

build passing

# go-gitignore

A fast gitignore matching library for Go.

This library use simple tree index for matching, so keep fast if gitignore file has many pattern.

## Usage

```
gitignore, _ := gitignore.NewGitIgnore("/path/to/gitignore")

path := "/path/to/file"
isDir := false
gitignore.Match(path, isDir)
```

### Specify base directory

go-gitignore treat `path` as a base directory. If you want to specify other base (e.g. current directory and Global gitignore), you can like the following.

```
gitignore, _ := gitignore.NewGitIgnore("/home/you/.gitignore", ".")
```

### From io.Reader

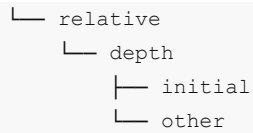
go-gitignore can initialize from io.Reader.

```
gitignore, _ := gitignore.NewGitIgnoreFromReader(base, reader)
```

## Simple tree index

go-gitignore parse gitignore file, and generate a simple tree index for matching like the following.

```
.
├── accept
│   ├── absolute
│   │   └── depth
│   │       ├── initial
│   │       └── other
│   └── relative
│       ├── depth
│       ├── initial
│       └── other
└── ignore
    ├── absolute
    │   └── depth
    │       ├── initial
    │       └── other
```



## Features

- Support absolute path (/path/to/ignore)
- Support relative path (path/to/ignore)
- Support accept pattern (!path/to/accept)
- Support directory pattern (path/to/directory/)
- Support glob pattern (path/to/\*.txt)

*note: glob pattern*

go-gitignore use [filepath.Match](#) for matching meta char pattern, so not support recursive pattern (path/ \*\* /file).

## Installation

```
$ go get github.com/monochromegane/go-gitignore
```

## Contribution

1. Fork it
2. Create a feature branch
3. Commit your changes
4. Rebase your local changes against the master branch
5. Run test suite with the `go test ./...` command and confirm that it passes
6. Run `gofmt -s`
7. Create new Pull Request

## License

[MIT](#)

## Author

[monochromegane](#)