zlib

package standard library

Version: go1.20.1 Latest | Published: Feb 14, 2023 | License: BSD-3-Clause | Imports: 8 |
Imported by: 7,260

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 zlib implements reading and writing of zlib format compressed data, as specified in RFC 1950.

The implementation provides filters that uncompress during reading and compress during writing. For example, to write compressed data to a buffer:

```
var b bytes.Buffer
w := zlib.NewWriter(&b)
w.Write([]byte("hello, world\n"))
w.Close()
```

and to read that data back:

```
r, err := zlib.NewReader(&b)
io.Copy(os.Stdout, r)
r.Close()
```

Index

Examples

Constants

View Source

```
const (
	NoCompression      = flate.NoCompression
	BestSpeed          = flate.BestSpeed
	BestCompression    = flate.BestCompression
	DefaultCompression = flate.DefaultCompression
	HuffmanOnly        = flate.HuffmanOnly
)
```

These constants are copied from the flate package, so that code that imports "compress/zlib" does not also have to import "compress/flate".

Variables

View Source

```
var (
	// ErrChecksum is returned when reading ZLIB data that has an invalid checksum.
	ErrChecksum = errors.New("zlib: invalid checksum")
	// ErrDictionary is returned when reading ZLIB data that has an invalid dictionary.
	ErrDictionary = errors.New("zlib: invalid dictionary")
	// ErrHeader is returned when reading ZLIB data that has an invalid header.
	ErrHeader = errors.New("zlib: invalid header")
)
```

Functions

func NewReader

```
func NewReader(r io.Reader) (io.ReadCloser, error)
```

NewReader creates a new ReadCloser. Reads from the returned ReadCloser read and decompress data from r. If r does not implement io.ByteReader, the decompressor may read more data than necessary from r. It is the caller's responsibility to call Close on the ReadCloser when done.

The ReadCloser returned by NewReader also implements Resetter.

▶ Example

func NewReaderDict

```
func NewReaderDict(r io.Reader, dict []byte) (io.ReadCloser, error)
```

NewReaderDict is like NewReader but uses a preset dictionary. NewReaderDict ignores the dictionary if the compressed data does not refer to it. If the compressed data refers to a different dictionary, NewReaderDict returns ErrDictionary.

The ReadCloser returned by NewReaderDict also implements Resetter.

Types

type Resetter

added in go1.4

```
type Resetter interface {
    // Reset discards any buffered data and resets the Resetter as if it was
    // newly initialized with the given reader.
    Reset(r io.Reader, dict []byte) error
}
```

Resetter resets a ReadCloser returned by NewReader or NewReaderDict to switch to a new underlying Reader. This permits reusing a ReadCloser instead of allocating a new one.

type Writer

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

A Writer takes data written to it and writes the compressed form of that data to an underlying writer (see NewWriter).

func NewWriter

```
func NewWriter(w io.Writer) *Writer
```

NewWriter creates a new Writer. Writes to the returned Writer are compressed and written to w.

It is the caller's responsibility to call Close on the Writer when done. Writes may be buffered and not flushed until Close.

▶ Example

func NewWriterLevel

```
func NewWriterLevel(w io.Writer, level int) (*Writer, error)
```

NewWriterLevel is like NewWriter but specifies the compression level instead of assuming DefaultCompression.

The compression level can be DefaultCompression, NoCompression, HuffmanOnly or any integer value between BestSpeed and BestCompression inclusive. The error returned will be nil if the level is valid.

func NewWriterLevelDict

```
func NewWriterLevelDict(w io.Writer, level int, dict []byte) (*Writer, error)
```

NewWriterLevelDict is like NewWriterLevel but specifies a dictionary to compress with.

The dictionary may be nil. If not, its contents should not be modified until the Writer is closed.

func (*Writer) Close

```
func (z *Writer) Close() error
```

Close closes the Writer, flushing any unwritten data to the underlying io.Writer, but does not close the underlying io.Writer.

func (*Writer) Flush

```
func (z *Writer) Flush() error
```

Flush flushes the Writer to its underlying io.Writer.

func (*Writer) Reset

added in go1.2

```
func (z *Writer) Reset(w io.Writer)
```

Reset clears the state of the Writer z such that it is equivalent to its initial state from NewWriterLevel or NewWriterLevelDict, but instead writing to w.

func (*Writer) Write

```
func (z *Writer) Write(p []byte) (n int, err error)
```

Write writes a compressed form of p to the underlying io.Writer. The compressed bytes are not necessarily flushed until the Writer is closed or explicitly flushed.

Source Files

View all

reader.go writer.go

Google