

### Basic Operations

create empty file `newFile, err := os.Create("test.txt")`

truncate a file `err := os.Truncate("test.txt", 100)`

get file info `fileInfo, err := os.Stat("test.txt")`

rename a file `err := os.Rename(oldPath, newPath)`

delete a file `err := os.Remove("test.txt")`

open a file for reading `file, err := os.Open("test.txt")`

open a file `file, err := os.Open("test.txt", os.O_APPEND, 0600)`

close a file `err := file.Close()`

change file permission `err := os.Chmod("test.txt", 0777)`

change file ownership `err := os.Chown("test.txt", os.Getuid(), os.Getgid())`

change file timestamps `err := os.Chtimes("test.txt", lastAccessTime, lastModifyTime)`

### file open flag

`os.O_RDONLY` open the file read only

`os.O_WRONLY` open the file write only

`os.O_RDWR` open the file read write

`os.O_APPEND` append data to the file when writing

`os.O_CREATE` create a new file if none exists

`os.O_EXCL` used with `O_CREATE`, file must not exist

`os.O_SYNC` open for synchronous I/O

`O_TRUNC` if possible, truncate file when opened

When opening file with `os.OpenFile`, flags control how the file behaves.

### Hard Link & Symbol Link

create a hard link `err := os.Link("test.txt", "test_copy.txt")`

create a symbol link `err := os.Symlink("test.txt", "test_sym.txt")`

get link file info `fileInfo, err := os.Lstat("test_sym.txt")`

change link file owner `err := os.Lchown("test_sym.txt", uid, gid)`

read a link `dest, err := os.ReadLink("link_file.txt")`

A hard link creates a new pointer to the same place. A file will only be deleted from disk after all links are removed. Hard links only work on the same file system. A hard link is what you might consider a 'normal' link.

A symbolic link, or soft link, only reference other files by name. They can point to files on different filesystems. Not all systems support symlinks.

### Read and Write

write bytes to file `n, err := file.Write([]byte("hello, world!\n"))`

write string to file `n, err := file.WriteString("Hello, world!\n")`

write at offset `n, err := file.WriteAt([]byte("Hello"), 10)`

read to byte `n, err := file.Read(byteSlice)`

read exactly n bytes `n, err := io.ReadFull(file, byteSlice)`

read at least n bytes `n, err := io.ReadAtLeast(file, byteSlice, minBytes)`

read all bytes of a file `byteSlice, err := ioutil.ReadAll(file)`

read from offset `n, err := file.ReadAt(byteSlice, 10)`



### Work with directories

create a directory `err := os.Mkdir("myDir", 0600)`

recursively create a directory `err := os.MkdirAll("dir/subdir/myDir", 0600)`

delete a directory recursively `err := os.RemoveAll("dir/")`

list directory files `fileInfo, err := ioutil.ReadDir(".")`

### Shortcuts

quick read from file `byteSlice, err := ioutil.ReadFile("test.txt")`

quick write to file `err := ioutil.WriteFile("test.txt", []byte("Hello"), 0666)`

copy file `n, err := io.Copy(newFile, originFile)`

write string to file `io.WriteString(file, "Hello, world")`

### Temporary files and directories

create temp dir `ioutil.TempDir(dir, prefix string) (name string, err error)`

create temp file `ioutil.TempFile(dir, prefix string) (*os.File, err error)`

### References

[Working with Files in Go](#)

[golang os standard library](#)

[golang ioutil standard library](#)

[golang iou standard library](#)

