

TakoStrLib

Generated by Doxygen 1.8.17

1 Class Index	1
1.1 Class List	1
2 File Index	3
2.1 File List	3
3 Class Documentation	5
3.1 CharBuf Struct Reference	5
3.1.1 Detailed Description	5
3.1.2 Member Data Documentation	5
3.1.2.1 buf	5
3.1.2.2 size	6
3.2 StrArray Struct Reference	6
3.2.1 Detailed Description	6
3.2.2 Member Data Documentation	6
3.2.2.1 lines	6
3.2.2.2 size	6
4 File Documentation	7
4.1 CharBuf.hh File Reference	7
4.1.1 Typedef Documentation	8
4.1.1.1 String	8
4.1.2 Function Documentation	8
4.1.2.1 cb_init() [1/2]	8
4.1.2.2 cb_init() [2/2]	8
4.1.2.3 cb_destr()	9
4.1.2.4 sa_init()	9
4.1.2.5 sa_destr()	9
4.1.2.6 sa_print()	10
4.2 filesys.hh File Reference	10
4.2.1 Function Documentation	10
4.2.1.1 tfl_fsize()	10
4.2.1.2 file_to_buf()	11
4.3 tsl.hh File Reference	11
4.3.1 Function Documentation	12
4.3.1.1 tsl_fputs()	12
4.3.1.2 tsl_puts()	12
4.3.1.3 tsl_strchr()	13
4.3.1.4 tsl_const_strchr()	13
4.3.1.5 tsl_strlen()	14
4.3.1.6 tsl_strcpy()	14
4.3.1.7 tsl_strncpy()	14
4.3.1.8 tsl_strcat()	15

4.3.1.9 <code>tsl_strncat()</code>	15
4.3.1.10 <code>tsl_fgets()</code>	16
4.3.1.11 <code>tsl_strdup()</code>	16
4.3.1.12 <code>tsl_test()</code>	16
4.3.1.13 <code>tsl_split_lines()</code>	17
4.3.1.14 <code>tsl_cb_cmp()</code>	17
4.3.1.15 <code>tsl_cb_back_cmp()</code>	17

Index	19
--------------	-----------

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

CharBuf	Struct which keeps dynamic char array and it's size	5
StrArray	Struct to keep all strings	6

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

CharBuf.hh	7
filesys.hh	10
tsl.hh	11

Chapter 3

Class Documentation

3.1 CharBuf Struct Reference

Struct which keeps dynamic char array and it's size.

```
#include <CharBuf.hh>
```

Public Attributes

- char * [buf](#)
- size_t [size](#)

3.1.1 Detailed Description

Struct which keeps dynamic char array and it's size.

Definition at line 10 of file CharBuf.hh.

3.1.2 Member Data Documentation

3.1.2.1 buf

```
char* CharBuf::buf
```

Definition at line 12 of file CharBuf.hh.

3.1.2.2 size

```
size_t CharBuf::size
```

Definition at line 13 of file CharBuf.hh.

The documentation for this struct was generated from the following file:

- [CharBuf.hh](#)

3.2 StrArray Struct Reference

Struct to keep all strings.

```
#include <CharBuf.hh>
```

Public Attributes

- [String * lines](#)
- [size_t size](#)

3.2.1 Detailed Description

Struct to keep all strings.

Definition at line 24 of file CharBuf.hh.

3.2.2 Member Data Documentation

3.2.2.1 lines

```
String* StrArray::lines
```

Definition at line 26 of file CharBuf.hh.

3.2.2.2 size

```
size_t StrArray::size
```

Definition at line 27 of file CharBuf.hh.

The documentation for this struct was generated from the following file:

- [CharBuf.hh](#)

Chapter 4

File Documentation

4.1 CharBuf.hh File Reference

```
#include <stdio.h>
#include <stdlib.h>
```

Classes

- struct [CharBuf](#)
Struct which keeps dynamic char array and it's size.
- struct [StrArray](#)
Struct to keep all strings.

Typedefs

- typedef [CharBuf](#) String
Alias for [CharBuf](#) struct.

Functions

- [CharBuf * cb_init](#) ([CharBuf](#) *cb, size_t elem_num)
CharBuf constructor.
- [CharBuf * cb_init](#) ([CharBuf](#) *cb_new, const [CharBuf](#) *cb_old)
CharBuf copy constructor.
- [CharBuf * cb_destr](#) ([CharBuf](#) *cb)
CharBuf destructor.
- [StrArray * sa_init](#) ([StrArray](#) *sa, size_t elnum)
StrArray constructor.
- [StrArray * sa_destr](#) ([StrArray](#) *sa)
StrArray destructor.
- void [sa_print](#) ([StrArray](#) sa, FILE *fp=stdout)
prints string array

4.1.1 Typedef Documentation

4.1.1.1 String

```
typedef CharBuf String
```

Alias for [CharBuf](#) struct.

Definition at line 19 of file CharBuf.hh.

4.1.2 Function Documentation

4.1.2.1 `cb_init()` [1/2]

```
CharBuf* cb_init (  
    CharBuf * cb,  
    size_t elem_num )
```

[CharBuf](#) constructor.

Parameters

out	<i>cb</i>	CharBuf object
in	<i>elem_num</i>	number of elements in buffer

Returns

pointer to input [CharBuf](#) object

4.1.2.2 `cb_init()` [2/2]

```
CharBuf* cb_init (  
    CharBuf * cb_new,  
    const CharBuf * cb_old )
```

[CharBuf](#) copy constructor.

Parameters

out	<i>cb_new</i>	pointer to uninitialized CharBuf
in	<i>cb_old</i>	old CharBuf

Returns

pointer to input [CharBuf](#) object

4.1.2.3 `cb_destr()`

```
CharBuf* cb_destr (
    CharBuf * cb )
```

[CharBuf](#) destructor.

Parameters

out	<i>cb</i>	CharBuf object
-----	-----------	--------------------------------

Returns

pointer to input [CharBuf](#) object

4.1.2.4 `sa_init()`

```
StrArray* sa_init (
    StrArray * sa,
    size_t elnum )
```

[StrArray](#) constructor.

Parameters

out	<i>sa</i>	StrArray object
in	<i>elnum</i>	number of lines

Returns

pointer to input object

4.1.2.5 `sa_destr()`

```
StrArray* sa_destr (
    StrArray * sa )
```

[StrArray](#) destructor.

Parameters

out	sa	StrArray object
-----	----	-----------------

Returns

pointer to input object

4.1.2.6 sa_print()

```
void sa_print (
    StrArray sa,
    FILE * fp = stdout )
```

prints string array

Parameters

in	sa	
----	----	--

4.2 filesystem.hh File Reference

```
#include <stdio.h>
#include "CharBuf.hh"
```

Functions

- ssize_t [tfl_fsize](#) (FILE *fp)
caclulates file's size
- int [file_to_buf](#) (const char *fname, [CharBuf](#) *raw_data)
filling buffer with data from file

4.2.1 Function Documentation**4.2.1.1 tfl_fsize()**

```
ssize_t tfl_fsize (
    FILE * fp )
```

caclulates file's size

Parameters

in	<i>fp</i>	pointer to file
----	-----------	-----------------

Returns

size of file in case of success, negative value otherwise.

4.2.1.2 file_to_buf()

```
int file_to_buf (
    const char * fname,
    CharBuf * raw_data )
```

filling buffer with data from file

Parameters

in	<i>fname</i>	filename
out	<i>raw_data</i>	buffer to fill

Returns

0 on success

4.3 tsl.hh File Reference

```
#include "CharBuf.hh"
```

Functions

- int [tsl_fputs](#) (const char *str, FILE *stream)
writes the string str to stream, without terminating null byte ('\0').
- int [tsl_puts](#) (const char *str)
writes the string str and a trailing newline to stdout.
- char * [tsl_strchr](#) (char *str, int ch)
returns a pointer to the first occurrence of the character c in the string str.
- const char * [tsl_const_strchr](#) (const char *str, int ch)
returns a pointer to the first occurrence of the character c in the string s.
- size_t [tsl_strlen](#) (const char *str)
calculates the length of the string pointed to by str, excluding the terminating null byte ('\0').
- char * [tsl_strcpy](#) (char *dst, const char *src)
copies the string pointed to by src, including the terminating null byte ('\0'), to the buffer pointed to by dest.
- char * [tsl_strncpy](#) (char *dst, const char *src, size_t n)

- similar to `tsl_strcpy`, except that at most n bytes of `src` are copied*

 - `char * tsl_strcat (char *dst, const char *src)`
appends the `src` string to the `dest` string, overwriting the terminating null byte ('`\0`') at the end of `dest`, and then adds a terminating null byte.
 - `char * tsl_strncat (char *dst, const char *src, size_t n)`
is similar to `tsl_strcat`, except that it will use at most n bytes from `src` and `src` does not need to be null-terminated if it contains n or more bytes.
 - `char * tsl_fgets (char *str, int size, FILE *stream)`
reads in at most one less than `size` characters from `stream` and stores them into the buffer pointed to by `str`.
 - `char * tsl_strdup (const char *str)`
returns a pointer to a new string which is a duplicate of the string `str`.
 - `int tsl_test ()`
unit test for `tsl` functions
 - `StrArray tsl_split_lines (CharBuf raw)`
split input buffer to lines, allocates memory in `StrArray::lines`.
 - `int tsl_cb_cmp (const void *lhs, const void *rhs)`
comparator for `CharBuf` strings
 - `int tsl_cb_back_cmp (const void *lhs, const void *rhs)`
backward comparator for `CharBuf` strings

4.3.1 Function Documentation

4.3.1.1 `tsl_fputs()`

```
int tsl_fputs (
    const char * str,
    FILE * stream )
```

writes the string `str` to `stream`, without terminating null byte ('`\0`').

Parameters

in	<code>str</code>	
in	<code>stream</code>	

Returns

nonnegative number on success, or EOF on error.

4.3.1.2 `tsl_puts()`

```
int tsl_puts (
    const char * str )
```

writes the string `str` and a trailing newline to `stdout`.

Parameters

in	<i>str</i>	
----	------------	--

Returns

a nonnegative number on success, or EOF on error.

4.3.1.3 tsl_strchr()

```
char* tsl_strchr (  
    char * str,  
    int ch )
```

returns a pointer to the first occurrence of the character *c* in the string *str*.

Parameters

in	<i>str</i>	
in	<i>ch</i>	

Returns

pointer to the matched character or NULL if the character is not found.

4.3.1.4 tsl_const_strchr()

```
const char* tsl_const_strchr (  
    const char * str,  
    int ch )
```

returns a pointer to the first occurrence of the character *c* in the string *s*.

Parameters

in	<i>str</i>	
in	<i>ch</i>	

Returns

pointer to the matched character or NULL if the character is not found.

4.3.1.5 `tsl_strlen()`

```
size_t tsl_strlen (  
    const char * str )
```

calculates the length of the string pointed to by *str*, excluding the terminating null byte ('\0').

Parameters

in	<i>str</i>	
----	------------	--

Returns

the number of bytes in the string pointed to by *s*.

4.3.1.6 `tsl_strcpy()`

```
char* tsl_strcpy (  
    char * dst,  
    const char * src )
```

copies the string pointed to by *src*, including the terminating null byte ('\0'), to the buffer pointed to by *dest*.

The strings may not overlap, and the destination string *dest* must be large enough to receive the copy.

Parameters

out	<i>dst</i>	
in	<i>src</i>	

Returns

pointer to the destination string.

4.3.1.7 `tsl_strncpy()`

```
char* tsl_strncpy (  
    char * dst,  
    const char * src,  
    size_t n )
```

similar to `tsl_strcpy`, except that at most *n* bytes of *src* are copied

Parameters

out	<i>dst</i>	
in	<i>src</i>	
in	<i>n</i>	

Returns

pointer to the destination string.

4.3.1.8 tsl_strcat()

```
char* tsl_strcat (
    char * dst,
    const char * src )
```

appends the *src* string to the *dest* string, overwriting the terminating null byte ('\0') at the end of *dest*, and then adds a terminating null byte.

Parameters

out	<i>dst</i>	
in	<i>src</i>	

Returns

pointer to the resulting string.

4.3.1.9 tsl_strncat()

```
char* tsl_strncat (
    char * dst,
    const char * src,
    size_t n )
```

is similar to `tsl_strcat`, except that it will use at most *n* bytes from *src* and *src* does not need to be null-terminated if it contains *n* or more bytes.

Parameters

out	<i>dst</i>	
in	<i>src</i>	
in	<i>n</i>	

Returns

pointer to the resulting string.

4.3.1.10 `tsl_fgets()`

```
char* tsl_fgets (
    char * str,
    int size,
    FILE * stream )
```

reads in at most one less than *size* characters from *stream* and stores them into the buffer pointed to by *str*.

Reading stops after an EOF or a newline. If a newline is read, it is stored into the buffer. A terminating null byte ('\0') is stored after the last character in the buffer.

Parameters

out	<i>str</i>	
in	<i>size</i>	
in	<i>stream</i>	

Returns

str on success, and `nullptr` on error or when end of file occurs while no characters have been read.

4.3.1.11 `tsl_strdup()`

```
char* tsl_strdup (
    const char * str )
```

returns a pointer to a new string which is a duplicate of the string *str*.

Memory for the new string is obtained with `malloc`, and can be freed with `free`.

Parameters

in	<i>str</i>	
----	------------	--

Returns

pointer to the duplicated string or `nullptr` in case of failure.

4.3.1.12 `tsl_test()`

```
int tsl_test ( )
```

unit test for `tsl` functions

Returns

0 on success

4.3.1.13 `tsl_split_lines()`

```
StrArray tsl_split_lines (
    CharBuf raw )
```

split input buffer to lines, allocates memory in [StrArray::lines](#).

Do not forget to free

Parameters

in	raw	input char buffer
----	-----	-------------------

Returns

[StrArray](#)

4.3.1.14 `tsl_cb_cmp()`

```
int tsl_cb_cmp (
    const void * lhs,
    const void * rhs )
```

comparator for [CharBuf](#) strings

Parameters

in	lhs	
in	rhs	

Returns

== 0 if elements are equal
 > 0 if lhs > rhs
 < 0 if lhs < rhs

4.3.1.15 `tsl_cb_back_cmp()`

```
int tsl_cb_back_cmp (
    const void * lhs,
    const void * rhs )
```

backward comparator for [CharBuf](#) strings

Parameters

in	<i>lhs</i>	
in	<i>rhs</i>	

Returns

== 0 if elements are equal

> 0 if lhs > rhs

< 0 if lhs < rhs

Index

- buf
 - CharBuf, 5
- cb_destr
 - CharBuf.hh, 9
- cb_init
 - CharBuf.hh, 8
- CharBuf, 5
 - buf, 5
 - size, 5
- CharBuf.hh, 7
 - cb_destr, 9
 - cb_init, 8
 - sa_destr, 9
 - sa_init, 9
 - sa_print, 10
 - String, 8
- file_to_buf
 - filesys.hh, 11
- filesys.hh, 10
 - file_to_buf, 11
 - tfl_fsize, 10
- lines
 - StrArray, 6
- sa_destr
 - CharBuf.hh, 9
- sa_init
 - CharBuf.hh, 9
- sa_print
 - CharBuf.hh, 10
- size
 - CharBuf, 5
 - StrArray, 6
- StrArray, 6
 - lines, 6
 - size, 6
- String
 - CharBuf.hh, 8
- tfl_fsize
 - filesys.hh, 10
- tsl.hh, 11
 - tsl_cb_back_cmp, 17
 - tsl_cb_cmp, 17
 - tsl_const_strchr, 13
 - tsl_fgets, 15
 - tsl_fputs, 12
 - tsl_puts, 12
 - tsl_split_lines, 16
 - tsl_strcat, 15
 - tsl_strchr, 13
 - tsl_strcpy, 14
 - tsl_strdup, 16
 - tsl_strlen, 13
 - tsl_strncat, 15
 - tsl_strncpy, 14
 - tsl_test, 16
- tsl_cb_back_cmp
 - tsl.hh, 17
- tsl_cb_cmp
 - tsl.hh, 17
- tsl_const_strchr
 - tsl.hh, 13
- tsl_fgets
 - tsl.hh, 15
- tsl_fputs
 - tsl.hh, 12
- tsl_puts
 - tsl.hh, 12
- tsl_split_lines
 - tsl.hh, 16
- tsl_strcat
 - tsl.hh, 15
- tsl_strchr
 - tsl.hh, 13
- tsl_strcpy
 - tsl.hh, 14
- tsl_strdup
 - tsl.hh, 16
- tsl_strlen
 - tsl.hh, 13
- tsl_strncat
 - tsl.hh, 15
- tsl_strncpy
 - tsl.hh, 14
- tsl_test
 - tsl.hh, 16