

airport-sim

1

Generated by Doxygen 1.8.5

Tue Sep 2 2014 20:44:57

## Contents

<b>1</b>	<b>Data Structure Index</b>	<b>1</b>
1.1	Data Structures . . . . .	1
<b>2</b>	<b>File Index</b>	<b>1</b>
2.1	File List . . . . .	1
<b>3</b>	<b>Data Structure Documentation</b>	<b>2</b>
3.1	airport Struct Reference . . . . .	2
3.1.1	Detailed Description . . . . .	2
3.1.2	Field Documentation . . . . .	2
3.2	bay Struct Reference . . . . .	2
3.2.1	Detailed Description . . . . .	3
3.2.2	Field Documentation . . . . .	3
3.3	plane Struct Reference . . . . .	3
3.3.1	Detailed Description . . . . .	3
3.3.2	Field Documentation . . . . .	3
<b>4</b>	<b>File Documentation</b>	<b>3</b>
4.1	airport-sim.c File Reference . . . . .	3
4.1.1	Detailed Description . . . . .	4
4.1.2	Function Documentation . . . . .	4
4.2	airport.c File Reference . . . . .	5
4.2.1	Detailed Description . . . . .	6
4.2.2	Function Documentation . . . . .	6
4.3	airport.h File Reference . . . . .	8
4.3.1	Detailed Description . . . . .	9
4.3.2	Function Documentation . . . . .	9
4.4	bay.c File Reference . . . . .	10
4.4.1	Detailed Description . . . . .	11
4.4.2	Function Documentation . . . . .	11
4.5	bay.h File Reference . . . . .	12
4.5.1	Detailed Description . . . . .	12
4.5.2	Function Documentation . . . . .	13
4.6	plane.c File Reference . . . . .	14
4.6.1	Detailed Description . . . . .	14
4.6.2	Function Documentation . . . . .	14
4.7	plane.h File Reference . . . . .	15
4.7.1	Detailed Description . . . . .	15
4.7.2	Function Documentation . . . . .	16

4.8	<a href="#">tools.c File Reference</a>	17
4.8.1	<a href="#">Detailed Description</a>	17
4.8.2	<a href="#">Function Documentation</a>	18
4.9	<a href="#">tools.h File Reference</a>	18
4.9.1	<a href="#">Detailed Description</a>	19
4.9.2	<a href="#">Function Documentation</a>	19

## [Index](#) 20

### 1 Data Structure Index

#### 1.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">airport</a>	Airport structure for representing an instance of an airport	2
<a href="#">bay</a>	Bay structure for representing an instance of a bay	2
<a href="#">plane</a>	Plane structure for representing an instance of a plane	3

### 2 File Index

#### 2.1 File List

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

<a href="#">airport-sim.c</a>	Main program file with <a href="#">main()</a> entry point	3
<a href="#">airport.c</a>	File containing public methods for airport class	5
<a href="#">airport.h</a>	Header containing the public accessible airport methods	8
<a href="#">bay.c</a>	File containing public methods for bay class	10
<a href="#">bay.h</a>	Header containing the public accessible bay methods	12
<a href="#">plane.c</a>	File containing the plane structure and its member methods	14
<a href="#">plane.h</a>	Header containing the public accessible plane methods	15

<a href="#">tools.c</a>	
File containing functions methods with general purpose	17
<a href="#">tools.h</a>	
Header file containing functions methods with general purpose	18

## 3 Data Structure Documentation

### 3.1 airport Struct Reference

Airport structure for representing an instance of an airport.

#### Data Fields

- char \* [name](#)
- bay \*\* [bays](#)
- pthread\_mutex\_t [runway](#)
- sem\_t [empty](#)
- sem\_t [full](#)

#### 3.1.1 Detailed Description

Airport structure for representing an instance of an airport.

#### 3.1.2 Field Documentation

##### 3.1.2.1 bay\*\* airport::bays

Bays in which planes can be parked. Has length NUM\_BAYS.

##### 3.1.2.2 sem\_t airport::empty

Semaphore to block on empty bay.

##### 3.1.2.3 sem\_t airport::full

Semaphore to block on full bay.

##### 3.1.2.4 char\* airport::name

Name of the airport.

##### 3.1.2.5 pthread\_mutex\_t airport::runway

A virtual runway, to prevent multiple take-off/landing operations at the same time.

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

- [airport.c](#)

### 3.2 bay Struct Reference

Bay structure for representing an instance of a bay.

#### Data Fields

- [plane \\* plane](#)
- [time\\_t parking\\_time](#)

#### 3.2.1 Detailed Description

Bay structure for representing an instance of a bay.

#### 3.2.2 Field Documentation

##### 3.2.2.1 `time_t bay::parking_time`

Time, the plane was parked or unparked.

##### 3.2.2.2 `plane* bay::plane`

Plane parked in bay, null if there is none.

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

- [bay.c](#)

### 3.3 plane Struct Reference

Plane structure for representing an instance of a plane.

#### Data Fields

- `char * name`

#### 3.3.1 Detailed Description

Plane structure for representing an instance of a plane.

#### 3.3.2 Field Documentation

##### 3.3.2.1 `char* plane::name`

Name of the plane.

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

- [plane.c](#)

## 4 File Documentation

### 4.1 airport-sim.c File Reference

Main program file with [main\(\)](#) entry point.

```
#include <pthread.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>
#include "airport.h"
#include "tools.h"
```

## Functions

- void `usage` (char \*pname)  
*Prints the help for airport-sim to the console.*
- void `print_banner` ()  
*Prints the startup banner of the airport to the console.*
- void `monitor_thread_func` ()  
*Monitor thread.*
- void `landing_thread_func` (int prob)  
*Landing thread.*
- void `takeoff_thread_func` (int prob)  
*Takeoff thread.*
- int `main` (int argc, char \*\*argv)  
*Main entry point of airport-sim.*

## Variables

- bool `airport_exit` = false  
*This is set to exit when the application should exit gracefully.*
- `airport` \* `ap`  
*Representation of the airport for the simulation.*

### 4.1.1 Detailed Description

Main program file with `main()` entry point.

#### Author

Lukas Elsner

#### Date

01-09-2014 The airport-sim application.

### 4.1.2 Function Documentation

#### 4.1.2.1 void `landing_thread_func` ( int *prob* )

Landing thread.

## Parameters

<i>int</i>	Landing probability
------------	---------------------

The landing thread lands a plane on the airport with the given probability.

4.1.2.2 `int main ( int argc, char ** argv )`

Main entry point of airport-sim.

## Parameters

<i>int</i>	Number of arguments
<i>char**</i>	Pointer to array of arguments

## Returns

Exit code of airport-sim

4.1.2.3 `void monitor_thread_func ( )`

Monitor thread.

The monitor thread interacts with the user while the airport-simulation is running. To print the state of the airport, the user can press 'p' or 'P'. To exit the application, the user can press 'q' or 'Q'.

4.1.2.4 `void takeoff_thread_func ( int prob )`

Takeoff thread.

## Parameters

<i>int</i>	Take-off probability
------------	----------------------

The take-off thread takes off a plane of the airport with the given probability.

4.1.2.5 `void usage ( char * pname )`

Prints the help for airport-sim to the console.

## Parameters

<i>char*</i>	Program name
--------------	--------------

## 4.2 airport.c File Reference

File containing public methods for airport class.

```
#include <stdlib.h>
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include <string.h>
#include <stdbool.h>
#include "airport.h"
#include "tools.h"
#include "bay.h"
```

## Data Structures

- struct [airport](#)

*Airport structure for representing an instance of an airport.*

## Macros

- `#define NUM_BAYS 10`  
*Number of parking slots the airport supplies.*

## Functions

- `bool airport_is_empty (airport *ap)`  
*Checks if a airport is empty.*
- `bool airport_is_full (airport *ap)`  
*Checks if a airport is full.*
- `int get_random_free_bay_nr (airport *ap)`  
*Gets a random free bay number.*
- `int get_random_alloc_bay_nr (airport *ap)`  
*Gets a random occupied bay number.*
- `airport * airport_init (char *name)`  
*constructor for airport*
- `void airport_land_plane (airport *ap)`  
*Lets a plane land on the airport.*
- `void airport_takeoff_plane (airport *ap)`  
*Lets a plane take off from the airport.*
- `char * airport_to_string (airport *ap)`  
*Method for getting a string representation of the current airport state.*
- `void airport_destroy (airport *ap)`  
*Destructor for airport.*

### 4.2.1 Detailed Description

File containing public methods for airport class.

#### Author

Lukas Elsner

#### Date

01-09-2014 The airport is the main data structure of this application. It provides the parking bays and thread safe take-off and landing functionality.

### 4.2.2 Function Documentation

#### 4.2.2.1 `void airport_destroy ( airport * )`

Destructor for airport.

##### Parameters

<i>airport*</i>	Pointer to structure to be freed
-----------------	----------------------------------

#### 4.2.2.2 `airport* airport_init ( char * )`

constructor for airport



## Parameters

<i>char*</i>	The name of the airport
--------------	-------------------------

## Returns

A pointer to the airport structure, representing the created object

After using this structure, it must be freed with [airport\\_destroy\(airport \\*\)](#)

#### 4.2.2.3 bool airport\_is\_empty ( airport \* ap )

Checks if a airport is empty.

## Parameters

<i>airport*</i>	Pointer to structure to work on
-----------------	---------------------------------

## Returns

True, if the airport is empty, false otherwise

This method queries the empty-semaphore to check if the given airport is empty.

#### 4.2.2.4 bool airport\_is\_full ( airport \* ap )

Checks if a airport is full.

## Parameters

<i>airport*</i>	Pointer to structure to work on
-----------------	---------------------------------

## Returns

True, if the airport is full, false otherwise

This method queries the full-semaphore to check if the given airport is full.

#### 4.2.2.5 void airport\_land\_plane ( airport \* )

Lets a plane land on the airport.

## Parameters

<i>airport*</i>	Pointer to structure to work on
-----------------	---------------------------------

This method creates a plane and parks it in a randomly chosen empty parking bay. It is thread safe. When the airport is full, it blocks for maximum of 500ms to get a free slot. After that it returns without any side effects.

#### 4.2.2.6 void airport\_takeoff\_plane ( airport \* )

Lets a plane take off from the airport.

## Parameters

<i>airport*</i>	Pointer to structure to work on
-----------------	---------------------------------

This method chooses a random plane from the parking bay to take off. It is thread safe. When the airport is empty, it blocks for maximum of 500ms to get a plane. After that it returns without any side effects.

#### 4.2.2.7 char\* airport\_to\_string ( airport \* )

Method for getting a string representation of the current airport state.

**Parameters**

<i>airport*</i>	Pointer to structure to work on
-----------------	---------------------------------

**Returns**

A pointer to a string representation of passed structure. Must be freed by caller.

This is a thread safe call to return the current airport state in a human readable form. We need this lock because of a very unlikely race condition, where a plane is taking off while this buffer is filled, causing a SIGSEGV while accessing the plane's name after taking off!

**4.2.2.8 int get\_random\_alloc\_bay\_nr ( airport \* ap )**

Gets a random occupied bay number.

**Parameters**

<i>airport*</i>	Pointer to structure to work on
-----------------	---------------------------------

**Returns**

A random occupied bay number

The caller has to make sure, that there is a occupied bay existing. Otherwise, this function never returns.

**4.2.2.9 int get\_random\_free\_bay\_nr ( airport \* ap )**

Gets a random free bay number.

**Parameters**

<i>airport*</i>	Pointer to structure to work on
-----------------	---------------------------------

**Returns**

A random free bay number

The caller has to make sure, that there is a free bay existing. Otherwise, this function never returns.

**4.3 airport.h File Reference**

Header containing the public accessible airport methods.

**Typedefs**

- typedef struct [airport](#) [airport](#)  
*Forward declaration for airport.*

**Functions**

- [airport \\*](#) [airport\\_init](#) (char \*)  
*constructor for airport*
- void [airport\\_land\\_plane](#) (airport \*)  
*Lets a plane land on the airport.*
- void [airport\\_takeoff\\_plane](#) (airport \*)  
*Lets a plane take off from the airport.*

- char \* [airport\\_to\\_string](#) (airport \*)  
*Method for getting a string representation of the current airport state.*
- void [airport\\_destroy](#) (airport \*)  
*Destructor for airport.*

#### 4.3.1 Detailed Description

Header containing the public accessible airport methods.

##### Author

Lukas Elsner

##### Date

01-09-2014 The airport is the main data structure of this application. It provides the parking bays and thread safe take-off and landing functionality.

#### 4.3.2 Function Documentation

##### 4.3.2.1 void [airport\\_destroy](#) ( airport \* )

Destructor for airport.

##### Parameters

<i>airport*</i>	Pointer to structure to be freed
-----------------	----------------------------------

##### 4.3.2.2 [airport\\*](#) [airport\\_init](#) ( char \* )

constructor for airport

##### Parameters

<i>char*</i>	The name of the airport
--------------	-------------------------

##### Returns

A pointer to the airport structure, representing the created object

After using this structure, it must be freed with [airport\\_destroy](#)(airport \*)

##### 4.3.2.3 void [airport\\_land\\_plane](#) ( airport \* )

Lets a plane land on the airport.

##### Parameters

<i>airport*</i>	Pointer to structure to work on
-----------------	---------------------------------

This method creates a plane and parks it in a randomly chosen empty parking bay. It is thread safe. When the airport is full, it blocks for maximum of 500ms to get a free slot. After that it returns without any side effects.

##### 4.3.2.4 void [airport\\_takeoff\\_plane](#) ( airport \* )

Lets a plane take off from the airport.

**Parameters**

<i>airport*</i>	Pointer to structure to work on
-----------------	---------------------------------

This method chooses a random plane from the parking bay to take off. It is thread safe. When the airport is empty, it blocks for maximum of 500ms to get a plane. After that it returns without any side effects.

**4.3.2.5 char\* airport\_to\_string ( airport \* )**

Method for getting a string representation of the current airport state.

**Parameters**

<i>airport*</i>	Pointer to structure to work on
-----------------	---------------------------------

**Returns**

A pointer to a string representation of passed structure. Must be freed by caller.

This is a thread safe call to return the current airport state in a human readable form. We need this lock because of a very unlikely race condition, where a plane is taking off while this buffer is filled, causing a SIGSEGV while accessing the plane's name after taking off!

**4.4 bay.c File Reference**

File containing public methods for bay class.

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include "tools.h"
#include "bay.h"
```

**Data Structures**

- struct [bay](#)

*Bay structure for representing an instance of a bay.*

**Functions**

- [bay \\* bay\\_init \(\)](#)

*constructor for bay*

- [time\\_t bay\\_get\\_occupation\\_time \(bay \\*b\)](#)

*Method for getting the time this bay has been occupied.*

- [void bay\\_park\\_plane \(bay \\*b, plane \\*p\)](#)

*Parks a plane in the given bay.*

- [plane \\* bay\\_unpark\\_plane \(bay \\*b\)](#)

*Unparks a plane from the given bay.*

- [plane \\* bay\\_get\\_plane \(bay \\*b\)](#)

*Gets the current parked plane.*

- [void bay\\_destroy \(bay \\*b\)](#)

*Destructor for bay.*

#### 4.4.1 Detailed Description

File containing public methods for bay class.

##### Author

Lukas Elsner

##### Date

01-09-2014 The airport .....

#### 4.4.2 Function Documentation

##### 4.4.2.1 void bay\_destroy ( bay \* )

Destructor for bay.

##### Parameters

<i>bay*</i>	Pointer to structure to be freed
-------------	----------------------------------

##### 4.4.2.2 time\_t bay\_get\_occupation\_time ( bay \* b )

Method for getting the time this bay has been occupied.

##### Parameters

<i>bay*</i>	Pointer to structure to work on
-------------	---------------------------------

##### Returns

The time since the current parking plane has landed if there is a parking plane currently, or the overall parking time, if plane has already taken off (bay->plane == NULL)

##### 4.4.2.3 plane\* bay\_get\_plane ( bay \* b )

Gets the current parked plane.

##### Parameters

<i>bay*</i>	Pointer to structure to work on
-------------	---------------------------------

##### Returns

The plane which is parked in the bay, or NULL if there is none.

##### 4.4.2.4 bay\* bay\_init ( )

constructor for bay

##### Returns

A pointer to the bay structure, representing the created object

After using this structure, it must be freed with [bay\\_destroy\(bay \\*\)](#)

##### 4.4.2.5 void bay\_park\_plane ( bay \* b, plane \* p )

Parks a plane in the given bay.

**Parameters**

<i>bay*</i>	Pointer to structure to work on
<i>plane*</i>	The plane to park

If the plane is parked, the current time is preserved, to request the overall parking time with [bay\\_get\\_occupation\\_time\(\)](#).

**4.4.2.6 plane\* bay\_unpark\_plane ( bay \* b )**

Unparks a plane from the given bay.

**Parameters**

<i>bay*</i>	Pointer to structure to work on
-------------	---------------------------------

**Returns**

The plane which was unparked

**4.5 bay.h File Reference**

Header containing the public accessible bay methods.

```
#include "plane.h"
```

**Typedefs**

- typedef struct [bay](#) bay  
*Forward declaration for bay.*

**Functions**

- [bay \\* bay\\_init](#) ()  
*constructor for bay*
- [time\\_t bay\\_get\\_occupation\\_time](#) (bay \*b)  
*Method for getting the time this bay has been occupied.*
- void [bay\\_park\\_plane](#) (bay \*b, plane \*p)  
*Parks a plane in the given bay.*
- [plane \\* bay\\_unpark\\_plane](#) (bay \*b)  
*Unparks a plane from the given bay.*
- [plane \\* bay\\_get\\_plane](#) (bay \*b)  
*Gets the current parked plane.*
- void [bay\\_destroy](#) (bay \*)  
*Destructor for bay.*

**4.5.1 Detailed Description**

Header containing the public accessible bay methods.

**Author**

Lukas Elsner

**Date**

01-09-2014 The airport uses bay for the parking areas, where airplane can be parked.

## 4.5.2 Function Documentation

## 4.5.2.1 void bay\_destroy ( bay \* )

Destructor for bay.

## Parameters

<i>bay*</i>	Pointer to structure to be freed
-------------	----------------------------------

## 4.5.2.2 time\_t bay\_get\_occupation\_time ( bay \* b )

Method for getting the time this bay has been occupied.

## Parameters

<i>bay*</i>	Pointer to structure to work on
-------------	---------------------------------

## Returns

The time since the current parking plane has landed if there is a parking plane currently, or the overall parking time, if plane has already taken off (bay->plane == NULL)

## 4.5.2.3 plane\* bay\_get\_plane ( bay \* b )

Gets the current parked plane.

## Parameters

<i>bay*</i>	Pointer to structure to work on
-------------	---------------------------------

## Returns

The plane which is parked in the bay, or NULL if there is none.

## 4.5.2.4 bay\* bay\_init ( )

constructor for bay

## Returns

A pointer to the bay structure, representing the created object

After using this structure, it must be freed with [bay\\_destroy\(bay \\*\)](#)

## 4.5.2.5 void bay\_park\_plane ( bay \* b, plane \* p )

Parks a plane in the given bay.

## Parameters

<i>bay*</i>	Pointer to structure to work on
<i>plane*</i>	The plane to park

If the plane is parked, the current time is preserved, to request the overall parking time with [bay\\_get\\_occupation\\_time\(\)](#).

## 4.5.2.6 plane\* bay\_unpark\_plane ( bay \* b )

Unparks a plane from the given bay.

**Parameters**

<i>bay*</i>	Pointer to structure to work on
-------------	---------------------------------

**Returns**

The plane which was unparked

**4.6 plane.c File Reference**

File containing the plane structure and its member methods.

```
#include <time.h>
#include <stdlib.h>
#include "tools.h"
#include "plane.h"
```

**Data Structures**

- struct [plane](#)  
*Plane structure for representing an instance of a plane.*

**Functions**

- [plane \\* plane\\_init](#) ()  
*constructor for plane*
- [char \\* plane\\_get\\_name](#) ([plane \\*p](#))  
*Method for getting the name of a plane structure.*
- [void plane\\_destroy](#) ([plane \\*p](#))  
*Destructor for plane.*

**4.6.1 Detailed Description**

File containing the plane structure and its member methods.

**Author**

Lukas Elsner

**Date**

01-09-2014 A plane is generated by the landing thread and stored in a bay of the airport.

**4.6.2 Function Documentation****4.6.2.1 void plane\_destroy ( plane \* )**

Destructor for plane.

**Parameters**



<i>plane*</i>	Pointer to structure to be freed
---------------	----------------------------------

#### 4.6.2.2 char\* plane\_get\_name ( plane \* )

Method for getting the name of a plane structure.

Parameters

<i>plane*</i>	Pointer to structure to work on
---------------	---------------------------------

Returns

A pointer to a string representation of passed structure. Must be freed by caller.

#### 4.6.2.3 plane\* plane\_init ( )

constructor for plane

Returns

A pointer to the plane structure, representing the created object

After using this structure, it must be freed with [plane\\_destroy\(plane \\*\)](#)

## 4.7 plane.h File Reference

Header containing the public accessible plane methods.

Typedefs

- typedef struct [plane](#) [plane](#)  
*Forward declaration for plane.*

Functions

- [plane \\*](#) [plane\\_init](#) ()  
*constructor for plane*
- char \* [plane\\_get\\_name](#) ([plane \\*](#))  
*Method for getting the name of a plane structure.*
- void [plane\\_destroy](#) ([plane \\*](#))  
*Destructor for plane.*

### 4.7.1 Detailed Description

Header containing the public accessible plane methods.

Author

Lukas Elsner

Date

01-09-2014 A plane is generated by the landing thread and stored in a bay of the airport.

#### 4.7.2 Function Documentation

##### 4.7.2.1 void plane\_destroy ( plane \* )

Destructor for plane.

## Parameters

<i>plane*</i>	Pointer to structure to be freed
---------------	----------------------------------

## 4.7.2.2 char\* plane\_get\_name ( plane \* )

Method for getting the name of a plane structure.

## Parameters

<i>plane*</i>	Pointer to structure to work on
---------------	---------------------------------

## Returns

A pointer to a string representation of passed structure. Must be freed by caller.

## 4.7.2.3 plane\* plane\_init ( )

constructor for plane

## Returns

A pointer to the plane structure, representing the created object

After using this structure, it must be freed with [plane\\_destroy\(plane \\*\)](#)

## 4.8 tools.c File Reference

File containing functions methods with general purpose.

```
#include <time.h>
#include <sys/time.h>
#include <stdlib.h>
#include "tools.h"
```

## Functions

- char \* [generate\\_rand](#) (int num\_l, int num\_n)  
*Generates a random name based on num\_l letters followed by num\_n numbers.*
- time\_t [current\\_timestamp](#) ()  
*Get the current timestamp.*
- bool [prob\\_bool](#) (int prob)  
*Boolean generator based on a given probability.*
- void [msleep](#) (long long m)  
*Sleep function with milliseconds granularity.*

## 4.8.1 Detailed Description

File containing functions methods with general purpose.

## Author

Lukas Elsner

## Date

02-09-2014

## 4.8.2 Function Documentation

### 4.8.2.1 `time_t current_timestamp ( )`

Get the current timestamp.

#### Returns

The current timestamp in milliseconds.

### 4.8.2.2 `char* generate_rand ( int num_l, int num_n )`

Generates a random name based on num\_l letters followed by num\_n numbers.

#### Returns

A pointer to a the generated string. Has to be freed by caller.

### 4.8.2.3 `void msleep ( long long )`

Sleep function with milliseconds granularity.

#### Parameters

<i>long-long</i>	The time in milliseconds
------------------	--------------------------

The current thread will block for the given time in milliseconds. This is implemented with help of the POSIX nanosleep function and a timespec struct.

### 4.8.2.4 `bool prob_bool ( int )`

Boolean generator based on a given probability.

#### Parameters

<i>int</i>	A probability value between 0 and 100.
------------	--

#### Returns

A randomly generated boolean value

Values below 0 will be handled as 0, values above 100 will be handled as 100.

## 4.9 tools.h File Reference

Header file containing functions methods with general purpose.

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

#### Functions

- `char * generate_rand (int num_l, int num_n)`  
*Generates a random name based on num\_l letters followed by num\_n numbers.*
- `time_t current_timestamp ()`  
*Get the current timestamp.*
- `bool prob_bool (int)`  
*Boolean generator based on a given probability.*
- `void msleep (long long)`  
*Sleep function with milliseconds granularity.*

#### 4.9.1 Detailed Description

Header file containing functions methods with general purpose.

##### Author

Lukas Elsner

##### Date

02-09-2014

#### 4.9.2 Function Documentation

##### 4.9.2.1 `time_t current_timestamp ( )`

Get the current timestamp.

##### Returns

The current timestamp in milliseconds.

##### 4.9.2.2 `char* generate_rand ( int num_l, int num_n )`

Generates a random name based on num\_l letters followed by num\_n numbers.

##### Returns

A pointer to a the generated string. Has to be freed by caller.

##### 4.9.2.3 `void msleep ( long long )`

Sleep function with milliseconds granularity.

##### Parameters

<i>long-long</i>	The time in milliseconds
------------------	--------------------------

The current thread will block for the given time in milliseconds. This is implemented with help of the POSIX nanosleep function and a timespec struct.

##### 4.9.2.4 `bool prob_bool ( int )`

Boolean generator based on a given probability.

##### Parameters

<i>int</i>	A probability value between 0 and 100.
------------	--

##### Returns

A randomly generated boolean value

Values below 0 will be handled as 0, values above 100 will be handled as 100.

## Index

- airport, 2
  - bays, 2
  - empty, 2
  - full, 2
  - name, 2
  - runway, 2
- airport-sim.c, 3
  - landing\_thread\_func, 4
  - main, 5
  - monitor\_thread\_func, 5
  - takeoff\_thread\_func, 5
  - usage, 5
- airport.c, 5
  - airport\_destroy, 6
  - airport\_init, 6
  - airport\_is\_empty, 7
  - airport\_is\_full, 7
  - airport\_land\_plane, 7
  - airport\_takeoff\_plane, 7
  - airport\_to\_string, 7
  - get\_random\_alloc\_bay\_nr, 8
  - get\_random\_free\_bay\_nr, 8
- airport.h, 8
  - airport\_destroy, 9
  - airport\_init, 9
  - airport\_land\_plane, 9
  - airport\_takeoff\_plane, 9
  - airport\_to\_string, 10
- airport\_destroy
  - airport.c, 6
  - airport.h, 9
- airport\_init
  - airport.c, 6
  - airport.h, 9
- airport\_is\_empty
  - airport.c, 7
- airport\_is\_full
  - airport.c, 7
- airport\_land\_plane
  - airport.c, 7
  - airport.h, 9
- airport\_takeoff\_plane
  - airport.c, 7
  - airport.h, 9
- airport\_to\_string
  - airport.c, 7
  - airport.h, 10
- bay, 2
  - parking\_time, 3
  - plane, 3
- bay.c, 10
  - bay\_destroy, 11
  - bay\_get\_occupation\_time, 11
  - bay\_get\_plane, 11
- bay\_init, 11
- bay\_park\_plane, 11
- bay\_unpark\_plane, 12
- bay.h, 12
  - bay\_destroy, 13
  - bay\_get\_occupation\_time, 13
  - bay\_get\_plane, 13
  - bay\_init, 13
  - bay\_park\_plane, 13
  - bay\_unpark\_plane, 13
- bay\_destroy
  - bay.c, 11
  - bay.h, 13
- bay\_get\_occupation\_time
  - bay.c, 11
  - bay.h, 13
- bay\_get\_plane
  - bay.c, 11
  - bay.h, 13
- bay\_init
  - bay.c, 11
  - bay.h, 13
- bay\_park\_plane
  - bay.c, 11
  - bay.h, 13
- bay\_unpark\_plane
  - bay.c, 12
  - bay.h, 13
- bays
  - airport, 2
- current\_timestamp
  - tools.c, 18
  - tools.h, 19
- empty
  - airport, 2
- full
  - airport, 2
- generate\_rand
  - tools.c, 18
  - tools.h, 19
- get\_random\_alloc\_bay\_nr
  - airport.c, 8
- get\_random\_free\_bay\_nr
  - airport.c, 8
- landing\_thread\_func
  - airport-sim.c, 4
- main
  - airport-sim.c, 5
  - monitor\_thread\_func
    - airport-sim.c, 5
- msleep

- tools.c, [18](#)
- tools.h, [19](#)
- name
  - airport, [2](#)
  - plane, [3](#)
- parking\_time
  - bay, [3](#)
- plane, [3](#)
  - bay, [3](#)
  - name, [3](#)
- plane.c, [14](#)
  - plane\_destroy, [14](#)
  - plane\_get\_name, [15](#)
  - plane\_init, [15](#)
- plane.h, [15](#)
  - plane\_destroy, [16](#)
  - plane\_get\_name, [17](#)
  - plane\_init, [17](#)
- plane\_destroy
  - plane.c, [14](#)
  - plane.h, [16](#)
- plane\_get\_name
  - plane.c, [15](#)
  - plane.h, [17](#)
- plane\_init
  - plane.c, [15](#)
  - plane.h, [17](#)
- prob\_bool
  - tools.c, [18](#)
  - tools.h, [19](#)
- runway
  - airport, [2](#)
- takeoff\_thread\_func
  - airport-sim.c, [5](#)
- tools.c, [17](#)
  - current\_timestamp, [18](#)
  - generate\_rand, [18](#)
  - msleep, [18](#)
  - prob\_bool, [18](#)
- tools.h, [18](#)
  - current\_timestamp, [19](#)
  - generate\_rand, [19](#)
  - msleep, [19](#)
  - prob\_bool, [19](#)
- usage
  - airport-sim.c, [5](#)