TRIVIAL UTF-8 MANUAL

Contents

1 Introduction 1

2 Links and Systems 1

3 Reference 2

4 Indices 2
- 4.1 Function and Macro Index . . . 3
- 4.2 Type Index . . . 3
- 4.3 Misc Index . . . 3

[in package TRIVIAL-UTF-8]

1 Introduction

Trivial UTF-8 is a small library for doing UTF-8-based in- and output on a Lisp implementation that already supports Unicode - meaning `char-code` and `code-char` deal with Unicode character codes.

The rationale for the existence of this library is that while Unicode-enabled implementations usually do provide some kind of interface to dealing with character encodings, these are typically not terribly flexible or uniform.

The Babel library solves a similar problem while understanding more encodings. Trivial UTF-8 was written before Babel existed, but for new projects you might be better off going with Babel. The one plus that Trivial UTF-8 has is that it doesn't depend on any other libraries.

2 Links and Systems

Here is the official repository and the HTML documentation for the latest version.

- **[system]** **"trivial-utf-8"**
  - *Description:* A small library for doing UTF-8-based input and output.
  - *Licence:* ZLIB
  - *Author:* Marijn Haverbeke marijnh@gmail.com
  - *Maintainer:* Gábor Melis mega@retes.hu

- *Homepage:* https://common-lisp.net/project/trivial-utf-8/
- *Bug tracker:* https://gitlab.common-lisp.net/trivial-utf-8/trivial-utf-8/-/issues
- *Source control:* GIT
- *Depends on:* mgl-pax-bootstrap

3 Reference

- **[function]** **utf-8-byte-length** *string*

  Calculate the amount of bytes needed to encode `string`.

- **[function]** **string-to-utf-8-bytes** *string &key null-terminate*

  Convert `string` into an array of unsigned bytes containing its UTF-8 representation. If `null-terminate`, add an extra 0 byte at the end.

- **[function]** **utf-8-group-size** *byte*

  Determine the amount of bytes that are part of the character whose encoding starts with `byte`. May signal `utf-8-decoding-error`.

- **[function]** **utf-8-bytes-to-string** *bytes &key (start 0) (end (length bytes))*

  Convert the `start`, `end` subsequence of the array of `bytes` containing UTF-8 encoded characters to a `string`. The element type of `bytes` may be anything as long as it can be `coerce`d into an `(unsigned-bytes 8)` array. May signal `utf-8-decoding-error`.

- **[function]** **read-utf-8-string** *input &key null-terminated stop-at-eof (char-length -1) (byte-length -1)*

  Read UTF-8 encoded data from `input`, a byte stream, and construct a string with the characters found. When `null-terminated` is given, stop reading at a null character. If `stop-at-eof`, then stop at `end-of-file` without raising an error. The `char-length` and `byte-length` parameters can be used to specify the max amount of characters or bytes to read, where -1 means no limit. May signal `utf-8-decoding-error`.

- **[function]** **write-utf-8-bytes** *string byte-stream &key null-terminate*

  Write `string` to `byte-stream`, encoding it as UTF-8. If `null-terminate`, write an extra 0 byte at the end.

- **[condition]** **utf-8-decoding-error** *simple-error*

4 Indices

Referrer definition type abbreviations:

- *f*: for definitions in the function namespace (macros, compiler macros and also methods)
- *t*: `DEFTYPE`s, classes, conditions, structs

- *d*: documentation sections and glossary terms
- *l*: definitions of definition types
- *s*: ASDF systems
- *p*: packages
- *n*: named readtables
- *v*: special variables and constants
- *r*: restarts
- *?*: other

4.1 Function and Macro Index

`read-utf-8-string` 2 *(fn)*
`string-to-utf-8-bytes` 2 *(fn)*
`utf-8-byte-length` 2 *(fn)*
`utf-8-bytes-to-string` 2 *(fn)*
`utf-8-group-size` 2 *(fn)*
`write-utf-8-bytes` 2 *(fn)*

4.2 Type Index

`utf-8-decoding-error` 2 *(condition)* ↩ *f*: `read-utf-8-string` 2, `utf-8-bytes-to-string` 2, `utf-8-group-size` 2

4.3 Misc Index

`trivial-utf-8` 1 *(asdf:system)*