tabwriter

package standard library

Version: go1.20.1 Latest | Published: Feb 14, 2023 | License: BSD-3-Clause | Imports: 2 |
Imported by: 19,314

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 tabwriter implements a write filter (tabwriter.Writer) that translates tabbed columns in input into properly aligned text.

The package is using the Elastic Tabstops algorithm described at http://nickgravgaard.com/elastictabstops/index.html.

The text/tabwriter package is frozen and is not accepting new features.

▶ Example (Elastic)

▶ Example (TrailingTab)

Index

Constants
type Writer
    func NewWriter(output io.Writer, minwidth, tabwidth, padding int, padchar byte, flags uint) *Writer
    func (b *Writer) Flush() error
    func (b *Writer) Init(output io.Writer, minwidth, tabwidth, padding int, padchar byte, flags uint) *Writer
    func (b *Writer) Write(buf []byte) (n int, err error)

Examples

Package (Elastic)
Package (TrailingTab)
Writer.Init

Constants

View Source

```
const (
	// Ignore html tags and treat entities (starting with '&'
	// and ending in ';') as single characters (width = 1).
	FilterHTML uint = 1 << iota

	// Strip Escape characters bracketing escaped text segments
	// instead of passing them through unchanged with the text.
	StripEscape

	// Force right-alignment of cell content.
	// Default is left-alignment.
	AlignRight

	// Handle empty columns as if they were not present in
	// the input in the first place.
	DiscardEmptyColumns

	// Always use tabs for indentation columns (i.e., padding of
	// leading empty cells on the left) independent of padchar.
	TabIndent

	// Print a vertical bar ('|') between columns (after formatting).
	// Discarded columns appear as zero-width columns ("||").
	Debug
)
```

Formatting can be controlled with these flags.

View Source

```
const Escape = '\xff'
```

To escape a text segment, bracket it with Escape characters. For instance, the tab in this string "Ignore this tab: \xff\t\xff" does not terminate a cell and constitutes a single character of width one for formatting purposes.

The value 0xff was chosen because it cannot appear in a valid UTF-8 sequence.

Variables

This section is empty.

Functions

This section is empty.

Types

type Writer

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

A Writer is a filter that inserts padding around tab-delimited columns in its input to align them in the output.

The Writer treats incoming bytes as UTF-8-encoded text consisting of cells terminated by horizontal ('\t') or vertical ('\v') tabs, and newline ('\n') or formfeed ('\f') characters; both newline and formfeed act as line breaks.

Tab-terminated cells in contiguous lines constitute a column. The Writer inserts padding as needed to make all cells in a column have the same width, effectively aligning the columns. It assumes that all characters have the same width, except for tabs for which a tabwidth must be specified. Column cells must be tab-terminated, not tab-separated: non-tab terminated trailing text at the end of a line forms a cell but that cell is not part of an aligned column. For instance, in this example (where | stands for a horizontal tab):

```
aaaa|bbb|d
aa  |b  |dd
a   |
aa  |cccc|eee
```

the b and c are in distinct columns (the b column is not contiguous all the way). The d and e are not in a column at all (there's no terminating tab, nor would the column be contiguous).

The Writer assumes that all Unicode code points have the same width; this may not be true in some fonts or if the string contains combining characters.

If DiscardEmptyColumns is set, empty columns that are terminated entirely by vertical (or "soft") tabs are discarded. Columns terminated by horizontal (or "hard") tabs are not affected by this flag.

If a Writer is configured to filter HTML, HTML tags and entities are passed through. The widths of tags and entities are assumed to be zero (tags) and one (entities) for formatting purposes.

A segment of text may be escaped by bracketing it with Escape characters. The tabwriter passes escaped text segments through unchanged. In particular, it does not interpret any tabs or line breaks within the segment. If the StripEscape flag is set, the Escape characters are stripped from the output; otherwise they are passed through as well. For the purpose of formatting, the width of the escaped text is always computed excluding the Escape characters.

The formfeed character acts like a newline but it also terminates all columns in the current line (effectively calling Flush). Tab- terminated cells in the next line start new columns. Unless found inside an HTML tag or inside an escaped text segment, formfeed characters appear as newlines in the output.

The Writer must buffer input internally, because proper spacing of one line may depend on the cells in future lines. Clients must call Flush when done calling Write.

func NewWriter

```
func NewWriter(output io.Writer, minwidth, tabwidth, padding int, padchar byte, flags
uint) *Writer
```

NewWriter allocates and initializes a new tabwriter.Writer. The parameters are the same as for the Init function.

func (*Writer) Flush

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

Flush should be called after the last call to Write to ensure that any data buffered in the Writer is written to output. Any incomplete escape sequence at the end is considered complete for formatting purposes.

func (*Writer) Init

```
func (b *Writer) Init(output io.Writer, minwidth, tabwidth, padding int, padchar byte,
flags uint) *Writer
```

A Writer must be initialized with a call to Init. The first parameter (output) specifies the filter output. The remaining parameters control the formatting:

```
minwidth    minimal cell width including any padding
tabwidth    width of tab characters (equivalent number of spaces)
padding     padding added to a cell before computing its width
padchar     ASCII char used for padding
        if padchar == '\t', the Writer will assume that the
        width of a '\t' in the formatted output is tabwidth,
        and cells are left-aligned independent of align_left
        (for correct-looking results, tabwidth must correspond
        to the tab width in the viewer displaying the result)
flags       formatting control
```

▶ Example

func (*Writer) Write

```
func (b *Writer) Write(buf []byte) (n int, err error)
```

Write writes buf to the writer b. The only errors returned are ones encountered while writing to the underlying output stream.

Source Files

View all

tabwriter.go