gds

Generated by Doxygen 1.8.1.2

Sat Nov 8 2014 16:55:51

Contents

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 gdt_generic_datatype Struct Reference

Generic datatype structure.

`#include <gdt.h>`

Data Fields

- enum gds_datatype type
- gds_cfunc compfunc
- union {
  char c
  unsigned char uc
  signed char sc
  int i
  unsigned int ui
  long l
  unsigned long ul
  long long int ll
  unsigned long long int ull
  size_t st
  double d
  char * pc
  void * p
  } data

3.1.1 Detailed Description

Generic datatype structure.

3.1.2 Field Documentation

3.1.2.1 char gdt_generic_datatype::c

char

3.1.2.2 gds_cfunc gdt_generic_datatype::compfunc

Comparison function pointer

3.1.2.3 double gdt_generic_datatype::d

double

3.1.2.4 union { ... } gdt_generic_datatype::data

Data union

3.1.2.5 int gdt_generic_datatype::i

int

3.1.2.6 long gdt_generic_datatype::l

long

3.1.2.7 long long int gdt_generic_datatype::ll

long long

3.1.2.8 void* gdt_generic_datatype::p

void *

3.1.2.9 char* gdt_generic_datatype::pc

char *, string

3.1.2.10 signed char gdt_generic_datatype::sc

signed char

3.1.2.11 size_t gdt_generic_datatype::st

size_t

3.1.2.12 enum gds_datatype gdt_generic_datatype::type

Data type

3.1.2.13 unsigned char gdt_generic_datatype::uc

unsigned char

3.1.2.14 unsigned int gdt_generic_datatype::ui

unsigned int

3.1.2.15 unsigned long gdt_generic_datatype::ul

unsigned long

3.1.2.16 unsigned long long int gdt_generic_datatype::ull

unsigned long long

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

- include/private/gdt.h

3.2 hms Struct Reference

Data Fields

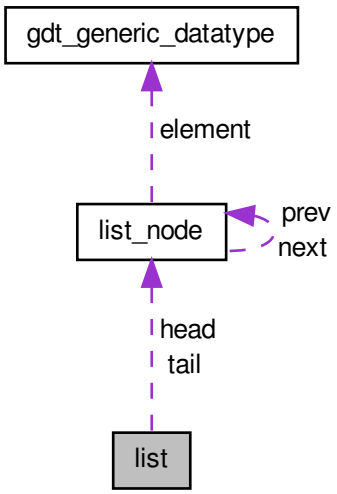
- int **hour**
- int **minute**
- int **second**

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

- tests/test_list.c
- tests/test_vector.c

3.3 list Struct Reference

Collaboration diagram for list:



Data Fields

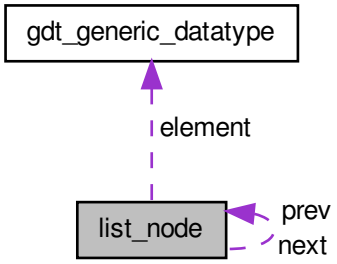
- size_t **length**
- enum gds_datatype **type**
- gds_cfunc **compfunc**
- struct list_node * **head**
- struct list_node * **tail**
- bool **free_on_destroy**
- bool **exit_on_error**

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

- src/list.c

3.4 list_node Struct Reference

Collaboration diagram for list_node:



Data Fields

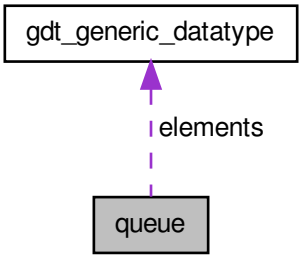
- struct gdt_generic_datatype **element**
- struct list_node * **prev**
- struct list_node * **next**

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

- src/list.c

3.5 queue Struct Reference

Collaboration diagram for queue:



Data Fields

- size_t **front**

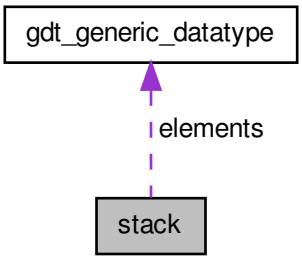
- size_t **back**
- size_t **capacity**
- size_t **size**
- enum gds_datatype **type**
- struct gdt_generic_datatype * **elements**
- bool **resizable**
- bool **free_on_destroy**
- bool **exit_on_error**

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

- src/queue.c

3.6 stack Struct Reference

Collaboration diagram for stack:



Data Fields

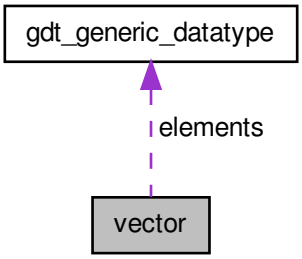
- size_t **top**
- size_t **capacity**
- enum gds_datatype **type**
- struct gdt_generic_datatype * **elements**
- bool **resizable**
- bool **free_on_destroy**
- bool **exit_on_error**

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

- src/stack.c

3.7 vector Struct Reference

Collaboration diagram for vector:



Data Fields

- size_t **length**
- size_t **capacity**
- enum gds_datatype **type**
- struct gdt_generic_datatype ∗ **elements**
- int(∗ **compfunc**)(const void ∗, const void ∗)
- bool **free_on_destroy**
- bool **exit_on_error**

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

- src/vector.c

Chapter 4

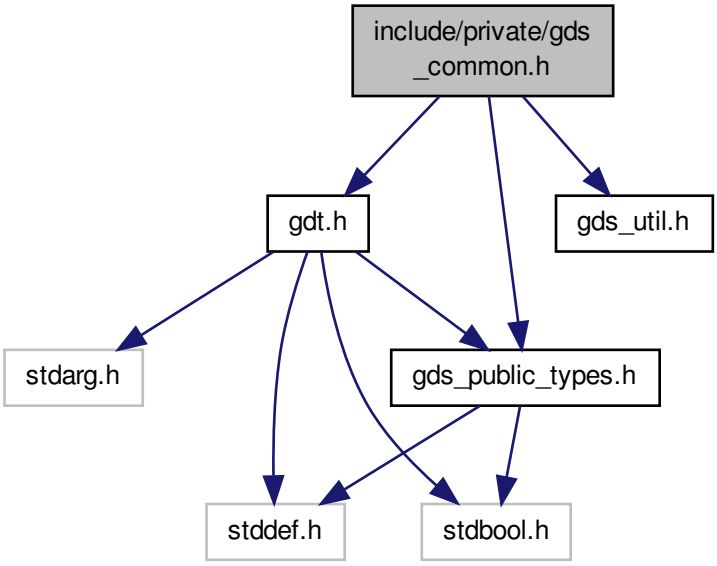
File Documentation

4.1 include/private/gds_common.h File Reference

Common internal headers for data structures.

```
#include "gds_public_types.h"
#include "gdt.h"
#include "gds_util.h"
```

Include dependency graph for gds_common.h:



4.1.1 Detailed Description

Common internal headers for data structures.

Author

Paul Griffiths

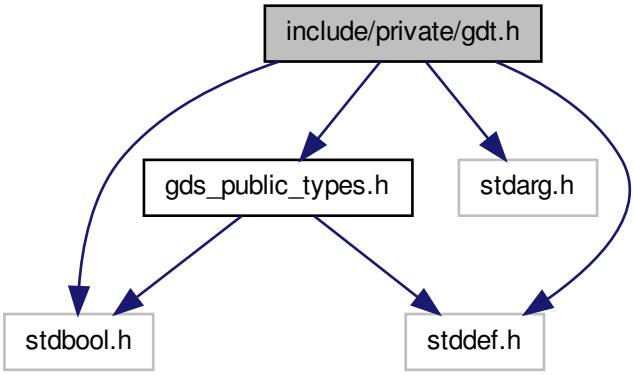
Copyright

4.2 include/private/gdt.h File Reference

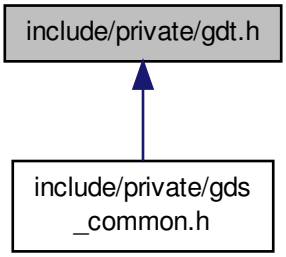
Interface to generic data element functionality.

```
#include <stdbool.h>
#include <stddef.h>
#include <stdarg.h>
#include "gds_public_types.h"
```

Include dependency graph for gdt.h:



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



Data Structures

- struct gdt_generic_datatype

  Generic datatype structure.

Functions

- void **gdt_set_value** (struct gdt_generic_datatype *data, const enum gds_datatype, gds_cfunc cfunc, va_list ap)
- void **gdt_get_value** (const struct gdt_generic_datatype *data, void *p)
- void **gdt_free** (struct gdt_generic_datatype *data)
- int **gdt_compare** (const struct gdt_generic_datatype *d1, const struct gdt_generic_datatype *d2)
- int **gdt_compare_void** (const void *p1, const void *p2)
- int **gdt_reverse_compare_void** (const void *p1, const void *p2)

4.2.1 Detailed Description

Interface to generic data element functionality.

Author

Paul Griffiths

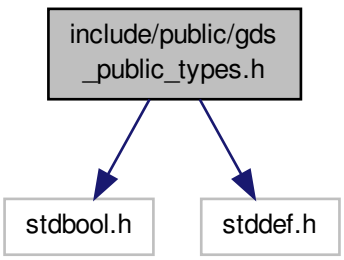
Copyright

4.3 include/public/gds_public_types.h File Reference

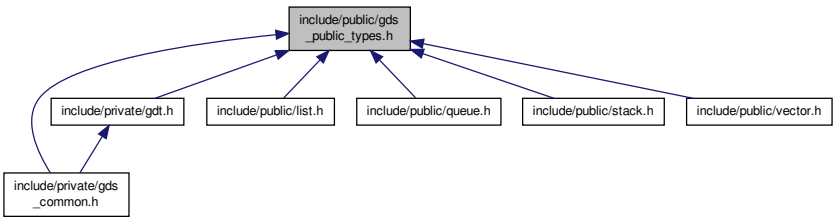
Common public types for generic data structures library.

```
#include <stdbool.h>
#include <stddef.h>
```

Include dependency graph for gds_public_types.h:



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



Typedefs

- typedef int(∗ gds_cfunc)(const void ∗, const void ∗)

  Type definition for comparison function pointer.

Enumerations

- enum gds_option { GDS_RESIZABLE = 1, GDS_FREE_ON_DESTROY = 2, GDS_EXIT_ON_ERROR = 4 }

  Enumeration type for data structure options.
- enum gds_datatype {
  DATATYPE_CHAR, DATATYPE_UNSIGNED_CHAR, DATATYPE_SIGNED_CHAR, DATATYPE_INT, DATATYPE_UNSIGNED_INT, DATATYPE_LONG, DATATYPE_UNSIGNED_LONG, DATATYPE_LONG_LONG,
  DATATYPE_UNSIGNED_LONG_LONG, DATATYPE_SIZE_T, DATATYPE_DOUBLE, DATATYPE_STRING,
  DATATYPE_POINTER }

  Enumeration type for data element type.

4.3.1 Detailed Description

Common public types for generic data structures library.

Author

Paul Griffiths

Copyright

4.3.2 Enumeration Type Documentation

4.3.2.1 enum gds_datatype

Enumeration type for data element type.

Enumerator:

DATATYPE_CHAR char

DATATYPE_UNSIGNED_CHAR unsigned char

DATATYPE_SIGNED_CHAR signed char

DATATYPE_INT int

DATATYPE_UNSIGNED_INT unsigned int

DATATYPE_LONG long

DATATYPE_UNSIGNED_LONG unsigned long

DATATYPE_LONG_LONG long long

DATATYPE_UNSIGNED_LONG_LONG unsigned long long

DATATYPE_SIZE_T size_t

DATATYPE_DOUBLE double

DATATYPE_STRING char *, string

DATATYPE_POINTER void *

4.3.2.2 enum gds_option

Enumeration type for data structure options.

Enumerator:

GDS_RESIZABLE Dynamically resizes on demand

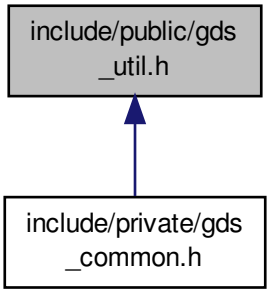
GDS_FREE_ON_DESTROY Automatically frees pointer members

GDS_EXIT_ON_ERROR Exits on error

4.4 include/public/gds_util.h File Reference

Interface to general utility functions.

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



Functions

- void gds_strerror_quit (const char *msg,...)

  Prints an error message with error number and exits.

- void gds_error_quit (const char *msg,...)

  Prints an error message exits.
- void gds_assert_quit (const char *msg,...)

  Prints an error message exits via assert().

4.4.1 Detailed Description

Interface to general utility functions.

Author

Paul Griffiths

Copyright

4.4.2 Function Documentation

4.4.2.1 void gds_assert_quit (const char * *msg*, ...)

Prints an error message exits via assert().

This function will do nothing if `NDEBUG` is defined. Otherwise, it behaves in a manner identical to `gds_error_quit()` except it terminates via `assert()`, rather than `exit()`.

Parameters

msg	The format string for the message to print. Format specifiers are the same as the `printf()` family of functions.
...	Any arguments to the format string.

4.4.2.2 void gds_error_quit (const char * *msg*, ...)

Prints an error message exits.

Parameters

msg	The format string for the message to print. Format specifiers are the same as the `printf()` family of functions.
...	Any arguments to the format string.

4.4.2.3 void gds_strerror_quit (const char * *msg*, ...)

Prints an error message with error number and exits.

This function can be called to print an error message and quit following a function which has indicated failure and has set `errno`.

Parameters

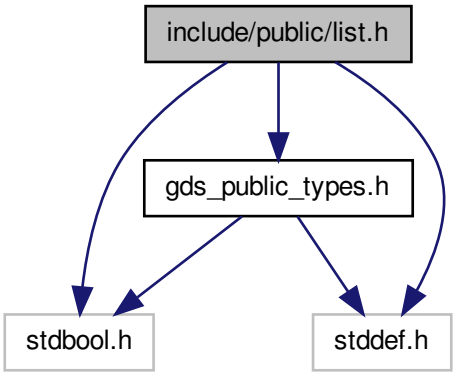
msg	The format string for the message to print. Format specifiers are the same as the `printf()` family of functions.
...	Any arguments to the format string.

4.5 include/public/list.h File Reference

Interface to generic list data structure.

```
#include <stdbool.h>
#include <stddef.h>
#include "gds_public_types.h"
```

Include dependency graph for list.h:



Typedefs

- typedef struct list * List

Functions

- List list_create (const enum gds_datatype type, const int opts,...)

  Creates a new list.
- void **list_destroy** (List list)
- bool **list_append** (List list,...)
- bool **list_prepend** (List list,...)
- bool **list_insert** (List list, const size_t index,...)
- bool **list_delete_index** (List list, const size_t index)
- bool **list_delete_front** (List list)
- bool **list_delete_back** (List list)
- bool **list_element_at_index** (List list, const size_t index, void *p)
- bool **list_set_element_at_index** (List list, const size_t index,...)
- bool **list_find** (List list, size_t *index,...)
- bool **list_is_empty** (List list)
- size_t **list_length** (List list)

4.5.1 Detailed Description

Interface to generic list data structure. The list is implemented as a double-ended, double-linked list.

Author

Paul Griffiths

Copyright

4.5.2 Typedef Documentation

4.5.2.1 typedef struct list* List

Opaque list type definition

4.5.3 Function Documentation

4.5.3.1 List list_create (const enum gds_datatype *type*, const int *opts*, ...) [read]

Creates a new list.

Parameters

type	The datatype for the list.
opts	The following options can be OR'd together: GDS_FREE_ON_DESTROY to automatically `free()` pointer members when they are deleted or when the list is destroyed; GDS_EXIT_ON_ERROR to print a message to the standard error stream and `exit()`, rather than returning a failure status.
...	If `type` is `DATATYPE_POINTER`, this argument should be a pointer to a comparison function. In all other cases, this argument is not required, and will be ignored if it is provided.

Return values

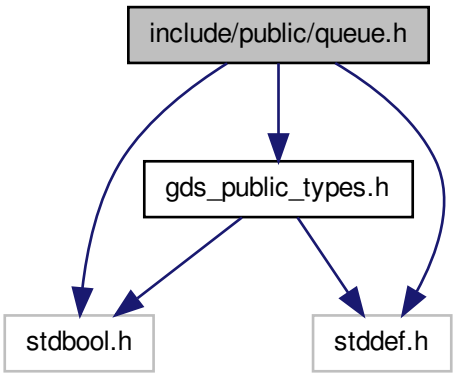
NULL	List creation failed.
non-NULL	A pointer to the new list.

4.6 include/public/queue.h File Reference

Interface to generic queue data structure.

```
#include <stdbool.h>
#include <stddef.h>
#include "gds_public_types.h"
```

Include dependency graph for queue.h:



Typedefs

- typedef struct queue * **Queue**

Functions

- Queue **queue_create** (const size_t capacity, const enum gds_datatype type, const int opts)
- void **queue_destroy** (Queue queue)
- bool **queue_push** (Queue queue,...)
- bool **queue_pop** (Queue queue, void *p)
- bool **queue_peek** (Queue queue, void *p)
- bool **queue_is_full** (Queue queue)
- bool **queue_is_empty** (Queue queue)
- size_t **queue_capacity** (Queue queue)
- size_t **queue_free_space** (Queue queue)
- size_t **queue_size** (Queue queue)

4.6.1 Detailed Description

Interface to generic queue data structure.

Author

Paul Griffiths

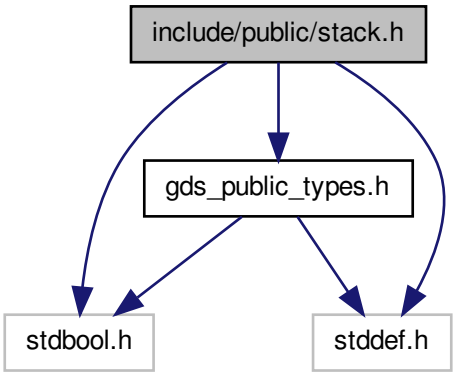
Copyright

4.7 include/public/stack.h File Reference

Interface to generic stack data structure.

```
#include <stdbool.h>
#include <stddef.h>
#include "gds_public_types.h"
```

Include dependency graph for stack.h:



Typedefs

- typedef struct stack * **Stack**

Functions

- Stack **stack_create** (const size_t capacity, const enum gds_datatype type, const int opts)
- void **stack_destroy** (Stack stack)
- bool **stack_push** (Stack stack,...)
- bool **stack_pop** (Stack stack, void *p)
- bool **stack_peek** (Stack stack, void *p)
- bool **stack_is_full** (Stack stack)
- bool **stack_is_empty** (Stack stack)
- size_t **stack_capacity** (Stack stack)
- size_t **stack_free_space** (Stack stack)
- size_t **stack_size** (Stack stack)

4.7.1 Detailed Description

Interface to generic stack data structure.

Author

Paul Griffiths

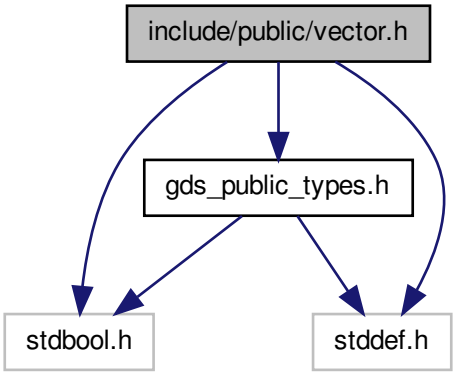
Copyright

4.8 include/public/vector.h File Reference

Interface to generic vector data structure.

```
#include <stdbool.h>
#include <stddef.h>
#include "gds_public_types.h"
```

Include dependency graph for vector.h:



Typedefs

- typedef struct vector * **Vector**

Functions

- Vector **vector_create** (const size_t capacity, const enum gds_datatype type, const int opts,...)
- void **vector_destroy** (Vector vector)
- bool **vector_append** (Vector vector,...)
- bool **vector_prepend** (Vector vector,...)
- bool **vector_insert** (Vector vector, const size_t index,...)
- bool **vector_delete_index** (Vector vector, const size_t index)
- bool **vector_delete_front** (Vector vector)
- bool **vector_delete_back** (Vector vector)
- bool **vector_element_at_index** (Vector vector, const size_t index, void *p)
- bool **vector_set_element_at_index** (Vector vector, const size_t index,...)
- bool **vector_find** (Vector vector, size_t *index,...)
- void **vector_sort** (Vector vector)
- void **vector_reverse_sort** (Vector vector)
- bool **vector_is_empty** (Vector vector)
- size_t **vector_length** (Vector vector)

- size_t **vector_capacity** (Vector vector)
- size_t **vector_free_space** (Vector vector)

4.8.1 Detailed Description

Interface to generic vector data structure.

Author

Paul Griffiths

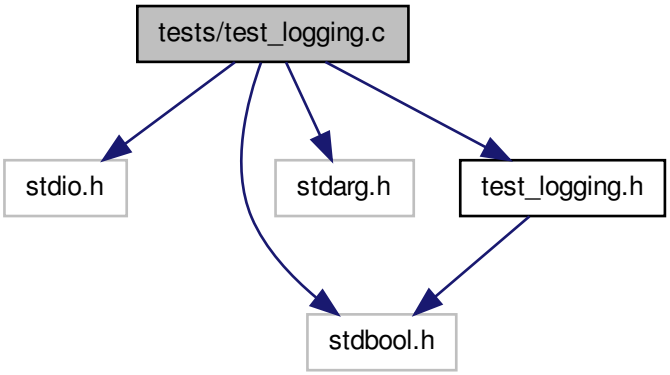
Copyright

4.9 tests/test_logging.c File Reference

Implementation of unit test logging functionality.

```
#include <stdio.h>
#include <stdbool.h>
#include <stdarg.h>
#include "test_logging.h"
```

Include dependency graph for test_logging.c:



Functions

- void tests_log_test (const bool success, const char *fmt,...)

  Logs the result of a unit test.
- int tests_get_total_tests (void)

  Returns the total number of tests run.
- int tests_get_successes (void)

  Returns the total number of successful tests.
- int tests_get_failures (void)

  Returns the total number of failed tests.

4.9.1 Detailed Description

Implementation of unit test logging functionality.

Author

Paul Griffiths

Copyright

4.9.2 Function Documentation

4.9.2.1 int tests_get_failures (void)

Returns the total number of failed tests.

Returns

The total number of failed tests.

4.9.2.2 int tests_get_successes (void)

Returns the total number of successful tests.

Returns

The total number of successful tests.

4.9.2.3 int tests_get_total_tests (void)

Returns the total number of tests run.

Returns

The total number of tests run.

4.9.2.4 void tests_log_test (const bool *success*, const char * *fmt*, ...)

Logs the result of a unit test.

Parameters

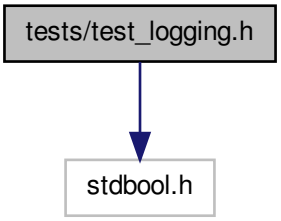
success	`true` if the test succeeded, `false` otherwise.
fmt	Format string for failure message.
...	Arguements to format string.

4.10 tests/test_logging.h File Reference

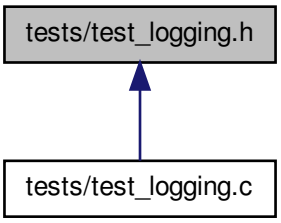
Interface to unit test logging functionality.

```
#include <stdbool.h>
```

Include dependency graph for test_logging.h:



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



Functions

- void tests_log_test (const bool success, const char *fmt,...)

  Logs the result of a unit test.
- int tests_get_total_tests (void)

  Returns the total number of tests run.
- int tests_get_successes (void)

  Returns the total number of successful tests.
- int tests_get_failures (void)

  Returns the total number of failed tests.

4.10.1 Detailed Description

Interface to unit test logging functionality.

Author

Paul Griffiths

Copyright

4.10.2 Function Documentation

4.10.2.1 int tests_get_failures (void)

Returns the total number of failed tests.

Returns

The total number of failed tests.

4.10.2.2 int tests_get_successes (void)

Returns the total number of successful tests.

Returns

The total number of successful tests.

4.10.2.3 int tests_get_total_tests (void)

Returns the total number of tests run.

Returns

The total number of tests run.

4.10.2.4 void tests_log_test (const bool *success*, const char * *fmt*, ...)

Logs the result of a unit test.

Parameters

success	`true` if the test succeeded, `false` otherwise.
fmt	Format string for failure message.
...	Arguements to format string.

Index