


[Discover Packages](#) > [Standard library](#) > [log](#) **log**

package

standard library

Version: [go1.20.1](#) **Latest** | Published: Feb 14, 2023 | License: [BSD-3-Clause](#) | Imports: 7 |

Imported by: 546,539

Details Valid [go.mod](#) file  Redistributable license  Tagged version  Stable version [Learn more](#)**Repository**[cs.opensource.google/go/go](#)**Links** [Report a Vulnerability](#) Documentation <> **Documentation****Overview**

Package log implements a simple logging package. It defines a type, `Logger`, with methods for formatting output. It also has a predefined 'standard' `Logger` accessible through helper functions `Print[f|ln]`, `Fatal[f|ln]`, and `Panic[f|ln]`, which are easier to use than creating a `Logger` manually. That logger writes to standard error and prints the date and time of each logged message. Every log message is output on a separate line: if the message being printed does not end in a newline, the logger will add one. The `Fatal` functions call `os.Exit(1)` after writing the log message. The `Panic` functions call `panic` after writing the log message.

Index[Constants](#)[func Fatal\(v ...any\)](#)[func Fatalf\(format string, v ...any\)](#)[func Fatalln\(v ...any\)](#)[func Flags\(\) int](#)[func Output\(calldepth int, s string\) error](#)[func Panic\(v ...any\)](#)[func Panicf\(format string, v ...any\)](#)[func Panicln\(v ...any\)](#)[func Prefix\(\) string](#)[func Print\(v ...any\)](#)[func Printf\(format string, v ...any\)](#)[func Println\(v ...any\)](#)

```

func SetFlags(flag int)
func SetOutput(w io.Writer)
func SetPrefix(prefix string)
func Writer() io.Writer
type Logger
    func Default() *Logger
    func New(out io.Writer, prefix string, flag int) *Logger
    func (l *Logger) Fatal(v ...any)
    func (l *Logger) Fatalf(format string, v ...any)
    func (l *Logger) Fatalln(v ...any)
    func (l *Logger) Flags() int
    func (l *Logger) Output(calldepth int, s string) error
    func (l *Logger) Panic(v ...any)
    func (l *Logger) Panicf(format string, v ...any)
    func (l *Logger) Panicln(v ...any)
    func (l *Logger) Prefix() string
    func (l *Logger) Print(v ...any)
    func (l *Logger) Printf(format string, v ...any)
    func (l *Logger) Println(v ...any)
    func (l *Logger) SetFlags(flag int)
    func (l *Logger) SetOutput(w io.Writer)
    func (l *Logger) SetPrefix(prefix string)
    func (l *Logger) Writer() io.Writer

```

Examples

[Logger](#)

[Logger.Output](#)

Constants

[View Source](#)

```

const (
    Ldate          = 1 << iota    // the date in the local time zone: 2009/01/23
    Ltime          // the time in the local time zone: 01:23:23
    Lmicroseconds  // microsecond resolution: 01:23:23.123123.  assumes Ltime is set
    Llongfile      // full file name and line number: /a/b/c/d.go:23
    Lshortfile     // final file name element and line number: d.go:23.  c
    LUTC           // if Ldate or Ltime is set, use UTC rather than the local time zone
    Lmsgprefix     // move the "prefix" from the beginning of the line to the end
    LstdFlags      = Ldate | Ltime // initial values for the standard logger
)

```

These flags define which text to prefix to each log entry generated by the `Logger`. Bits are or'ed together to control what's printed. With the exception of the `Lmsgprefix` flag, there is no control over the order they appear (the order listed here) or the format they present (as described in the comments). The prefix is followed by a colon only when `Llongfile` or `Lshortfile` is specified. For example, flags `Ldate | Ltime` (or `LstdFlags`) produce,

```
2009/01/23 01:23:23 message
```

while flags Ldate | Ltime | Lmicroseconds | Llongfile produce,

```
2009/01/23 01:23:23.123123 /a/b/c/d.go:23: message
```

Variables

This section is empty.

Functions

func Fatal

```
func Fatal(v ...any)
```

Fatal is equivalent to Print() followed by a call to os.Exit(1).

func Fatalf

```
func Fatalf(format string, v ...any)
```

Fatalf is equivalent to Printf() followed by a call to os.Exit(1).

func Fatalln

```
func Fatalln(v ...any)
```

Fatalln is equivalent to Println() followed by a call to os.Exit(1).

func Flags

```
func Flags() int
```

Flags returns the output flags for the standard logger. The flag bits are Ldate, Ltime, and so on.

func Output

added in go1.5

```
func Output(calldepth int, s string) error
```

Output writes the output for a logging event. The string s contains the text to print after the prefix specified by the flags of the Logger. A newline is appended if the last character of s is not already a newline. Calldepth is the count of the number of frames to skip when computing the file name and line number if Llongfile or Lshortfile is set; a value of 1 will print the details for the caller of Output.

func Panic

```
func Panic(v ...any)
```

Panic is equivalent to Print() followed by a call to panic().

func Panicf

```
func Panicf(format string, v ...any)
```

Panicf is equivalent to Printf() followed by a call to panic().

func Panicln

```
func Panicln(v ...any)
```

Panicln is equivalent to Println() followed by a call to panic().

func Prefix

```
func Prefix() string
```

Prefix returns the output prefix for the standard logger.

func Print

```
func Print(v ...any)
```

Print calls Output to print to the standard logger. Arguments are handled in the manner of fmt.Print.

func Printf

```
func Printf(format string, v ...any)
```

Printf calls Output to print to the standard logger. Arguments are handled in the manner of fmt.Printf.

func Println

```
func Println(v ...any)
```

Println calls Output to print to the standard logger. Arguments are handled in the manner of fmt.Println.

func SetFlags

```
func SetFlags(flag int)
```

SetFlags sets the output flags for the standard logger. The flag bits are Ldate, Ltime, and so on.

func SetOutput

```
func SetOutput(w io.Writer)
```

SetOutput sets the output destination for the standard logger.

func SetPrefix

```
func SetPrefix(prefix string)
```

SetPrefix sets the output prefix for the standard logger.

func Writer

added in go1.13

```
func Writer() io.Writer
```

Writer returns the output destination for the standard logger.

Types

type Logger

```
type Logger struct {  
    // contains filtered or unexported fields  
}
```

A Logger represents an active logging object that generates lines of output to an `io.Writer`. Each logging operation makes a single call to the Writer's `Write` method. A Logger can be used simultaneously from multiple goroutines; it guarantees to serialize access to the Writer.

► [Example](#)

func Default

added in go1.16

```
func Default() *Logger
```

Default returns the standard logger used by the package-level output functions.

func New

```
func New(out io.Writer, prefix string, flag int) *Logger
```

New creates a new Logger. The `out` variable sets the destination to which log data will be written. The `prefix` appears at the beginning of each generated log line, or after the log header if the `Lmsgprefix` flag is provided. The `flag` argument defines the logging properties.

func (*Logger) Fatal

```
func (l *Logger) Fatal(v ...any)
```

Fatal is equivalent to `l.Print()` followed by a call to `os.Exit(1)`.

func (*Logger) **Fatalf**

```
func (l *Logger) Fatalf(format string, v ...any)
```

Fatalf is equivalent to `l.Printf()` followed by a call to `os.Exit(1)`.

func (*Logger) **Fatalln**

```
func (l *Logger) Fatalln(v ...any)
```

Fatalln is equivalent to `l.Println()` followed by a call to `os.Exit(1)`.

func (*Logger) **Flags**

```
func (l *Logger) Flags() int
```

Flags returns the output flags for the logger. The flag bits are `Ldate`, `Ltime`, and so on.

func (*Logger) **Output**

```
func (l *Logger) Output(calldepth int, s string) error
```

Output writes the output for a logging event. The string `s` contains the text to print after the prefix specified by the flags of the Logger. A newline is appended if the last character of `s` is not already a newline. `Calldepth` is used to recover the PC and is provided for generality, although at the moment on all pre-defined paths it will be 2.

► [Example](#)

func (*Logger) **Panic**

```
func (l *Logger) Panic(v ...any)
```

Panic is equivalent to `l.Print()` followed by a call to `panic()`.

func (*Logger) **Panicf**

```
func (l *Logger) Panicf(format string, v ...any)
```

Panicf is equivalent to `l.Printf()` followed by a call to `panic()`.

func (*Logger) **Panicln**

```
func (l *Logger) Panicln(v ...any)
```

Panicln is equivalent to l.Println() followed by a call to panic().

func (*Logger) Prefix

```
func (l *Logger) Prefix() string
```

Prefix returns the output prefix for the logger.

func (*Logger) Print

```
func (l *Logger) Print(v ...any)
```

Print calls l.Output to print to the logger. Arguments are handled in the manner of fmt.Print.

func (*Logger) Printf

```
func (l *Logger) Printf(format string, v ...any)
```

Printf calls l.Output to print to the logger. Arguments are handled in the manner of fmt.Printf.

func (*Logger) Println

```
func (l *Logger) Println(v ...any)
```

Println calls l.Output to print to the logger. Arguments are handled in the manner of fmt.Println.

func (*Logger) SetFlags

```
func (l *Logger) SetFlags(flag int)
```

SetFlags sets the output flags for the logger. The flag bits are Ldate, Ltime, and so on.

func (*Logger) SetOutput

added in go1.5

```
func (l *Logger) SetOutput(w io.Writer)
```

SetOutput sets the output destination for the logger.

func (*Logger) SetPrefix

```
func (l *Logger) SetPrefix(prefix string)
```

SetPrefix sets the output prefix for the logger.

func (*Logger) Writer

added in go1.12

```
func (l *Logger) Writer() io.Writer
```

Writer returns the output destination for the logger.

Source Files

[View all](#) 

[log.go](#)

Directories

[syslog](#)

Package syslog provides a simple interface to the system log service.

Why Go

[Use Cases](#)

[Case Studies](#)

Get Started

[Playground](#)

[Tour](#)

[Stack Overflow](#)

[Help](#)

Packages

[Standard Library](#)

[About Go Packages](#)

About

[Download](#)

[Blog](#)

[Issue Tracker](#)

[Release Notes](#)

[Brand Guidelines](#)

[Code of Conduct](#)

Connect

[Twitter](#)

[GitHub](#)

[Slack](#)

[r/golang](#)

[Meetup](#)

[Golang Weekly](#)

[Copyright](#)

[Terms of Service](#)

[Privacy Policy](#)

[Report an Issue](#)



Google