log.Println("event:", event)
                if event.Op&fsnotify.Write == fsnotify.Write {
                    log.Println("modified file:", event.Name)
                }
            case err, ok := <-watcher.Errors:
                if !ok {
                    return
                }
                log.Println("error:", err)
            }
        }
    }()

    err = watcher.Add("/tmp/foo")
    if err != nil {
        log.Fatal(err)
    }
    <-done
}
```

Contributing

Please refer to CONTRIBUTING before opening an issue or pull request.

Example

See example_test.go.

FAQ

When a file is moved to another directory is it still being watched?

No (it shouldn't be, unless you are watching where it was moved to).

When I watch a directory, are all subdirectories watched as well?

No, you must add watches for any directory you want to watch (a recursive watcher is on the roadmap #18).

Do I have to watch the Error and Event channels in a separate goroutine?

As of now, yes. Looking into making this single-thread friendly (see howeyc #7)

Why am I receiving multiple events for the same file on OS X?

Spotlight indexing on OS X can result in multiple events (see howeyc #62). A temporary workaround is to add your folder(s) to the *Spotlight Privacy settings* until we have a native FSEvents implementation (see #11).

How many files can be watched at once?

There are OS-specific limits as to how many watches can be created:

- Linux: /proc/sys/fs/inotify/max_user_watches contains the limit, reaching this limit results in a "no space left on device" error.
- BSD / OSX: sysctl variables "kern.maxfiles" and "kern.maxfilesperproc", reaching these limits results in a "too many open files" error.

Why don't notifications work with NFS filesystems or filesystem in userspace (FUSE)?

fsnotify requires support from underlying OS to work. The current NFS protocol does not provide network level support for file notifications.

Related Projects

- notify
- fsevents