



**Full credit is given to the above companies including the Operating System (OS) that this PDF file was generated!**

### ***Rocky Enterprise Linux 9.2 Manual Pages on command 'gofmt.1'***

**\$ man gofmt.1**

GOFMT(1)

General Commands Manual

GOFMT(1)

#### NAME

gofmt - format Go programs

#### SYNOPSIS

gofmt [flags] [path ...]

#### DESCRIPTION

Gofmt formats Go programs. It uses tabs for indentation and blanks for alignment. Align?ment assumes that an editor is using a fixed-width font.

Without an explicit path, it processes the standard input. Given a file, it operates on that file; given a directory, it operates on all .go files in that directory, recursively. (Files starting with a period are ignored.) By default, gofmt prints the reformatted sources to standard output.

#### OPTIONS

-d Do not print reformatted sources to standard output. If a file's formatting is different than gofmt's, print diffs to standard output.

-e Print all (including spurious) errors.

-l Do not print reformatted sources to standard output. If a file's formatting is different from gofmt's, print its name to standard output.

-r rule

Apply the rewrite rule to the source before reformatting.

-s Try to simplify code (after applying the rewrite rule, if any).

-w Do not print reformatted sources to standard output. If a file's formatting is different from gofmt's, overwrite it with gofmt's version. If an error occurred

during overwriting, the original file is restored from an automatic backup.

Debugging support:

-cpuprofile filename

Write cpu profile to the specified file.

The rewrite rule specified with the -r flag must be a string of the form:

pattern -> replacement

Both pattern and replacement must be valid Go expressions. In the pattern, single-character lowercase identifiers serve as wildcards matching arbitrary sub-expressions; those expressions will be substituted for the same identifiers in the replacement.

When gofmt reads from standard input, it accepts either a full Go program or a program fragment. A program fragment must be a syntactically valid declaration list, statement list, or expression. When formatting such a fragment, gofmt preserves leading indentation as well as leading and trailing spaces, so that individual sections of a Go program can be formatted by piping them through gofmt.

## EXAMPLES

To check files for unnecessary parentheses:

```
gofmt -r '(a) -> a' -l *.go
```

To remove the parentheses:

```
gofmt -r '(a) -> a' -w *.go
```

To convert the package tree from explicit slice upper bounds to implicit ones:

```
gofmt -r '?[:len(?)] -> ?[?:]' -w $GOROOT/src/pkg
```

The simplify command

When invoked with -s gofmt will make the following source transformations where possible.

An array, slice, or map composite literal of the form:

```
[]T{T{}}, T{()}
```

will be simplified to:

```
[]T{}, {}
```

A slice expression of the form:

```
s[a:len(s)]
```

will be simplified to:

```
s[a:]
```

A range of the form:

```
for x, _ = range v {...}
```

will be simplified to:

```
for x = range v {...}
```

A range of the form:

```
for _ = range v {...}
```

will be simplified to:

```
for range v {...}
```

This may result in changes that are incompatible with earlier versions of Go.

## AUTHOR

This manual page was written by Michael Stapelberg <stapelberg@debian.org> and is maintained by the Debian Go Compiler Team <team+go-compiler@tracker.debian.org> based on the output of 'go doc cmd/gofmt' for the Debian project (and may be used by others).

2021-09-06

GOFMT(1)