

INF558

Generated by Doxygen 1.8.17



<b>1 Data Structure Index</b>	<b>1</b>
1.1 Data Structures	1
<b>2 File Index</b>	<b>3</b>
2.1 File List	3
<b>3 Data Structure Documentation</b>	<b>5</b>
3.1 buffer_t Struct Reference	5
3.1.1 Field Documentation	5
3.1.1.1 length	5
3.1.1.2 size	5
3.1.1.3 tab	5
<b>4 File Documentation</b>	<b>7</b>
4.1 base64.c File Reference	7
4.1.1 Detailed Description	8
4.1.2 Typedef Documentation	8
4.1.2.1 uchar	8
4.1.3 Function Documentation	8
4.1.3.1 CodeBase64()	8
4.1.3.2 DecodeBase64()	8
4.2 base64.h File Reference	9
4.2.1 Typedef Documentation	9
4.2.1.1 uchar	9
4.2.2 Function Documentation	9
4.2.2.1 CodeBase64()	9
4.2.2.2 DecodeBase64()	10
4.3 buffer.c File Reference	10
4.3.1 Detailed Description	11
4.3.2 Macro Definition Documentation	11
4.3.2.1 DEBUG	11
4.3.3 Function Documentation	11
4.3.3.1 buffer_append()	12
4.3.3.2 buffer_append_uchar()	12
4.3.3.3 buffer_clear()	12
4.3.3.4 buffer_clone()	12
4.3.3.5 buffer_equality()	12
4.3.3.6 buffer_from_base64()	13
4.3.3.7 buffer_from_file()	13
4.3.3.8 buffer_from_string()	13
4.3.3.9 buffer_init()	13
4.3.3.10 buffer_print()	13
4.3.3.11 buffer_print_int()	14

4.3.3.12 <a href="#">buffer_random()</a>	14
4.3.3.13 <a href="#">buffer_reset()</a>	14
4.3.3.14 <a href="#">buffer_resize()</a>	14
4.3.3.15 <a href="#">buffer_to_base64()</a>	14
4.3.3.16 <a href="#">string_from_buffer()</a>	15
4.4 <a href="#">buffer.h</a> File Reference	15
4.4.1 Typedef Documentation	16
4.4.1.1 <a href="#">uchar</a>	16
4.4.2 Function Documentation	16
4.4.2.1 <a href="#">buffer_append()</a>	16
4.4.2.2 <a href="#">buffer_append_uchar()</a>	16
4.4.2.3 <a href="#">buffer_clear()</a>	17
4.4.2.4 <a href="#">buffer_clone()</a>	17
4.4.2.5 <a href="#">buffer_equality()</a>	17
4.4.2.6 <a href="#">buffer_from_base64()</a>	17
4.4.2.7 <a href="#">buffer_from_file()</a>	17
4.4.2.8 <a href="#">buffer_from_string()</a>	18
4.4.2.9 <a href="#">buffer_init()</a>	18
4.4.2.10 <a href="#">buffer_print()</a>	18
4.4.2.11 <a href="#">buffer_print_int()</a>	18
4.4.2.12 <a href="#">buffer_random()</a>	18
4.4.2.13 <a href="#">buffer_reset()</a>	19
4.4.2.14 <a href="#">buffer_resize()</a>	19
4.4.2.15 <a href="#">buffer_to_base64()</a>	19
4.4.2.16 <a href="#">string_from_buffer()</a>	19
4.5 <a href="#">random.c</a> File Reference	19
4.5.1 Detailed Description	20
4.5.2 Function Documentation	20
4.5.2.1 <a href="#">random_seed()</a>	20
4.6 <a href="#">random.h</a> File Reference	20
4.6.1 Function Documentation	20
4.6.1.1 <a href="#">random_seed()</a>	20

# Chapter 1

## Data Structure Index

### 1.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">buffer_t</a> . . . . .	5
------------------------------------	---



## Chapter 2

# File Index

### 2.1 File List

Here is a list of all files with brief descriptions:

<a href="#">base64.c</a>	7
<a href="#">base64.h</a>	9
<a href="#">buffer.c</a>	10
<a href="#">buffer.h</a>	15
<a href="#">random.c</a>	19
<a href="#">random.h</a>	20





## Chapter 3

# Data Structure Documentation

### 3.1 `buffer_t` Struct Reference

```
#include <buffer.h>
```

#### Data Fields

- `uchar * tab`
- `size_t size`
- `size_t length`

#### 3.1.1 Field Documentation

##### 3.1.1.1 `length`

```
size_t buffer_t::length
```

##### 3.1.1.2 `size`

```
size_t buffer_t::size
```

##### 3.1.1.3 `tab`

```
uchar* buffer_t::tab
```

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

- `buffer.h`



## Chapter 4

# File Documentation

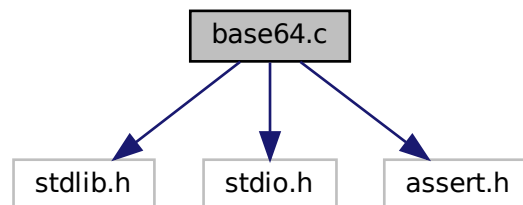
### 4.1 base64.c File Reference

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

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

```
#include <assert.h>
```

Include dependency graph for base64.c:



### Typedefs

- typedef unsigned char [uchar](#)

### Functions

- int [CodeBase64](#) ([uchar](#) out[], const [uchar](#) in[], const int N)

*Code in[] in base 64: three characters of in[0..N[ become four characters in out[]; if N is not a multiple of 3, we pad following the RFC.*

- void [DecodeBase64](#) ([uchar](#) \*out, const [uchar](#) \*in, const int N64)

*Decode in[0..N64[ in base 64 with resultat in out[]. Assume that  $4 \mid N64$  and out[] has size  $\geq 3*N64/4$ . Returns the true length.*

### 4.1.1 Detailed Description

#### Author

François Morain ( [morain@lix.polytechnique.fr](mailto:morain@lix.polytechnique.fr) )

### 4.1.2 Typedef Documentation

#### 4.1.2.1 uchar

```
typedef unsigned char uchar
```

### 4.1.3 Function Documentation

#### 4.1.3.1 CodeBase64()

```
int CodeBase64 (
    uchar out[],
    const uchar in[],
    const int N )
```

Code in[] in base 64: three characters of in[0..N[ become four characters in out[]; if N is not a multiple of 3, we padd following the RFC.

Suppose that out has size  $\geq 4*N/3$ .

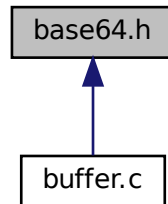
#### 4.1.3.2 DecodeBase64()

```
void DecodeBase64 (
    uchar * out,
    const uchar * in,
    const int N64 )
```

Decode in[0..N64[ in base 64 with resultat in out[]. Assume that  $4 \mid N64$  and out[] has size  $\geq 3*N64/4$ . Returns the true length.

## 4.2 base64.h File Reference

This graph shows which files directly or indirectly include this file:



### Typedefs

- typedef unsigned char [uchar](#)

### Functions

- int [CodeBase64](#) ([uchar](#) \*out, const [uchar](#) \*in, const int N)
- void [DecodeBase64](#) ([uchar](#) \*out, const [uchar](#) \*in, const int N64)

*Decode in[0..N64[ in base 64 with resultat in out[]]. Assume that  $4 \mid N64$  and out[] has size  $\geq 3*N64/4$ . Returns the true length.*

### 4.2.1 Typedef Documentation

#### 4.2.1.1 uchar

```
typedef unsigned char uchar
```

### 4.2.2 Function Documentation

#### 4.2.2.1 CodeBase64()

```
int CodeBase64 (  
    uchar * out,  
    const uchar * in,  
    const int N )
```

#### 4.2.2.2 DecodeBase64()

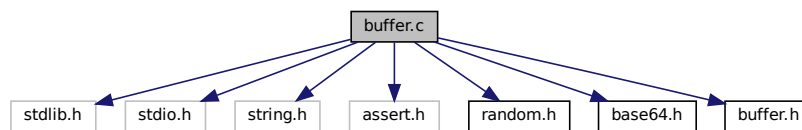
```
void DecodeBase64 (
    uchar * out,
    const uchar * in,
    const int N64 )
```

Decode in[0..N64[ in base 64 with resultat in out[]. Assume that  $4 \mid N64$  and out[] has size  $\geq 3*N64/4$ . Returns the true length.

### 4.3 buffer.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <assert.h>
#include "random.h"
#include "base64.h"
#include "buffer.h"
```

Include dependency graph for buffer.c:



### Macros

- `#define DEBUG 0`  
*set DEBUG to 1 to have a complete equality test*

### Functions

- `int buffer_init (buffer_t *buf, size_t size)`  
*initialize buf as an array of maximal size size.*
- `void buffer_clear (buffer_t *buf)`  
*Clears and frees the memory occupied by the buffer.*
- `int buffer_print (FILE *ofile, buffer_t *buf)`  
*Prints the content of buf to file descriptor ofile.*
- `int buffer_print_int (FILE *ofile, buffer_t *buf)`  
*Prints the content of buf as integers to file descriptor ofile.*
- `int buffer_resize (buffer_t *buf, size_t len)`  
*Reallocates buf->tab to size len.*
- `int buffer_append (buffer_t *buf, buffer_t *next)`  
*Appends the contents of next at the end of the contents of buf. If buf is too small, it is resized.*

- int `buffer_append_uchar` (`buffer_t` \*buf, `uchar` c)  
*Appends an element at the end of the buffer.*
- void `buffer_reset` (`buffer_t` \*buf)  
*Reset buffer to 0.*
- int `buffer_equality` (`buffer_t` \*buf1, `buffer_t` \*buf2)  
*Return 1 if buf1 == buf2 (same length, same content).*
- void `buffer_clone` (`buffer_t` \*out, `buffer_t` \*in)  
*Performs out <- in.*
- int `buffer_from_string` (`buffer_t` \*buf, `uchar` \*str, `size_t` len)  
*A string is an array of uchar's that ends with a '\0'. The '\0' is not put in the buffer. If the length is unknown give default value -1.*
- `uchar` \* `string_from_buffer` (`buffer_t` \*buf)  
*Creates a C string that contains buf.*
- int `buffer_from_file` (`buffer_t` \*buf, const char \*file\_name)  
*Fills in buf from file if exists.*
- void `buffer_random` (`buffer_t` \*out, int byte\_length)  
*Fills in a buffer with byte\_length random bytes Pseudo-random generator should have been initialised before.*
- void `buffer_to_base64` (`buffer_t` \*out, `buffer_t` \*in)
- void `buffer_from_base64` (`buffer_t` \*out, `buffer_t` \*in)

### 4.3.1 Detailed Description

#### Author

François Morain ( [morain@lix.polytechnique.fr](mailto:morain@lix.polytechnique.fr) )

Alain Couvreur ( [alain.couvreur@lix.polytechnique.fr](mailto:alain.couvreur@lix.polytechnique.fr) )

#### Date

September 29, 2018.

A `buffer_t` mimics an array of `uchar` of variable size. A buffer is created with a maximal size; each time an operation is performed, its size may be increased.

### 4.3.2 Macro Definition Documentation

#### 4.3.2.1 DEBUG

```
#define DEBUG 0
```

set DEBUG to 1 to have a complete equality test

### 4.3.3 Function Documentation

#### 4.3.3.1 `buffer_append()`

```
int buffer_append (
    buffer_t * buf,
    buffer_t * next )
```

Appends the contents of `next` at the end of the contents of `buf`. If `buf` is too small, it is resized.

#### 4.3.3.2 `buffer_append_uchar()`

```
int buffer_append_uchar (
    buffer_t * buf,
    uchar c )
```

Appends an element at the end of the buffer.

#### 4.3.3.3 `buffer_clear()`

```
void buffer_clear (
    buffer_t * buf )
```

Clears and frees the memory occupied by the buffer.

#### 4.3.3.4 `buffer_clone()`

```
void buffer_clone (
    buffer_t * out,
    buffer_t * in )
```

Performs `out <- in`.

#### 4.3.3.5 `buffer_equality()`

```
int buffer_equality (
    buffer_t * buf1,
    buffer_t * buf2 )
```

Return 1 if `buf1 == buf2` (same length, same content).



#### 4.3.3.6 buffer\_from\_base64()

```
void buffer_from_base64 (
    buffer_t * out,
    buffer_t * in )
```

#### 4.3.3.7 buffer\_from\_file()

```
int buffer_from_file (
    buffer_t * buf,
    const char * file_name )
```

Fills in *buf* from *file* if exists.

#### 4.3.3.8 buffer\_from\_string()

```
int buffer_from_string (
    buffer_t * buf,
    uchar * str,
    size_t len )
```

A string is an array of uchar's that ends with a '\0'. The '\0' is *not* put in the buffer. If the length is unknown give default value -1.

#### 4.3.3.9 buffer\_init()

```
int buffer_init (
    buffer_t * buf,
    size_t size )
```

initialize *buf* as an array of maximal size *size*.

#### 4.3.3.10 buffer\_print()

```
int buffer_print (
    FILE * ofile,
    buffer_t * buf )
```

Prints the content of *buf* to file descriptor *ofile*.

#### 4.3.3.11 `buffer_print_int()`

```
int buffer_print_int (
    FILE * ofile,
    buffer_t * buf )
```

Prints the content of *buf* as integers to file descriptor *ofile*.

#### 4.3.3.12 `buffer_random()`

```
void buffer_random (
    buffer_t * out,
    int byte_length )
```

Fills in a buffer with *byte\_length* random bytes Pseudo-random generator should have been initialised before.

#### 4.3.3.13 `buffer_reset()`

```
void buffer_reset (
    buffer_t * buf )
```

Reset buffer to 0.

#### 4.3.3.14 `buffer_resize()`

```
int buffer_resize (
    buffer_t * buf,
    size_t len )
```

Reallocates *buf*->*tab* to size *len*.

#### 4.3.3.15 `buffer_to_base64()`

```
void buffer_to_base64 (
    buffer_t * out,
    buffer_t * in )
```

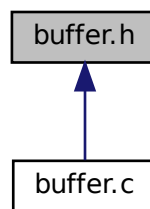
#### 4.3.3.16 string\_from\_buffer()

```
uchar* string_from_buffer (
    buffer_t * buf )
```

Creates a C string that contains *buf*.

## 4.4 buffer.h File Reference

This graph shows which files directly or indirectly include this file:



### Data Structures

- struct `buffer_t`

### Typedefs

- typedef unsigned char `uchar`

### Functions

- int `buffer_init` (`buffer_t` \*buf, size\_t size)  
*initialize buf as an array of maximal size size.*
- void `buffer_clear` (`buffer_t` \*buf)  
*Clears and frees the memory occupied by the buffer.*
- int `buffer_print` (FILE \*ofile, `buffer_t` \*buf)  
*Prints the content of buf to file descriptor ofile.*
- int `buffer_print_int` (FILE \*ofile, `buffer_t` \*buf)  
*Prints the content of buf as integers to file descriptor ofile.*
- int `buffer_resize` (`buffer_t` \*buf, size\_t len)  
*Reallocates buf->tab to size len.*
- int `buffer_append` (`buffer_t` \*buf, `buffer_t` \*next)  
*Appends the contents of next at the end of the contents of buf. If buf is too small, it is resized.*
- int `buffer_append_uchar` (`buffer_t` \*buf, uchar c)

- Appends an element at the end of the buffer.*

  - void `buffer_reset` (`buffer_t` \*buf)

*Reset buffer to 0.*
- int `buffer_equality` (`buffer_t` \*buf1, `buffer_t` \*buf2)

*Return 1 if buf1 == buf2 (same length, same content).*
- void `buffer_clone` (`buffer_t` \*out, `buffer_t` \*in)

*Performs out <- in.*
- int `buffer_from_string` (`buffer_t` \*buf, `uchar` \*str, `size_t` len)

*A string is an array of uchar's that ends with a '\0'. The '\0' is not put in the buffer. If the length is unknown give default value -1.*
- `uchar` \* `string_from_buffer` (`buffer_t` \*buf)

*Creates a C string that contains buf.*
- int `buffer_from_file` (`buffer_t` \*buf, const char \*file\_name)

*Fills in buf from file if exists.*
- void `buffer_random` (`buffer_t` \*out, int byte\_length)

*Fills in a buffer with byte\_length random bytes Pseudo-random generator should have been initialised before.*
- void `buffer_to_base64` (`buffer_t` \*out, `buffer_t` \*in)
- void `buffer_from_base64` (`buffer_t` \*out, `buffer_t` \*in)

## 4.4.1 Typedef Documentation

### 4.4.1.1 uchar

```
typedef unsigned char uchar
```

## 4.4.2 Function Documentation

### 4.4.2.1 buffer\_append()

```
int buffer_append (
    buffer_t * buf,
    buffer_t * next )
```

Appends the contents of next at the end of the contents of buf. If buf is too small, it is resized.

### 4.4.2.2 buffer\_append\_uchar()

```
int buffer_append_uchar (
    buffer_t * buf,
    uchar c )
```

Appends an element at the end of the buffer.

#### 4.4.2.3 buffer\_clear()

```
void buffer_clear (
    buffer_t * buf )
```

Clears and frees the memory occupied by the buffer.

#### 4.4.2.4 buffer\_clone()

```
void buffer_clone (
    buffer_t * out,
    buffer_t * in )
```

Performs out <- in.

#### 4.4.2.5 buffer\_equality()

```
int buffer_equality (
    buffer_t * buf1,
    buffer_t * buf2 )
```

Return 1 if *buf1* == *buf2* (same length, same content).

#### 4.4.2.6 buffer\_from\_base64()

```
void buffer_from_base64 (
    buffer_t * out,
    buffer_t * in )
```

#### 4.4.2.7 buffer\_from\_file()

```
int buffer_from_file (
    buffer_t * buf,
    const char * file_name )
```

Fills in *buf* from *file* if exists.

#### 4.4.2.8 buffer\_from\_string()

```
int buffer_from_string (
    buffer_t * buf,
    uchar * str,
    size_t len )
```

A string is an array of uchar's that ends with a '\0'. The '\0' is *not* put in the buffer. If the length is unknown give default value -1.

#### 4.4.2.9 buffer\_init()

```
int buffer_init (
    buffer_t * buf,
    size_t size )
```

initialize *buf* as an array of maximal size *size*.

#### 4.4.2.10 buffer\_print()

```
int buffer_print (
    FILE * ofile,
    buffer_t * buf )
```

Prints the content of *buf* to file descriptor *ofile*.

#### 4.4.2.11 buffer\_print\_int()

```
int buffer_print_int (
    FILE * ofile,
    buffer_t * buf )
```

Prints the content of *buf* as integers to file descriptor *ofile*.

#### 4.4.2.12 buffer\_random()

```
void buffer_random (
    buffer_t * out,
    int byte_length )
```

Fills in a buffer with *byte\_length* random bytes Pseudo-random generator should have been initialised before.

#### 4.4.2.13 buffer\_reset()

```
void buffer_reset (
    buffer_t * buf )
```

Reset buffer to 0.

#### 4.4.2.14 buffer\_resize()

```
int buffer_resize (
    buffer_t * buf,
    size_t len )
```

Reallocates *buf*->*tab* to size *len*.

#### 4.4.2.15 buffer\_to\_base64()

```
void buffer_to_base64 (
    buffer_t * out,
    buffer_t * in )
```

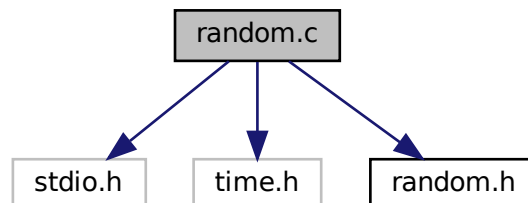
#### 4.4.2.16 string\_from\_buffer()

```
uchar* string_from_buffer (
    buffer_t * buf )
```

Creates a C string that contains *buf*.

## 4.5 random.c File Reference

```
#include <stdio.h>
#include <time.h>
#include "random.h"
Include dependency graph for random.c:
```



## Functions

- unsigned int [random\\_seed](#) ()  
*returns a random seed as random as it can be.*

### 4.5.1 Detailed Description

Author

Alain Couvreur

### 4.5.2 Function Documentation

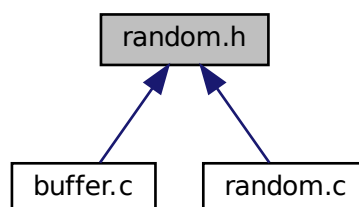
#### 4.5.2.1 [random\\_seed\(\)](#)

```
unsigned int random_seed ( )
```

returns a random seed as random as it can be.

## 4.6 random.h File Reference

This graph shows which files directly or indirectly include this file:



## Functions

- unsigned int [random\\_seed](#) ()  
*returns a random seed as random as it can be.*

### 4.6.1 Function Documentation

#### 4.6.1.1 [random\\_seed\(\)](#)

```
unsigned int random_seed ( )
```

returns a random seed as random as it can be.



# Index

- base64.c, [7](#)
  - CodeBase64, [8](#)
  - DecodeBase64, [8](#)
  - uchar, [8](#)
- base64.h, [9](#)
  - CodeBase64, [9](#)
  - DecodeBase64, [9](#)
  - uchar, [9](#)
- buffer.c, [10](#)
  - buffer\_append, [11](#)
  - buffer\_append\_uchar, [12](#)
  - buffer\_clear, [12](#)
  - buffer\_clone, [12](#)
  - buffer\_equality, [12](#)
  - buffer\_from\_base64, [12](#)
  - buffer\_from\_file, [13](#)
  - buffer\_from\_string, [13](#)
  - buffer\_init, [13](#)
  - buffer\_print, [13](#)
  - buffer\_print\_int, [13](#)
  - buffer\_random, [14](#)
  - buffer\_reset, [14](#)
  - buffer\_resize, [14](#)
  - buffer\_to\_base64, [14](#)
  - DEBUG, [11](#)
  - string\_from\_buffer, [14](#)
- buffer.h, [15](#)
  - buffer\_append, [16](#)
  - buffer\_append\_uchar, [16](#)
  - buffer\_clear, [16](#)
  - buffer\_clone, [17](#)
  - buffer\_equality, [17](#)
  - buffer\_from\_base64, [17](#)
  - buffer\_from\_file, [17](#)
  - buffer\_from\_string, [17](#)
  - buffer\_init, [18](#)
  - buffer\_print, [18](#)
  - buffer\_print\_int, [18](#)
  - buffer\_random, [18](#)
  - buffer\_reset, [18](#)
  - buffer\_resize, [19](#)
  - buffer\_to\_base64, [19](#)
  - string\_from\_buffer, [19](#)
  - uchar, [16](#)
- buffer\_append
  - buffer.c, [11](#)
  - buffer.h, [16](#)
- buffer\_append\_uchar
  - buffer.c, [12](#)
- buffer.h, [16](#)
  - buffer\_clear
    - buffer.c, [12](#)
    - buffer.h, [16](#)
  - buffer\_clone
    - buffer.c, [12](#)
    - buffer.h, [17](#)
  - buffer\_equality
    - buffer.c, [12](#)
    - buffer.h, [17](#)
  - buffer\_from\_base64
    - buffer.c, [12](#)
    - buffer.h, [17](#)
  - buffer\_from\_file
    - buffer.c, [13](#)
    - buffer.h, [17](#)
  - buffer\_from\_string
    - buffer.c, [13](#)
    - buffer.h, [17](#)
  - buffer\_init
    - buffer.c, [13](#)
    - buffer.h, [18](#)
  - buffer\_print
    - buffer.c, [13](#)
    - buffer.h, [18](#)
  - buffer\_print\_int
    - buffer.c, [13](#)
    - buffer.h, [18](#)
  - buffer\_random
    - buffer.c, [14](#)
    - buffer.h, [18](#)
  - buffer\_reset
    - buffer.c, [14](#)
    - buffer.h, [18](#)
  - buffer\_resize
    - buffer.c, [14](#)
    - buffer.h, [19](#)
  - buffer\_t, [5](#)
    - length, [5](#)
    - size, [5](#)
    - tab, [5](#)
  - buffer\_to\_base64
    - buffer.c, [14](#)
    - buffer.h, [19](#)
  - CodeBase64
    - base64.c, [8](#)
    - base64.h, [9](#)
  - DEBUG

- buffer.c, [11](#)
- DecodeBase64
  - base64.c, [8](#)
  - base64.h, [9](#)
- length
  - buffer\_t, [5](#)
- random.c, [19](#)
  - random\_seed, [20](#)
- random.h, [20](#)
  - random\_seed, [20](#)
- random\_seed
  - random.c, [20](#)
  - random.h, [20](#)
- size
  - buffer\_t, [5](#)
- string\_from\_buffer
  - buffer.c, [14](#)
  - buffer.h, [19](#)
- tab
  - buffer\_t, [5](#)
- uchar
  - base64.c, [8](#)
  - base64.h, [9](#)
  - buffer.h, [16](#)