

-gfile

gfile/

```
import "github.com/gogf/gf/v2/os/gfile"
```

<https://pkg.go.dev/github.com/gogf/gf/v2/os/gfile>

 <https://pkg.go.dev/github.com/gogf/gf/v2/os/gfile>

GetContents

-
- :

```
func GetContents(path string) string
```

-

```
func ExampleGetContents() {
    // init
    var (
        fileName = "gfile_example.txt"
        tempDir  = gfile.TempDir("gfile_example_content")
        tempFile = gfile.Join(tempDir, fileName)
    )

    // write contents
    gfile.PutContents(tempFile, "goframe example content")

    // It reads and returns the file content as string.
    // It returns empty string if it fails reading, for example,
    with permission or IO error.
    fmt.Println(gfile.GetContents(tempFile))

    // Output:
    // goframe example content
}
```

GetContentsWithCache

-
- :

```
func GetContentsWithCache(path string, duration ...time.Duration)
string
```

-

Content Menu

-
- - GetContents
 - GetContentsWithCache
 - GetBytesWithCache
 - GetBytes
 - GetBytesTilChar
 - GetBytesByTwoOffsets
 - PutContents
 - PutBytes
 - PutContentsAppend
 - PutBytesAppend
 - GetNextCharOffset
 - GetNextCharOffsetByPath
 - GetBytesTilCharByPath
 - GetBytesByTwoOffsetsByPath
 - ReadLines
 - ReadLinesBytes
 - Truncate
- - ReplaceFile
 - ReplaceFileFunc
 - ReplaceDir
 - ReplaceDirFunc
- - MTime
 - MTimestamp
 - MTimestampMilli
- - Size
 - SizeFormat
 - ReadableSize
 - StrToSize
 - FormatSize
- - SortFiles
- - Search
- - ScanDir
 - ScanDirFile
 - ScanDirFunc
 - ScanDirFileFunc
- - Pwd
 - Home
 - Temp
 - SelfPath
- - IsDir
 - IsFile
- - IsReadable
 - IsWritable
 - Chmod

```

func ExampleGetContentsWithCache() {
    // init
    var (
        fileName = "gfile_example.txt"
        tempDir   = gfile.TempDir("gfile_example_cache")
        tempFile = gfile.Join(tempDir, fileName)
    )

    // write contents
    gfile.PutContents(tempFile, "goframe example content")

    // It reads the file content with cache duration of one
minute,
    // which means it reads from cache after then without any IO
operations within on minute.
    fmt.Println(gfile.GetContentsWithCache(tempFile, time.
Minute))

    // write new contents will clear its cache
    gfile.PutContents(tempFile, "new goframe example content")

    // There's some delay for cache clearing after file content
change.
    time.Sleep(time.Second * 1)

    // read contents
    fmt.Println(gfile.GetContentsWithCache(tempFile))

    // May Output:
    // goframe example content
    // new goframe example content
}

```

- /
 - Mkdir
 - Create
 - Open
 - OpenFile
 - OpenWithFalg
 - OpenWithFalgPerm
 - Stat
 - Copy
 - CopyFile
 - CopyDir
 - Move
 - Rename
 - Remove
 - IsEmpty
 - DirNames
 - Glob
 - Exists
 - Chdir
- - Join
 - Abs
 - RealPath
 - SelfName
 - Basename
 - Name
 - Dir
 - Ext
 - ExtName
 - MainPkgPath

GetBytesWithCache

- []byte
- :

```

func GetBytesWithCache(path string, duration ...time.Duration) []byte

```

-

```

func ExampleGetBytesWithCache() {
    // init
    var (
        fileName = "gfile_example.txt"
        tempDir   = gfile.TempDir("gfile_example_cache")
        tempFile  = gfile.Join(tempDir, fileName)
    )

    // write contents
    gfile.PutContents(tempFile, "goframe example content")

    // It reads the file content with cache duration of one
minute,
    // which means it reads from cache after then without any IO
operations within on minute.
    fmt.Println(gfile.GetBytesWithCache(tempFile, time.Minute))

    // write new contents will clear its cache
    gfile.PutContents(tempFile, "new goframe example content")

    // There's some delay for cache clearing after file content
change.
    time.Sleep(time.Second * 1)

    // read contents
    fmt.Println(gfile.GetBytesWithCache(tempFile))

    // Output:
    // [103 111 102 114 97 109 101 32 101 120 97 109 112 108 101
32 99 111 110 116 101 110 116]
    // [110 101 119 32 103 111 102 114 97 109 101 32 101 120 97
109 112 108 101 32 99 111 110 116 101 110 116]
}

```

GetBytes

-
- :

```

func GetBytes(path string) []byte

```

-

```

func ExampleGetBytes() {
    // init
    var (
        fileName = "gfile_example.txt"
        tempDir   = gfile.TempDir("gfile_example_content")
        tempFile  = gfile.Join(tempDir, fileName)
    )

    // write contents
    gfile.PutContents(tempFile, "goframe example content")

    // It reads and returns the file content as []byte.
    // It returns nil if it fails reading, for example, with
permission or IO error.
    fmt.Println(gfile.GetBytes(tempFile))

    // Output:
    // [103 111 102 114 97 109 101 32 101 120 97 109 112 108 101
32 99 111 110 116 101 110 116]
}

```

GetBytesTilChar

-
- :

```
func GetBytesTilChar(reader io.ReaderAt, char byte, start int64) ([]byte, int64)
```

-

```
func ExampleGetBytesTilChar() {
    // init
    var (
        fileName = "gfile_example.txt"
        tempDir   = gfile.TempDir("gfile_example_content")
        tempFile = gfile.Join(tempDir, fileName)
    )

    // write contents
    gfile.PutContents(tempFile, "goframe example content")

    f, _ := gfile.OpenWithFlagPerm(tempFile, os.O_RDONLY, gfile.DefaultPermOpen)

    // GetBytesTilChar returns the contents of the file as []byte
    // until the next specified byte `char` position.
    char, i := gfile.GetBytesTilChar(f, 'f', 0)
    fmt.Println(char)
    fmt.Println(i)

    // Output:
    // [103 111 102]
    // 2
}
```

GetBytesByTwoOffsets

-
- :

```
func GetBytesByTwoOffsets(reader io.ReaderAt, start int64, end int64) []byte
```

-

```

func ExampleGetBytesByTwoOffsets() {
    // init
    var (
        fileName = "gfile_example.txt"
        tempDir   = gfile.TempDir("gfile_example_content")
        tempFile  = gfile.Join(tempDir, fileName)
    )

    // write contents
    gfile.PutContents(tempFile, "goframe example content")

    f, _ := gfile.OpenWithFlagPerm(tempFile, os.O_RDONLY, gfile.
DefaultPermOpen)

    // GetBytesTilChar returns the contents of the file as []byte
    // until the next specified byte `char` position.
    char := gfile.GetBytesByTwoOffsets(f, 0, 3)
    fmt.Println(char)

    // Output:
    // [103 111 102]
}

```

PutContents

-
- :

```

func putContents(path string, data []byte, flag int, perm os.
 FileMode) error

```

-

```

func ExamplePutContents() {
    // init
    var (
        fileName = "gfile_example.txt"
        tempDir   = gfile.TempDir("gfile_example_content")
        tempFile  = gfile.Join(tempDir, fileName)
    )

    // It creates and puts content string into specifies file
path.
    // It automatically creates directory recursively if it does
not exist.
    gfile.PutContents(tempFile, "goframe example content")

    // read contents
    fmt.Println(gfile.GetContents(tempFile))

    // Output:
    // goframe example content
}

```

PutBytes

-
- :

```

func PutBytes(path string, content []byte) error

```

-

```

func ExamplePutBytes() {
    // init
    var (
        fileName = "gfile_example.txt"
        tempDir   = gfile.TempDir("gfile_example_content")
        tempFile  = gfile.Join(tempDir, fileName)
    )

    // write contents
    gfile.PutBytes(tempFile, []byte("goframe example content"))

    // read contents
    fmt.Println(gfile.GetContents(tempFile))

    // Output:
    // goframe example content
}

```

PutContentsAppend

-
- :

```

func PutContentsAppend(path string, content string) error

```

-

```

func ExamplePutContentsAppend() {
    // init
    var (
        fileName = "gfile_example.txt"
        tempDir   = gfile.TempDir("gfile_example_content")
        tempFile  = gfile.Join(tempDir, fileName)
    )

    // write contents
    gfile.PutContents(tempFile, "goframe example content")

    // read contents
    fmt.Println(gfile.GetContents(tempFile))

    // It creates and append content string into specifies file
    path.
    // It automatically creates directory recursively if it does
    not exist.
    gfile.PutContentsAppend(tempFile, " append content")

    // read contents
    fmt.Println(gfile.GetContents(tempFile))

    // Output:
    // goframe example content
    // goframe example content append content
}

```

PutBytesAppend

-
- :

```

func PutBytesAppend(path string, content []byte) error

```

-

```
func ExamplePutBytesAppend() {
    // init
    var (
        fileName = "gfile_example.txt"
        tempDir   = gfile.TempDir("gfile_example_content")
        tempFile  = gfile.Join(tempDir, fileName)
    )

    // write contents
    gfile.PutContents(tempFile, "goframe example content")

    // read contents
    fmt.Println(gfile.GetContents(tempFile))

    // write contents
    gfile.PutBytesAppend(tempFile, []byte(" append"))

    // read contents
    fmt.Println(gfile.GetContents(tempFile))

    // Output:
    // goframe example content
    // goframe example content append
}
```

GetNextCharOffset

-
- :

```
func GetNextCharOffset(reader io.ReaderAt, char byte, start int64)
int64
```

-

```
func ExampleGetNextCharOffset() {
    // init
    var (
        fileName = "gfile_example.txt"
        tempDir   = gfile.TempDir("gfile_example_content")
        tempFile  = gfile.Join(tempDir, fileName)
    )

    // write contents
    gfile.PutContents(tempFile, "goframe example content")

    f, err := gfile.OpenWithFlagPerm(tempFile, os.O_RDONLY,
DefaultPermOpen)
    defer f.Close()

    // read contents
    index := gfile.GetNextCharOffset(f, 'f', 0)
    fmt.Println(index)

    // Output:
    // 2
}
```

GetNextCharOffsetByPath

-
- :

```
func GetNextCharOffsetByPath(path string, char byte, start int64)
int64
```

-

```
func ExampleGetNextCharOffsetByPath() {
    // init
    var (
        fileName = "gfile_example.txt"
        tempDir   = gfile.TempDir("gfile_example_content")
        tempFile  = gfile.Join(tempDir, fileName)
    )

    // write contents
    gfile.PutContents(tempFile, "goframe example content")

    // read contents
    index := gfile.GetNextCharOffsetByPath(tempFile, 'f', 0)
    fmt.Println(index)

    // Output:
    // 2
}
```

GetBytesTilCharByPath

-
- :

```
func GetBytesTilCharByPath(path string, char byte, start int64) ([]
byte, int64)
```

-

```
func ExampleGetBytesTilCharByPath() {
    // init
    var (
        fileName = "gfile_example.txt"
        tempDir   = gfile.TempDir("gfile_example_content")
        tempFile  = gfile.Join(tempDir, fileName)
    )

    // write contents
    gfile.PutContents(tempFile, "goframe example content")

    // read contents
    fmt.Println(gfile.GetBytesTilCharByPath(tempFile, 'f', 0))

    // Output:
    // [103 111 102] 2
}
```

GetBytesByTwoOffsetsByPath

-
- :

```
func GetBytesByTwoOffsetsByPath(path string, start int64, end int64)
[]byte
```


-

```
func ExampleGetBytesByTwoOffsetsByPath() {
    // init
    var (
        fileName = "gfile_example.txt"
        tempDir   = gfile.TempDir("gfile_example_content")
        tempFile  = gfile.Join(tempDir, fileName)
    )

    // write contents
    gfile.PutContents(tempFile, "goframe example content")

    // read contents
    fmt.Println(gfile.GetBytesByTwoOffsetsByPath(tempFile, 0, 7))

    // Output:
    // [103 111 102 114 97 109 101]
}
```

ReadLines

-
- :

```
func ReadLines(file string, callback func(text string) error) error
```

-

```
func ExampleReadLines() {
    // init
    var (
        fileName = "gfile_example.txt"
        tempDir   = gfile.TempDir("gfile_example_content")
        tempFile  = gfile.Join(tempDir, fileName)
    )

    // write contents
    gfile.PutContents(tempFile, "L1 goframe example content\nL2
goframe example content")

    // read contents
    gfile.ReadLines(tempFile, func(text string) error {
        // Process each line
        fmt.Println(text)
        return nil
    })

    // Output:
    // L1 goframe example content
    // L2 goframe example content
}
```

ReadLinesBytes

-
- :

```
func ReadLinesBytes(file string, callback func(bytes []byte) error)
error
```

-

```

func ExampleReadLinesBytes() {
    // init
    var (
        fileName = "gfile_example.txt"
        tempDir   = gfile.TempDir("gfile_example_content")
        tempFile  = gfile.Join(tempDir, fileName)
    )

    // write contents
    gfile.PutContents(tempFile, "L1 goframe example content\nL2
goframe example content")

    // read contents
    gfile.ReadLinesBytes(tempFile, func(bytes []byte) error {
        // Process each line
        fmt.Println(bytes)
        return nil
    })

    // Output:
    // [76 49 32 103 111 102 114 97 109 101 32 101 120 97 109
112 108 101 32 99 111 110 116 101 110 116]
    // [76 50 32 103 111 102 114 97 109 101 32 101 120 97 109
112 108 101 32 99 111 110 116 101 110 116]
}

```

Truncate

-
-
- :

```

func Truncate(path string, size int) error

```

-

```

func ExampleTruncate(){
    // init
    var (
        path = gfile.Join(gfile.TempDir
("gfile_example_basic_dir"), "file1")
    )

    // Check whether the `path` size
    stat, _ := gfile.Stat(path)
    fmt.Println(stat.Size())

    // Truncate file
    gfile.Truncate(path, 0)

    // Check whether the `path` size
    stat, _ = gfile.Stat(path)
    fmt.Println(stat.Size())

    // Output:
    // 13
    // 0
}

```

ReplaceFile

-
- :

```
func ReplaceFile(search, replace, path string) error
```

-

```
func ExampleReplaceFile() {
    // init
    var (
        fileName = "gfile_example.txt"
        tempDir   = gfile.TempDir("gfile_example_replace")
        tempFile = gfile.Join(tempDir, fileName)
    )

    // write contents
    gfile.PutContents(tempFile, "goframe example content")

    // read contents
    fmt.Println(gfile.GetContents(tempFile))

    // It replaces content directly by file path.
    gfile.ReplaceFile("content", "replace word", tempFile)

    fmt.Println(gfile.GetContents(tempFile))

    // Output:
    // goframe example content
    // goframe example replace word
}
```

ReplaceFileFunc

-
- :

```
func ReplaceFileFunc(f func(path, content string) string, path
string) error
```

-

```

func ExampleReplaceFileFunc() {
    // init
    var (
        fileName = "gfile_example.txt"
        tempDir   = gfile.TempDir("gfile_example_replace")
        tempFile  = gfile.Join(tempDir, fileName)
    )

    // write contents
    gfile.PutContents(tempFile, "goframe example 123")

    // read contents
    fmt.Println(gfile.GetContents(tempFile))

    // It replaces content directly by file path and callback
    function.
    gfile.ReplaceFileFunc(func(path, content string) string {
        // Replace with regular match
        reg, _ := regexp.Compile(`\d{3}`)
        return reg.ReplaceAllString(content, "[num]")
    }, tempFile)

    fmt.Println(gfile.GetContents(tempFile))

    // Output:
    // goframe example 123
    // goframe example [num]
}

```

ReplaceDir

-
- :

```

func ReplaceDir(search, replace, path, pattern string, recursive ...
bool) error

```

-

```

func ExampleReplaceDir() {
    // init
    var (
        fileName = "gfile_example.txt"
        tempDir   = gfile.TempDir("gfile_example_replace")
        tempFile  = gfile.Join(tempDir, fileName)
    )

    // write contents
    gfile.PutContents(tempFile, "goframe example content")

    // read contents
    fmt.Println(gfile.GetContents(tempFile))

    // It replaces content of all files under specified
    // directory recursively.
    gfile.ReplaceDir("content", "replace word", tempDir,
        "gfile_example.txt", true)

    // read contents
    fmt.Println(gfile.GetContents(tempFile))

    // Output:
    // goframe example content
    // goframe example replace word
}

```

ReplaceDirFunc

-
- :

```

func ReplaceDirFunc(f func(path, content string) string, path,
    pattern string, recursive ...bool) error

```

-

```

func ExampleReplaceDirFunc() {
    // init
    var (
        fileName = "gfile_example.txt"
        tempDir   = gfile.TempDir("gfile_example_replace")
        tempFile  = gfile.Join(tempDir, fileName)
    )

    // write contents
    gfile.PutContents(tempFile, "goframe example 123")

    // read contents
    fmt.Println(gfile.GetContents(tempFile))

    // It replaces content of all files under specified
    // directory with custom callback function recursively.
    gfile.ReplaceDirFunc(func(path, content string) string {
        // Replace with regular match
        reg, _ := regexp.Compile(`\d{3}`)
        return reg.ReplaceAllString(content, "[num]")
    }, tempDir, "gfile_example.txt", true)

    fmt.Println(gfile.GetContents(tempFile))

    // Output:
    // goframe example 123
    // goframe example [num]
}

```

MTime

-
- :

```
func MTime(path string) time.Time
```

-

```

func ExampleMTime() {
    t := gfile.MTime(gfile.TempDir())
    fmt.Println(t)

    // May Output:
    // 2021-11-02 15:18:43.901141 +0800 CST
}

```

MTimestamp

-
- :

```
func MTimestamp(path string) int64
```

-

```
func ExampleMTimestamp() {
    t := gfile.MTimestamp(gfile.TempDir())
    fmt.Println(t)

    // May Output:
    // 1635838398
}
```

MTimestampMilli

-
- :

```
func MTimestampMilli(path string) int64
```

-

```
func ExampleMTimestampMilli() {
    t := gfile.MTimestampMilli(gfile.TempDir())
    fmt.Println(t)

    // May Output:
    // 1635838529330
}
```

Size

-
- :

```
func Size(path string) int64
```

-

```
func ExampleSize() {
    // init
    var (
        fileName = "gfile_example.txt"
        tempDir   = gfile.TempDir("gfile_example_size")
        tempFile  = gfile.Join(tempDir, fileName)
    )

    // write contents
    gfile.PutContents(tempFile, "0123456789")
    fmt.Println(gfile.Size(tempFile))

    // Output:
    // 10
}
```

SizeFormat

-
- :

```
func SizeFormat(path string) string
```

-

```
func ExampleSizeFormat() {  
    // init  
    var (  
        fileName = "gfile_example.txt"  
        tempDir  = gfile.TempDir("gfile_example_size")  
        tempFile = gfile.Join(tempDir, fileName)  
    )  
  
    // write contents  
    gfile.PutContents(tempFile, "0123456789")  
    fmt.Println(gfile.SizeFormat(tempFile))  
  
    // Output:  
    // 10.00B  
}
```

ReadableSize

-
- :

```
func ReadableSize(path string) string
```

-

```
func ExampleReadableSize() {  
    // init  
    var (  
        fileName = "gfile_example.txt"  
        tempDir  = gfile.TempDir("gfile_example_size")  
        tempFile = gfile.Join(tempDir, fileName)  
    )  
  
    // write contents  
    gfile.PutContents(tempFile, "01234567899876543210")  
    fmt.Println(gfile.ReadableSize(tempFile))  
  
    // Output:  
    // 20.00B  
}
```

StrToSize

-
- :

```
func StrToSize(sizeStr string) int64
```

-


```
func ExampleStrToSize() {
    size := gfile.StrToSize("100MB")
    fmt.Println(size)

    // Output:
    // 104857600
}
```

FormatSize

- `Kmgtpb`
- :

```
func FormatSize(raw int64) string
```

-

```
func ExampleFormatSize() {
    sizeStr := gfile.FormatSize(104857600)
    fmt.Println(sizeStr)
    sizeStr0 := gfile.FormatSize(1024)
    fmt.Println(sizeStr0)
    sizeStr1 := gfile.FormatSize(999999999999999999)
    fmt.Println(sizeStr1)

    // Output:
    // 100.00M
    // 1.00K
    // 888.18P
}
```

SortFiles

-
- :

```
func SortFiles(files []string) []string
```

-

```

func ExampleSortFiles() {
    files := []string{
        "/aaa/bbb/ccc.txt",
        "/aaa/bbb/",
        "/aaa/",
        "/aaa",
        "/aaa/ccc/ddd.txt",
        "/bbb",
        "/0123",
        "/ddd",
        "/ccc",
    }
    sortOut := gfile.SortFiles(files)
    fmt.Println(sortOut)

    // Output:
    // [/0123 /aaa /aaa/ /aaa/bbb/ /aaa/bbb/ccc.txt /aaa/ccc/ddd.
    txt /bbb /ccc /ddd]
}

```

Search

-
- :

```

func Search(name string, prioritySearchPaths ...string) (realPath
string, err error)

```

-

```

func ExampleSearch() {
    // init
    var (
        fileName = "gfile_example.txt"
        tempDir   = gfile.TempDir("gfile_example_search")
        tempFile  = gfile.Join(tempDir, fileName)
    )

    // write contents
    gfile.PutContents(tempFile, "goframe example content")

    // search file
    realPath, _ := gfile.Search(fileName, tempDir)
    fmt.Println(gfile.Basename(realPath))

    // Output:
    // gfile_example.txt
}

```

ScanDir

-
- :

```
func ScanDir(path string, pattern string, recursive ...bool) ([]
string, error)
```

-

```
func ExampleScanDir() {
    // init
    var (
        fileName = "gfile_example.txt"
        tempDir   = gfile.TempDir("gfile_example_scan_dir")
        tempFile  = gfile.Join(tempDir, fileName)

        tempSubDir = gfile.Join(tempDir, "sub_dir")
        tempSubFile = gfile.Join(tempSubDir, fileName)
    )

    // write contents
    gfile.PutContents(tempFile, "goframe example content")
    gfile.PutContents(tempSubFile, "goframe example content")

    // scans directory recursively
    list, _ := gfile.ScanDir(tempDir, "*", true)
    for _, v := range list {
        fmt.Println(gfile.Basename(v))
    }

    // Output:
    // gfile_example.txt
    // sub_dir
    // gfile_example.txt
}
```

ScanDirFile

-
- :

```
func ScanDirFile(path string, pattern string, recursive ...bool) ([]
string, error)
```

-

```

func ExampleScanDirFile() {
    // init
    var (
        fileName = "gfile_example.txt"
        tempDir   = gfile.TempDir
    ("gfile_example_scan_dir_file")
        tempFile = gfile.Join(tempDir, fileName)

        tempSubDir = gfile.Join(tempDir, "sub_dir")
        tempSubFile = gfile.Join(tempSubDir, fileName)
    )

    // write contents
    gfile.PutContents(tempFile, "goframe example content")
    gfile.PutContents(tempSubFile, "goframe example content")

    // scans directory recursively exclusive of directories
    list, _ := gfile.ScanDirFile(tempDir, "*.txt", true)
    for _, v := range list {
        fmt.Println(gfile.Basename(v))
    }

    // Output:
    // gfile_example.txt
    // gfile_example.txt
}

```

ScanDirFunc

-
- :

```

func ScanDirFunc(path string, pattern string, recursive bool,
    handler func(path string) string) ([]string, error)

```

-

```

func ExampleScanDirFunc() {
    // init
    var (
        fileName = "gfile_example.txt"
        tempDir   = gfile.TempDir
    ("gfile_example_scan_dir_func")
        tempFile = gfile.Join(tempDir, fileName)

        tempSubDir = gfile.Join(tempDir, "sub_dir")
        tempSubFile = gfile.Join(tempSubDir, fileName)
    )

    // write contents
    gfile.PutContents(tempFile, "goframe example content")
    gfile.PutContents(tempSubFile, "goframe example content")

    // scans directory recursively
    list, _ := gfile.ScanDirFunc(tempDir, "*", true, func(path
string) string {
        // ignores some files
        if gfile.BaseName(path) == "gfile_example.txt" {
            return ""
        }
        return path
    })
    for _, v := range list {
        fmt.Println(gfile.BaseName(v))
    }

    // Output:
    // sub_dir
}

```

ScanDirFileFunc

-
- :

```

func ScanDirFileFunc(path string, pattern string, recursive bool,
handler func(path string) string) ([]string, error)

```

-

```

func ExampleScanDirFileFunc() {
    // init
    var (
        fileName = "gfile_example.txt"
        tempDir   = gfile.TempDir
        ("gfile_example_scan_dir_file_func")
        tempFile = gfile.Join(tempDir, fileName)

        fileName1 = "gfile_example_ignores.txt"
        tempFile1  = gfile.Join(tempDir, fileName1)

        tempSubDir = gfile.Join(tempDir, "sub_dir")
        tempSubFile = gfile.Join(tempSubDir, fileName)
    )

    // write contents
    gfile.PutContents(tempFile, "goframe example content")
    gfile.PutContents(tempFile1, "goframe example content")
    gfile.PutContents(tempSubFile, "goframe example content")

    // scans directory recursively exclusive of directories
    list, _ := gfile.ScanDirFileFunc(tempDir, "*.txt", true, func
(path string) string {
        // ignores some files
        if gfile.BaseName(path) == "gfile_example_ignores.
txt" {
            return ""
        }
        return path
    })
    for _, v := range list {
        fmt.Println(gfile.BaseName(v))
    }

    // Output:
    // gfile_example.txt
    // gfile_example.txt
}

```

Pwd

-
- :

```
func Pwd() string
```

-

```

func ExamplePwd() {
    // Get absolute path of current working directory.
    fmt.Println(gfile.Pwd())

    // May Output:
    // xxx/gf/os/gfile
}

```

Home

-
- :

```
func Home(names ...string) (string, error)
```

-

```
func ExampleHome() {  
    // user's home directory  
    homePath, _ := gfile.Home()  
    fmt.Println(homePath)  
  
    // May Output:  
    // C:\Users\hailaz  
}
```

Temp

-
- :

```
func Temp(names ...string) string
```

-

```
func ExampleTempDir() {  
    // init  
    var (  
        fileName = "gfile_example_basic_dir"  
    )  
  
    // fetch an absolute representation of path.  
    path := gfile.Temp(fileName)  
  
    fmt.Println(path)  
  
    // Output:  
    // /tmp/gfile_example_basic_dir  
}
```

SelfPath

-
- :

```
func SelfPath() string
```

-

```
func ExampleSelfPath() {  
  
    // Get absolute file path of current running process  
    fmt.Println(gfile.SelfPath())  
  
    // May Output:  
    // xxx/___github_com_gogf_gf_v2_os_gfile__ExampleSelfPath  
}
```

IsDir

-
- :

```
func IsDir(path string) bool
```

-

```
func ExampleIsDir() {  
    // init  
    var (  
        path      = gfile.TempDir("gfile_example_basic_dir")  
        filePath = gfile.Join(gfile.TempDir  
("gfile_example_basic_dir"), "file1")  
    )  
    // Checks whether given `path` a directory.  
    fmt.Println(gfile.IsDir(path))  
    fmt.Println(gfile.IsDir(filePath))  
  
    // Output:  
    // true  
    // false  
}
```

IsFile

-
- :

```
func IsFile(path string) bool
```

-

```
func ExampleIsFile() {  
    // init  
    var (  
        filePath = gfile.Join(gfile.TempDir  
("gfile_example_basic_dir"), "file1")  
        dirPath  = gfile.TempDir("gfile_example_basic_dir")  
    )  
    // Checks whether given `path` a file, which means it's not  
a directory.  
    fmt.Println(gfile.IsFile(filePath))  
    fmt.Println(gfile.IsFile(dirPath))  
  
    // Output:  
    // true  
    // false  
}
```

IsReadable

-
- :

```
func IsReadable(path string) bool
```

-


```
func ExampleIsReadable() {
    // init
    var (
        path = gfile.Pwd() + gfile.Separator + "testdata
/readline/file.log"
    )

    // Checks whether given `path` is readable.
    fmt.Println(gfile.IsReadable(path))

    // Output:
    // true
}
```

IsWritable

-
- :

```
func IsWritable(path string) bool
```

-

```
func ExampleIsWritable() {
    // init
    var (
        path = gfile.Pwd() + gfile.Separator + "testdata
/readline/file.log"
    )

    // Checks whether given `path` is writable.
    fmt.Println(gfile.IsWritable(path))

    // Output:
    // true
}
```

Chmod

-
- :

```
func Chmod(path string, mode os.FileMode) error
```

-

```

func ExampleChmod() {
    // init
    var (
        path = gfile.Join(gfile.TempDir
("gfile_example_basic_dir"), "file1")
    )

    // Get a FileInfo describing the named file.
    stat, err := gfile.Stat(path)
    if err != nil {
        fmt.Println(err.Error())
    }
    // Show original mode
    fmt.Println(stat.Mode())

    // Change file model
    gfile.Chmod(path, gfile.DefaultPermCopy)

    // Get a FileInfo describing the named file.
    stat, _ = gfile.Stat(path)
    // Show the modified mode
    fmt.Println(stat.Mode())

    // Output:
    // -rw-r--r--
    // -rwxrwxrwx
}

```

/

Mkdir

- ,drwxr-xr-x
- :

```
func Mkdir(path string) error
```

-

```

func ExampleMkdir() {
    // init
    var (
        path = gfile.TempDir("gfile_example_basic_dir")
    )

    // Creates directory
    gfile.Mkdir(path)

    // Check if directory exists
    fmt.Println(gfile.IsDir(path))

    // Output:
    // true
}

```

Create

- /,-rw-r-r-
- :

```
func Create(path string) (*os.File, error)
```

-

```
func ExampleCreate() {
    // init
    var (
        path      = gfile.Join(gfile.TempDir
("gfile_example_basic_dir"), "file1")
        dataByte = make([]byte, 50)
    )
    // Check whether the file exists
    isFile := gfile.IsFile(path)

    fmt.Println(isFile)

    // Creates file with given `path` recursively
    fileHandle, _ := gfile.Create(path)
    defer fileHandle.Close()

    // Write some content to file
    n, _ := fileHandle.WriteString("hello goframe")

    // Check whether the file exists
    isFile = gfile.IsFile(path)

    fmt.Println(isFile)

    // Reset file uintptr
    unix.Seek(int(fileHandle.Fd()), 0, 0)
    // Reads len(b) bytes from the File
    fileHandle.Read(dataByte)

    fmt.Println(string(dataByte[:n]))

    // Output:
    // false
    // true
    // hello goframe
}
```

Open

- /
- :

```
func Open(path string) (*os.File, error)
```

-

```

func ExampleOpen() {
    // init
    var (
        path      = gfile.Join(gfile.TempDir
("gfile_example_basic_dir"), "file1")
        dataByte = make([]byte, 4096)
    )
    // Open file or directory with READONLY model
    file, _ := gfile.Open(path)
    defer file.Close()

    // Read data
    n, _ := file.Read(dataByte)

    fmt.Println(string(dataByte[:n]))

    // Output:
    // hello goframe
}

```

OpenFile

- `flag` `perm`/
- :

```

func OpenFile(path string, flag int, perm os.FileMode) (*os.File,
error)

```

-

```

func ExampleOpenFile() {
    // init
    var (
        path      = gfile.Join(gfile.TempDir
("gfile_example_basic_dir"), "file1")
        dataByte = make([]byte, 4096)
    )
    // Opens file/directory with custom `flag` and `perm`
    // Create if file does not exist, it is created in a readable
and writable mode, perm 0777
    openFile, _ := gfile.OpenFile(path, os.O_CREATE|os.O_RDWR,
gfile.DefaultPermCopy)
    defer openFile.Close()

    // Write some content to file
    writeLength, _ := openFile.WriteString("hello goframe test
open file")

    fmt.Println(writeLength)

    // Read data
    unix.Seek(int(openFile.Fd()), 0, 0)
    n, _ := openFile.Read(dataByte)

    fmt.Println(string(dataByte[:n]))

    // Output:
    // 28
    // hello goframe test open file
}

```

OpenWithFalg

- `flag`/
- :

```
func OpenWithFlag(path string, flag int) (*os.File, error)
```

-

```
func ExampleOpenWithFlag() {
    // init
    var (
        path      = gfile.Join(gfile.TempDir
("gfile_example_basic_dir"), "file1")
        dataByte = make([]byte, 4096)
    )

    // Opens file/directory with custom `flag`
    // Create if file does not exist, it is created in a readable
and writable mode with default `perm` is 0666
    openFile, _ := gfile.OpenWithFlag(path, os.O_CREATE|os.
O_RDWR)
    defer openFile.Close()

    // Write some content to file
    writeLength, _ := openFile.WriteString("hello goframe test
open file with flag")

    fmt.Println(writeLength)

    // Read data
    unix.Seek(int(openFile.Fd()), 0, 0)
    n, _ := openFile.Read(dataByte)

    fmt.Println(string(dataByte[:n]))

    // Output:
    // 38
    // hello goframe test open file with flag
}
```

OpenWithFalgPerm

- `flag`perm`/
- :

```
func OpenWithFlagPerm(path string, flag int, perm os.FileMode) (*os.
File, error)
```

-

```

func ExampleOpenWithFlagPerm() {
    // init
    var (
        path      = gfile.Join(gfile.TempDir
("gfile_example_basic_dir"), "file1")
        dataByte = make([]byte, 4096)
    )

    // Opens file/directory with custom `flag` and `perm`
    // Create if file does not exist, it is created in a readable
and writable mode with `perm` is 0777
    openFile, _ := gfile.OpenWithFlagPerm(path, os.O_CREATE|os.
O_RDWR, gfile.DefaultPermCopy)
    defer openFile.Close()

    // Write some content to file
    writeLength, _ := openFile.WriteString("hello goframe test
open file with flag and perm")

    fmt.Println(writeLength)

    // Read data
    unix.Seek(int(openFile.Fd()), 0, 0)
    n, _ := openFile.Read(dataByte)

    fmt.Println(string(dataByte[:n]))

    // Output:
    // 38
    // hello goframe test open file with flag
}

```

Stat

-
- :

```

func Stat(path string) (os.FileInfo, error)

```

-

```

func ExampleStat() {
    // init
    var (
        path = gfile.Join(gfile.TempDir
("gfile_example_basic_dir"), "file1")
    )
    // Get a FileInfo describing the named file.
    stat, _ := gfile.Stat(path)

    fmt.Println(stat.Name())
    fmt.Println(stat.IsDir())
    fmt.Println(stat.Mode())
    fmt.Println(stat.ModTime())
    fmt.Println(stat.Size())
    fmt.Println(stat.Sys())

    // May Output:
    // file1
    // false
    // -rwxr-xr-x
    // 2021-12-02 11:01:27.261441694 +0800 CST
    // &{16777220 33261 1 8597857090 501 20 0 [0 0 0 0]
{1638414088 192363490} {1638414087 261441694} {1638414087 261441694}
{1638413480 485068275} 38 8 4096 0 0 0 [0 0]}
    }
}

```

Copy

-
- :

```

func Copy(src string, dst string) error

```

-

```

func ExampleCopy() {
    // init
    var (
        srcFileName = "gfile_example.txt"
        srcTempDir   = gfile.TempDir("gfile_example_copy_src")
        srcTempFile = gfile.Join(srcTempDir, srcFileName)

        // copy file
        dstFileName = "gfile_example_copy.txt"
        dstTempFile = gfile.Join(srcTempDir, dstFileName)

        // copy dir
        dstTempDir = gfile.TempDir("gfile_example_copy_dst")
    )

    // write contents
    gfile.PutContents(srcTempFile, "goframe example copy")

    // copy file
    gfile.Copy(srcTempFile, dstTempFile)

    // read contents after copy file
    fmt.Println(gfile.GetContents(dstTempFile))

    // copy dir
    gfile.Copy(srcTempDir, dstTempDir)

    // list copy dir file
    fList, _ := gfile.ScanDir(dstTempDir, "*", false)
    for _, v := range fList {
        fmt.Println(gfile.Basename(v))
    }

    // Output:
    // goframe example copy
    // gfile_example.txt
    // gfile_example_copy.txt
}

```

CopyFile

-
- :

```
func CopyFile(src, dst string) (err error)
```

-


```

func ExampleCopyFile() {
    // init
    var (
        srcFileName = "gfile_example.txt"
        srcTempDir   = gfile.TempDir("gfile_example_copy_src")
        srcTempFile = gfile.Join(srcTempDir, srcFileName)

        // copy file
        dstFileName = "gfile_example_copy.txt"
        dstTempFile = gfile.Join(srcTempDir, dstFileName)
    )

    // write contents
    gfile.PutContents(srcTempFile, "goframe example copy")

    // copy file
    gfile.CopyFile(srcTempFile, dstTempFile)

    // read contents after copy file
    fmt.Println(gfile.GetContents(dstTempFile))

    // Output:
    // goframe example copy
}

```

CopyDir

- :

```
func CopyDir(src string, dst string) error
```

-

```

func ExampleCopyDir() {
    // init
    var (
        srcTempDir = gfile.TempDir("gfile_example_copy_src")

        // copy file
        dstFileName = "gfile_example_copy.txt"
        dstTempFile = gfile.Join(srcTempDir, dstFileName)

        // copy dir
        dstTempDir = gfile.TempDir("gfile_example_copy_dst")
    )
    // read contents after copy file
    fmt.Println(gfile.GetContents(dstTempFile))

    // copy dir
    gfile.CopyDir(srcTempDir, dstTempDir)

    // list copy dir file
    fList, _ := gfile.ScanDir(dstTempDir, "", false)
    for _, v := range fList {
        fmt.Println(gfile.BaseName(v))
    }

    // Output:
    // gfile_example.txt
    // gfile_example_copy.txt
}

```

Move

- srcdst
- dst
- :

```
func Move(src string, dst string) error
```

-

```
func ExampleMove() {
    // init
    var (
        srcPath = gfile.Join(gfile.TempDir
("gfile_example_basic_dir"), "file1")
        dstPath = gfile.Join(gfile.TempDir
("gfile_example_basic_dir"), "file2")
    )
    // Check is file
    fmt.Println(gfile.IsFile(dstPath))

    // Moves `src` to `dst` path.
    // If `dst` already exists and is not a directory, it'll be
    replaced.
    gfile.Move(srcPath, dstPath)

    fmt.Println(gfile.IsFile(srcPath))
    fmt.Println(gfile.IsFile(dstPath))

    // Output:
    // false
    // false
    // true
}
```

Rename

- Movesrcdst
- dst
- :

```
func Rename(src string, dst string) error
```

-

```

func ExampleRename() {
    // init
    var (
        srcPath = gfile.Join(gfile.TempDir
("gfile_example_basic_dir"), "file2")
        dstPath = gfile.Join(gfile.TempDir
("gfile_example_basic_dir"), "file1")
    )
    // Check is file
    fmt.Println(gfile.IsFile(dstPath))

    // renames (moves) `src` to `dst` path.
    // If `dst` already exists and is not a directory, it'll be
    replaced.
    gfile.Rename(srcPath, dstPath)

    fmt.Println(gfile.IsFile(srcPath))
    fmt.Println(gfile.IsFile(dstPath))

    // Output:
    // false
    // false
    // true
}

```

Remove

-
- :

```
func Remove(path string) error
```

-

```

func ExampleRemove() {
    // init
    var (
        path = gfile.Join(gfile.TempDir
("gfile_example_basic_dir"), "file1")
    )

    // Checks whether given `path` a file, which means it's not
    a directory.
    fmt.Println(gfile.IsFile(path))

    // deletes all file/directory with `path` parameter.
    gfile.Remove(path)

    // Check again
    fmt.Println(gfile.IsFile(path))

    // Output:
    // true
    // false
}

```

IsEmpty

-
- :

```
func IsEmpty(path string) bool
```

-

```
func ExampleIsEmpty() {  
    // init  
    var (  
        path = gfile.Join(gfile.TempDir  
("gfile_example_basic_dir"), "file1")  
    )  
  
    // Check whether the `path` is empty  
    fmt.Println(gfile.IsEmpty(path))  
  
    // Truncate file  
    gfile.Truncate(path, 0)  
  
    // Check whether the `path` is empty  
    fmt.Println(gfile.IsEmpty(path))  
  
    // Output:  
    // false  
    // true  
}
```

DirNames

-
- :

```
func DirNames(path string) ([]string, error)
```

-

```
func ExampleDirNames() {  
    // init  
    var (  
        path = gfile.TempDir("gfile_example_basic_dir")  
    )  
    // Get sub-file names of given directory `path`.  
    dirNames, _ := gfile.DirNames(path)  
  
    fmt.Println(dirNames)  
  
    // May Output:  
    // [file1]  
}
```

Glob

-
- :

```
func Glob(pattern string, onlyNames ...bool) ([]string, error)
```

-

```

func ExampleGlob() {
    // init
    var (
        path = gfile.Pwd() + gfile.Separator +
        "_example_basic_test.go"
    )
    // Get sub-file names of given directory `path`.
    // Only show file name
    matchNames, _ := gfile.Glob(path, true)

    fmt.Println(matchNames)

    // Show full path of the file
    matchNames, _ = gfile.Glob(path, false)

    fmt.Println(matchNames)

    // May Output:
    // [gfile_z_example_basic_test.go]
    // [xxx/gf/os/gfile/gfile_z_example_basic_test.go]
}

```

Exists

-
- :

```

func Exists(path string) bool

```

-

```

func ExampleExists() {
    // init
    var (
        path = gfile.Join(gfile.TempDir
        ("gfile_example_basic_dir"), "file1")
    )
    // Checks whether given `path` exist.
    fmt.Println(gfile.Exists(path))

    // Output:
    // true
}

```

Chdir

-
- :

```

func Chdir(dir string) error

```

-

```

func ExampleChdir() {
    // init
    var (
        path = gfile.Join(gfile.TempDir
("gfile_example_basic_dir", "file1")
    )
    // Get current working directory
    fmt.Println(gfile.Pwd())

    // Changes the current working directory to the named
directory.
    gfile.Chdir(path)

    // Get current working directory
    fmt.Println(gfile.Pwd())

    // May Output:
    // xxx/gf/os/gfile
    // /tmp/gfile_example_basic_dir/file1
}

```

Join

- `
- :

```

func Join(paths ...string) string

```

-

```

func ExampleJoin() {
    // init
    var (
        dirPath = gfile.TempDir("gfile_example_basic_dir")
        filePath = "file1"
    )

    // Joins string array paths with file separator of current
system.
    joinString := gfile.Join(dirPath, filePath)

    fmt.Println(joinString)

    // Output:
    // /tmp/gfile_example_basic_dir/file1
}

```

Abs

-
- :

```

func Abs(path string) string

```

-

```

func ExampleAbs() {
    // init
    var (
        path = gfile.Join(gfile.TempDir
("gfile_example_basic_dir"), "file1")
    )

    // Get an absolute representation of path.
    fmt.Println(gfile.Abs(path))

    // Output:
    // /tmp/gfile_example_basic_dir/file1
}

```

RealPath

-
-
- :

```

func RealPath(path string) string

```

-

```

func ExampleRealPath() {
    // init
    var (
        realPath = gfile.Join(gfile.TempDir
("gfile_example_basic_dir"), "file1")
        worryPath = gfile.Join(gfile.TempDir
("gfile_example_basic_dir"), "worryFile")
    )

    // fetch an absolute representation of path.
    fmt.Println(gfile.RealPath(realPath))
    fmt.Println(gfile.RealPath(worryPath))

    // Output:
    // /tmp/gfile_example_basic_dir/file1
    //
}

```

SelfName

-
- :

```

func SelfName() string

```

-

```

func ExampleSelfName() {

    // Get file name of current running process
    fmt.Println(gfile.SelfName())

    // May Output:
    // __github_com_gogf_gf_v2_os_gfile__ExampleSelfName
}

```

BaseName

-
- :

```
func BaseName(path string) string
```

-

```
func ExampleBaseName() {  
    // init  
    var (  
        path = gfile.Pwd() + gfile.Separator + "testdata  
/readline/file.log"  
    )  
  
    // Get the last element of path, which contains file  
extension.  
    fmt.Println(gfile.BaseName(path))  
  
    // Output:  
    // file.log  
}
```

Name

-
- :

```
func Name(path string) string
```

-

```
func ExampleName() {  
    // init  
    var (  
        path = gfile.Pwd() + gfile.Separator + "testdata  
/readline/file.log"  
    )  
  
    // Get the last element of path without file extension.  
    fmt.Println(gfile.Name(path))  
  
    // Output:  
    // file  
}
```

Dir

-
- :

```
func Dir(path string) string
```

-


```
func ExampleDir() {
    // init
    var (
        path = gfile.Join(gfile.TempDir
("gfile_example_basic_dir", "file1")
    )

    // Get all but the last element of path, typically the
    path's directory.
    fmt.Println(gfile.Dir(path))

    // Output:
    // /tmp/gfile_example_basic_dir
}
```

Ext

- ``
- :

```
func Ext(path string) string
```

-

```
func ExampleExt() {
    // init
    var (
        path = gfile.Pwd() + gfile.Separator + "testdata
/readline/file.log"
    )

    // Get the file name extension used by path.
    fmt.Println(gfile.Ext(path))

    // Output:
    // .log
}
```

ExtName

- ``
- :

```
func ExtName(path string) string
```

-

```

func ExampleExtName() {
    // init
    var (
        path = gfile.Pwd() + gfile.Separator + "testdata
/readline/file.log"
    )

    // Get the file name extension used by path but the result
    does not contains symbol '.'.
    fmt.Println(gfile.ExtName(path))

    // Output:
    // log
}

```

MainPkgPath

- main
- - build
 - goroutine
- :

```

func MainPkgPath() string

```

-

```

func Test() {
    fmt.Println("main pkg path on main :", gfile.MainPkgPath())
    char := make(chan int, 1)
    go func() {
        fmt.Println("main pkg path on goroutine :", gfile.
MainPkgPath())
        char <- 1
    }()
    select {
    case <-char:
    }
    // Output:
    // /xxx/xx/xxx/xx
    // /xxx/xx/xxx/xx
}
//
$ ./testDemo
main pkg path on main : /xxx/xx/xxx/xx
main pkg path on goroutine : /xxx/xx/xxx/xx

```