

testing を眺める

2025-08-30 Zli OB/OG LT



matumoto

自己紹介

- 学籍番号：s1280136
- 所属：元会津大学コンピュータ理工学部
- 今興味のあること：
- 趣味：
 - VTuber
 - スト6を見る
- ひとこと：
 - 久々にスライド作成でひりついた

Goのtestingパッケージ
を見ていきます

testing

Testing

packagestandard library

Version: go1.25.0Latest | Published: Aug 12, 2025 | License: BSD-3-Clause | Imports: 25 | Imported by: 103,324

TestMain is a low-level primitive and should not be necessary for casual testing needs, where ordinary test functions suffice.

Jump to ...

Documentation

Overview

Index

Examples

Constants

Variables

Functions

Types

Source Files

Directories

Index

func AllocsPerRun(runs int, f func()) (avg float64)

func CoverMode() string

func Coverage() float64

func Init()

func Main(matchString func(pat, str string) (bool, error), tests []InternalTest, ...)

func RegisterCover(c Cover)

func RunBenchmarks(matchString func(pat, str string) (bool, error), ...)

func RunExamples(matchString func(pat, str string) (bool, error), examples []InternalExample) (ok bool)

func RunTests(matchString func(pat, str string) (bool, error), tests []InternalTest) (ok bool)

func Short() bool

func Testing() bool

func Verbose() bool

type B

func (c *B) Attr(key, value string)

func (c *B) Chdir(dir string)

func (c *B) Cleanup(f func())

func (c *B) Context() context.Context

func (b *B) Elapsed() time.Duration

func (c *B) Error(args ...any)

func (c *B) Errorf(format string, args ...any)

func (c *B) Fail()

func (c *B) FailNow()

func (c *B) Failed() bool

func (c *B) Fatal(args ...any)

func (c *B) Fatalf(format string, args ...any)

func (c *B) Helper()

func (c *B) Log(args ...any)

func (c *B) Logf(format string, args ...any)

func (b *B) Loop() bool

func (c *B) Name() string

func (c *B) Output() io.Writer

func (b *B) ReportAllocs()

func (b *B) ReportMetric(n float64, unit string)

testing パッケージ

- testing.T
 - testing.B
 - testing.TB
 - testing.PB
 - testing.M
 - testing.F
-
- いつもお世話になっております

testing.T: テスト

```
run test | debug test
func TestAdd(t *testing.T) {
    run test | debug test
    t.Run("1 + 2 = 3", func(t *testing.T) {
        if Add(1, 2) != 3 {
            t.Error("NO!!!")
        }
    })
}
```

testing.B: ベンチマーク

```
run benchmark | debug benchmark
func BenchmarkAdd(b *testing.B) {
    for range b.N {
        Add(1, 2)
    }
}
```

testing.B: ベンチマーク

```
● ⇒ go test -bench .
```

```
goos: linux
```

```
goarch: amd64
```

```
pkg: gp
```

```
cpu: Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz
```

```
BenchmarkAdd-8          10000000000          0.2538 ns/op
```

```
PASS
```

```
ok      gp      0.289s
```

testing.TB: バンジーガム



testing.PB: パラレルベンチマーク

```
run benchmark | debug benchmark
func BenchmarkAddParallel(b *testing.B) {
    b.RunParallel(func(p *testing.PB) {
        for p.Next() {
            _ = Add(1, 2)
        }
    })
}
```

testing.M: テストのメイン関数

```
func TestMain(m *testing.M) {  
    fmt.Println("setup")  
  
    code := m.Run()  
  
    fmt.Println("teardown")  
  
    os.Exit(code)  
}
```

testing.F: ファジーテスト

```
run test | debug test
func FuzzAdd(f *testing.F) {
    f.Fuzz(func (t *testing.T, a, b int) {
        if Add(a, b) != a + b {
            t.Error("NO!!!!!!")
        }
    })
}
```

testing.Short()

- go test -short 指定の際に分岐できる

```
run test | debug test
func TestAdd(t *testing.T) {
    if testing.Short() {
        t.Skip("skip!! in -short flag mode")
    }
    // ...
}
```

testing/ 配下

▼ syscall	Package syscall contains an interface to the low-level operating system primitives.
▼ testing	Package testing provides support for automated testing of Go packages.
fstest	Package fstest implements support for testing implementations and users of file systems.
iotest	Package iotest implements Readers and Writers useful mainly for testing.
quick	Package quick implements utility functions to help with black box testing.
slogtest	Package slogtest implements support for testing implementations of log/slog.Handler.
synctest	Package synctest provides support for testing concurrent code.
► text	

testing/ 配下

syscall	Package syscall contains an interface to the low-level operating system primitives.
testing	Package testing provides support for automated testing of Go packages.
<u>fstest</u>	Package fstest provides support for testing implementations and users of file systems.
iotest	Package iotest implements Readers and Writers useful mainly for testing.
quick	Package quick implements utility functions to help with black box testing.
slogtest	Package slogtest implements support for testing implementations of log/slog.Handler.
synctest	Package synctest provides support for testing concurrent code.
text	

今回は testing/fstest を紹介

testing/fstest

こんな関数を考えてみる

```
func joinFiles(name1, name2 string) error {
    body1, _ := os.ReadFile(name1)

    body2, _ := os.ReadFile(name2)

    bodies := [][]byte{body1, body2}

    const separator = "\n"

    combinedFile := bytes.Join(bodies, []byte(separator))

    _ = os.WriteFile("joined.txt", combinedFile, 0755)

    return nil
}

func main() {
    joinFiles("a.txt", "b.txt")
}
```

```
func joinFiles(name1, name2 string) error {
    body1, _ := os.ReadFile(name1)

    body2, _ := os.ReadFile(name2)

    bodies := [][]byte{body1, body2}

    const separator = "\n"

    combinedFile := bytes.Join(bodies, []byte(separator))

    _ = os.WriteFile("joined.txt", combinedFile, 0755)

    return nil
}

func main() {
    joinFiles("a.txt", "b.txt")
}
```

ファイルシステムをいじる関数のテストは大変...

- そもそもロジックとI/Oを分ければよいのでは？
 - => いけそう！

```
func joinFiles(name1, name2 string) error {
    body1, _ := os.ReadFile(name1)

    body2, _ := os.ReadFile(name2)

    bodies := [][]byte{body1, body2}

    const separator = "\n"

    combinedFile := bytes.Join(bodies, []byte(separator))

    _ = os.WriteFile("joined.txt", combinedFile, 0755)

    return nil
}

func main() {
    joinFiles("a.txt", "b.txt")
}
```

```
func joinFiles(name1, name2 string) error {
```

```
    body1, _ := os.ReadFile(name1)
```

```
    body2, _ := os.ReadFile(name2)
```

```
    bodies := [][]byte{body1, body2}
```

```
    const separator = "\n"
```

```
    combinedFile := bytes.Join(bodies, []byte(separator))
```

```
    _ = os.WriteFile("joined.txt", combinedFile, 0755)
```

```
    return nil
```

```
}
```

```
func main() {
```

```
    joinFiles("a.txt", "b.txt")
```

```
}
```

Input

ロジック

Output

```
func joinFiles(name1, name2 string) error {
```

```
    body1, _ := os.ReadFile(name1)
```

```
    body2, _ := os.ReadFile(name2)
```

```
    combinedFile := joinFileBody(body1, body2)
```

```
    _ = os.WriteFile("joined.txt", combinedFile, 0755)
```

```
    return nil
```

```
}
```

Input

Output

```
func joinFileBody(body1, body2 []byte) []byte {
```

```
    bodies := [][]byte{body1, body2}
```

```
    const separator = "\n"
```

```
    return bytes.Join(bodies, []byte(separator))
```

```
}
```

ロジック

これでロジックのテストは書きやすい！

- ロジック部分は純粹関数なので単体テスト書くだけ！
- じゃあ、Inputのときにファイル読み取れるかのテストって...？

これでロジックのテストは書きやすい！

- ロジック部分は純粋関数なので単体テスト書くだけ！
- じゃあ、Inputのときにファイル読み取れるかのテストって...?



ファイルシステムをテストしやすくしよう！

io/fs パッケージの fs.FS インターフェース

type FS

```
type FS interface {  
    // Open opens the named file.  
    // [File.Close] must be called to release any associated resources.  
    //  
    // When Open returns an error, it should be of type *PathError  
    // with the Op field set to "open", the Path field set to name,  
    // and the Err field describing the problem.  
    //  
    // Open should reject attempts to open names that do not satisfy  
    // ValidPath(name), returning a *PathError with Err set to  
    // ErrInvalid or ErrNotExist.  
    Open(name string) (File, error)  
}
```

An FS provides access to a hierarchical file system.

io/fs パッケージの fs.ReadFile

func ReadFile

```
func ReadFile(fsys FS, name string) ([]byte, error)
```

ReadFile reads the named file from the file system fs and returns its contents.

os.ReadFile → fs.ReadFile に置き換える

```
func joinFiles(filesystem fs.FS, name1, name2 string) error {  
    body1, _ := fs.ReadFile(filesystem, name1)  
  
    body2, _ := fs.ReadFile(filesystem, name2)  
  
    combinedFile := joinFileBody(body1, body2)  
  
    _ = os.WriteFile("joined.txt", combinedFile, 0755)  
  
    return nil  
}
```

mainではファイルシステムを渡す

```
func main() {  
    |   joinFiles(os.DirFS("."), "a.txt", "b.txt")  
    |  
}
```

テストでは fstest.MapFS を渡す

```
func TestJoinFiles(t *testing.T) {  
    mapFS := fstest.MapFS{  
        "a.txt": {Data: []byte("aaa")},  
        "b.txt": {Data: []byte("bbb")},  
    }  
  
    joinFiles(mapFS, "a.txt", "b.txt")  
}
```

これでInputのテストは書きやすい！

- Input部分は好きに単体テスト書くだけ！
- じゃあ、Outputのときにファイル書き出せるかのテストって...?
- io/fs.FS は基本読み取りだけ..。



t.TempDir で一時ファイルを作ろう

t.TempDir

```
func (*T) TempDir
```

```
func (c *T) TempDir() string
```

TempDir returns a temporary directory for the test to use. The directory is automatically removed when the test completes. Each subsequent call to TempDir returns a unique directory; if the directory creation fails, it calls Fatal.

テスト用の一時的なディレクトリを作ってくれる

```
run test | debug test
func TestA(t *testing.T) {
    dir := t.TempDir()

    name := filepath.Join(dir, "a.txt")

    fmt.Println(name) // /tmp/TestA2119408837/001/a.txt

    _ = os.WriteFile(name, []byte("aaa"), 0755)
}
```

ファイル書き出し後に、それを確かめる

```
func TestJoinFiles(t *testing.T) {  
    dir := t.TempDir()  
    _ = os.Chdir(dir)  
  
    mapFS := fstest.MapFS{  
        "a.txt": {Data: []byte("aaa")},  
        "b.txt": {Data: []byte("bbb")},  
    }  
  
    joinFiles(mapFS, "a.txt", "b.txt")  
  
    body, _ := os.ReadFile("joined.txt")  
  
    if string(body) != "aaa\nbbb" {  
        t.Error("NO!!!")  
    }  
}
```

ファイ

```
func TestJoinFiles(t *testing.T) {  
    dir := t.TempDir()  
    _ = os.Chdir(dir)  
  
    mapFS := fstest.MapFS{  
        "a.txt": {Data: []byte("aaa")},  
        "b.txt": {Data: []byte("bbb")},  
    }  
  
    joinFiles(mapFS, "a.txt", "b.txt")  
  
    body, _ := os.ReadFile("joined.txt")  
  
    if string(body) != "aaa\nbbb" {  
        t.Error("NO!!!")  
    }  
}
```

まとめ

- testing.T, testing.B, testing.PB, ... は便利
- testing.TB はTとB、両方の性質を併せ持つ...♥
- testing/fstest はファイルシステムのテストに便利！

おしまい

testing/iotest, testing/synctest
もおもしろい