gds

Generated by Doxygen 1.8.1.2

Sat Nov 8 2014 16:28:18

Contents

1 Data Structure Index **1**

1.1 Data Structures 1

2 File Index **3**

2.1 File List 3

3 Data Structure Documentation **5**

3.1 gdt_generic_datatype Struct Reference 5

3.2 hms Struct Reference 5

3.3 list Struct Reference 6

3.4 list_node Struct Reference 7

3.5 queue Struct Reference 7

3.6 stack Struct Reference 8

3.7 vector Struct Reference 9

4 File Documentation **11**

4.1 include/public/gds_public_types.h File Reference 11

4.1.1 Detailed Description 12

4.1.2 Enumeration Type Documentation 12

4.1.2.1 gds_datatype 12

4.1.2.2 gds_option 13

4.2 include/public/gds_util.h File Reference 13

4.2.1 Detailed Description 13

4.2.2 Function Documentation 14

4.2.2.1 gds_assert_quit 14

4.2.2.2 gds_error_quit 14

4.2.2.3 gds_strerror_quit 14

4.3 include/public/list.h File Reference 14

4.3.1 Detailed Description 15

4.3.2 Typedef Documentation 16

4.3.2.1 List 16

4.3.3 Function Documentation 16

4.3.3.1 list_create 16
4.4 tests/test_logging.c File Reference 16
4.4.1 Detailed Description 17
4.4.2 Function Documentation 17
4.4.2.1 tests_get_failures 17
4.4.2.2 tests_get_successes 17
4.4.2.3 tests_get_total_tests 17
4.4.2.4 tests_log_test 18
4.5 tests/test_logging.h File Reference 18
4.5.1 Detailed Description 19
4.5.2 Function Documentation 19
4.5.2.1 tests_get_failures 19
4.5.2.2 tests_get_successes 19
4.5.2.3 tests_get_total_tests 19
4.5.2.4 tests_log_test 19

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

gdt_generic_datatype 5
hms 5
list 6
list_node 7
queue 7
stack 8
vector 9

Chapter 2

File Index

2.1 File List

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

include/private/**gds_common.h** . . . . **??**
include/private/**gdt.h** . . . . **??**
include/public/gds_public_types.h
Common public types for generic data structures library 11
include/public/gds_util.h
Interface to general utility functions 13
include/public/list.h
Interface to generic list data structure 14
include/public/**queue.h** . . . . **??**
include/public/**stack.h** . . . . **??**
include/public/**vector.h** . . . . **??**
tests/**test_list.h** . . . . **??**
tests/test_logging.c
Implementation of unit test logging functionality 16
tests/test_logging.h
Interface to unit test logging functionality 18
tests/**test_queue.h** . . . . **??**
tests/**test_stack.h** . . . . **??**
tests/**test_vector.h** . . . . **??**

Chapter 3

Data Structure Documentation

3.1 gdt_generic_datatype Struct Reference

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**

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

- include/private/gdt.h

3.2 hms Struct Reference

Data Fields

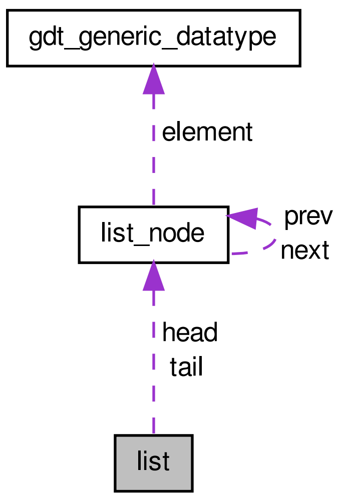
- int **hour**
- int **minute**
- int **second**

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

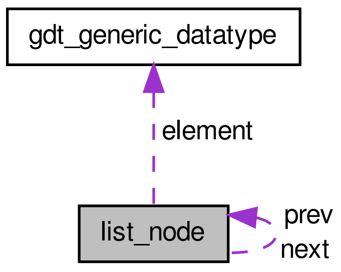
- size_t **length**
- enum gds_datatype **type**
- gds_cfunc **compfunc**
- struct list_node ∗ **head**
- struct list_node ∗ **tail**
- bool **free_on_destroy**
- bool **exit_on_error**

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

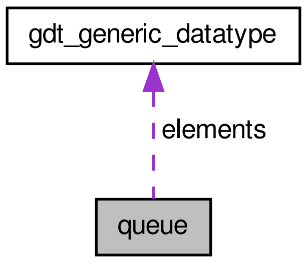
- struct gdt_generic_datatype **element**
- struct list_node ∗ **prev**
- struct list_node ∗ **next**

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**

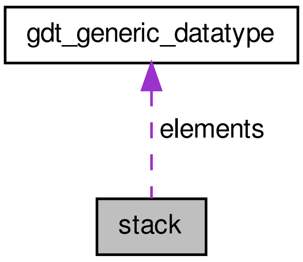
- size_t **back**
- size_t **capacity**
- size_t **size**
- enum gds_datatype **type**
- struct gdt_generic_datatype ∗ **elements**
- bool **resizable**
- bool **free_on_destroy**
- bool **exit_on_error**

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

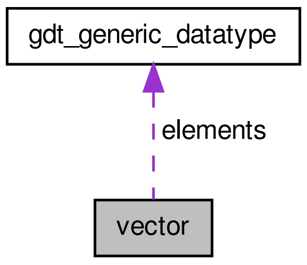
- size_t **top**
- size_t **capacity**
- enum gds_datatype **type**
- struct gdt_generic_datatype ∗ **elements**
- bool **resizable**
- bool **free_on_destroy**
- bool **exit_on_error**

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

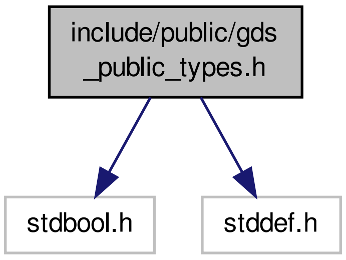
File Documentation

4.1 include/public/gds_public_types.h File Reference

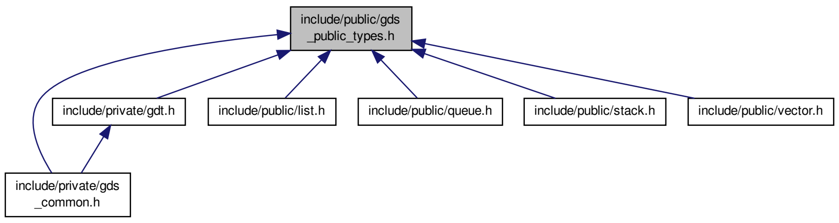
Common public types for generic data structures library.

```
#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.

4.1.1 Detailed Description

Common public types for generic data structures library.

Author

Paul Griffiths

Copyright

4.1.2 Enumeration Type Documentation

4.1.2.1 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 $*$

4.1.2.2 enum gds_option

Enumeration type for data structure options.

Enumerator:

GDS_RESIZABLE Dynamically resizes on demand

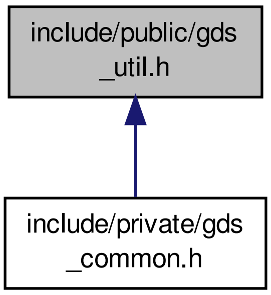
GDS_FREE_ON_DESTROY Automatically frees pointer members

GDS_EXIT_ON_ERROR Exits on error

4.2 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().

4.2.1 Detailed Description

Interface to general utility functions.

Author

Paul Griffiths

Copyright

4.2.2 Function Documentation

4.2.2.1 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.

4.2.2.2 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.

4.2.2.3 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

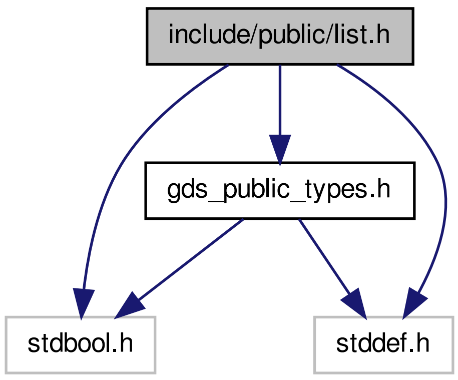
msg	The format string for the message to print. Format specifiers are the same as the `printf()` family of functions.
...	Any arguments to the format string.

4.3 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)

4.3.1 Detailed Description

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

Author

Paul Griffiths

Copyright

4.3.2 Typedef Documentation

4.3.2.1 typedef struct list∗ List

Opaque list type definition

4.3.3 Function Documentation

4.3.3.1 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

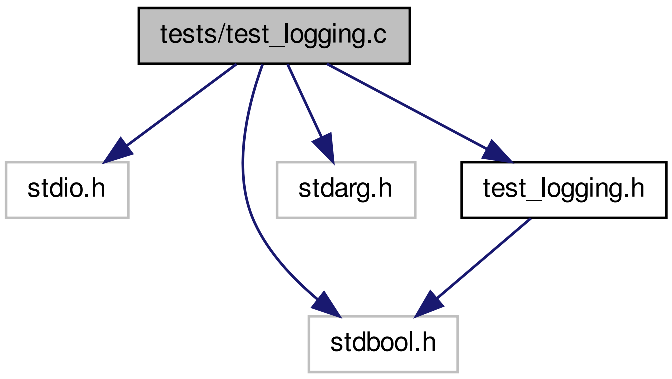
NULL	List creation failed.
non-NULL	A pointer to the new list.

4.4 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.

4.4.1 Detailed Description

Implementation of unit test logging functionality.

Author

Paul Griffiths

Copyright

4.4.2 Function Documentation

4.4.2.1 int tests_get_failures (void)

Returns the total number of failed tests.

Returns

The total number of failed tests.

4.4.2.2 int tests_get_successes (void)

Returns the total number of successful tests.

Returns

The total number of successful tests.

4.4.2.3 int tests_get_total_tests (void)

Returns the total number of tests run.

Returns

The total number of tests run.

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

Logs the result of a unit test.

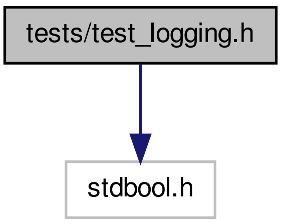
Parameters

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

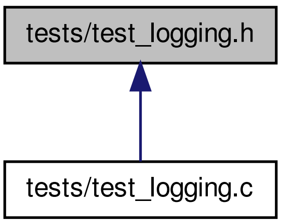
4.5 tests/test_logging.h File Reference

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.

4.5.1 Detailed Description

Interface to unit test logging functionality.

Author

Paul Griffiths

Copyright

4.5.2 Function Documentation

4.5.2.1 int tests_get_failures (void)

Returns the total number of failed tests.

Returns

The total number of failed tests.

4.5.2.2 int tests_get_successes (void)

Returns the total number of successful tests.

Returns

The total number of successful tests.

4.5.2.3 int tests_get_total_tests (void)

Returns the total number of tests run.

Returns

The total number of tests run.

4.5.2.4 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

DATATYPE_CHAR
  gds_public_types.h, 12
DATATYPE_DOUBLE
  gds_public_types.h, 12
DATATYPE_INT
  gds_public_types.h, 12
DATATYPE_LONG
  gds_public_types.h, 12
DATATYPE_LONG_LONG
  gds_public_types.h, 12
DATATYPE_POINTER
  gds_public_types.h, 12
DATATYPE_SIGNED_CHAR
  gds_public_types.h, 12
DATATYPE_SIZE_T
  gds_public_types.h, 12
DATATYPE_STRING
  gds_public_types.h, 12
DATATYPE_UNSIGNED_CHAR
  gds_public_types.h, 12
DATATYPE_UNSIGNED_INT
  gds_public_types.h, 12
DATATYPE_UNSIGNED_LONG
  gds_public_types.h, 12
DATATYPE_UNSIGNED_LONG_LONG
  gds_public_types.h, 12

GDS_EXIT_ON_ERROR
  gds_public_types.h, 13
GDS_FREE_ON_DESTROY
  gds_public_types.h, 13
GDS_RESIZABLE
  gds_public_types.h, 13
gds_public_types.h
  DATATYPE_CHAR, 12
  DATATYPE_DOUBLE, 12
  DATATYPE_INT, 12
  DATATYPE_LONG, 12
  DATATYPE_LONG_LONG, 12
  DATATYPE_POINTER, 12
  DATATYPE_SIGNED_CHAR, 12
  DATATYPE_SIZE_T, 12
  DATATYPE_STRING, 12
  DATATYPE_UNSIGNED_CHAR, 12
  DATATYPE_UNSIGNED_INT, 12
  DATATYPE_UNSIGNED_LONG, 12
  DATATYPE_UNSIGNED_LONG_LONG, 12
  GDS_EXIT_ON_ERROR, 13
  GDS_FREE_ON_DESTROY, 13
  GDS_RESIZABLE, 13
gds_assert_quit
  gds_util.h, 14
gds_datatype
  gds_public_types.h, 12
gds_error_quit
  gds_util.h, 14
gds_option
  gds_public_types.h, 12
gds_public_types.h
  gds_datatype, 12
  gds_option, 12
gds_strerror_quit
  gds_util.h, 14
gds_util.h
  gds_assert_quit, 14
  gds_error_quit, 14
  gds_strerror_quit, 14
gdt_generic_datatype, 5

hms, 5

include/public/gds_public_types.h, 11
include/public/gds_util.h, 13
include/public/list.h, 14

List
  list.h, 16
list, 6
list.h
  List, 16
  list_create, 16
list_create
  list.h, 16
list_node, 7

queue, 7

stack, 8

test_logging.c
  tests_get_failures, 17
  tests_get_successes, 17
  tests_get_total_tests, 17
  tests_log_test, 17
test_logging.h
  tests_get_failures, 19
  tests_get_successes, 19
  tests_get_total_tests, 19
  tests_log_test, 19
tests/test_logging.c, 16
tests/test_logging.h, 18

tests_get_failures
    test_logging.c, 17
    test_logging.h, 19
tests_get_successes
    test_logging.c, 17
    test_logging.h, 19
tests_get_total_tests
    test_logging.c, 17
    test_logging.h, 19
tests_log_test
    test_logging.c, 17
    test_logging.h, 19

vector, 9