Threading Class 1.0

Generated by Doxygen 1.8.7

Sun Jul 13 2014 22:44:34

## **Contents**

1	Clas	s Index			1								
	1.1	Class I	₋ist		1								
2	Clas	ss Documentation 3											
	2.1	Mailbo	Color of the section of the secti										
		2.1.1	Construct	or & Destructor Documentation	3								
			2.1.1.1	Mailbox	4								
			2.1.1.2	Mailbox	4								
		2.1.2	Member F	Function Documentation	4								
			2.1.2.1	GetMessage	4								
			2.1.2.2	IsEmpty	4								
			2.1.2.3	IsFull	4								
			2.1.2.4	PutMessage	4								
			2.1.2.5	SetSize	4								
		2.1.3	Member E	Member Data Documentation									
			2.1.3.1	m_mailBox	5								
			2.1.3.2	m_maxSize	5								
			2.1.3.3	mutex	5								
	2.2	Mutex	Class Refe	rence	5								
		2.2.1	Member F	Function Documentation	5								
			2.2.1.1	Lock	5								
			2.2.1.2	Unlock	5								
	2.3	Thread	l Class Ref	erence	6								
		2.3.1	Member F	Function Documentation	6								
			2.3.1.1	GetThreadId	6								
			2.3.1.2	SetParam	6								
			2.3.1.3	SetThreadFunction	6								
			2.3.1.4	Start	6								
			2.3.1.5	Stop	7								
		2.3.2	Member E	Data Documentation	7								
			2321	mp. function	7								

iv		COI	NTENTS
	2.3.2.2	mp_param	7
Index			8

# Chapter 1

# **Class Index**

1	1	Class	Liet
		22KL)	ı ıçı

Here are the classes,	structs, i	unions a	nd interfaces	with bri	ef descriptions:

Mailbox	<	<b>T</b> :	>	 								 						 				 			3
Mutex				 								 						 				 			5
Thread				 							 	 						 				 			6

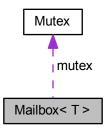
2 Class Index

## **Chapter 2**

## **Class Documentation**

## 2.1 Mailbox < T > Class Template Reference

Collaboration diagram for Mailbox< T >:



## **Public Member Functions**

- Mailbox ()
- Mailbox (int size)
- int SetSize (int size)
- int PutMessage (T message)
- bool IsEmpty ()
- bool IsFull ()
- T GetMessage ()

## **Protected Attributes**

- std::vector< T > m\_mailBox
- int m\_maxSize
- Mutex mutex

## 2.1.1 Constructor & Destructor Documentation

4 Class Documentation

```
2.1.1.1 template < class T > Mailbox < T >::Mailbox ( )
```

Create the mailbox with no size limet

2.1.1.2 template < class T > Mailbox < T >::Mailbox ( int size )

**Parameters** 

size The maximum size of the mailbox 0 for an unlimmeted size

```
2.1.2 Member Function Documentation
```

```
2.1.2.1 template < class T > T Mailbox < T >::GetMessage ( )
```

Gives back the last message run IsEmpty() first!

Returns

the last message

Warning

if there is no massage in the buffer it returns random vallues

2.1.2.2 template < class T > bool Mailbox < T >::IsEmpty ( )

Checks if the mailbox is empty

Returns

true if empty else false

2.1.2.3 template < class T > bool Mailbox < T >::IsFull ( )

Checks if the mailbox is full

Returns

true if full else false

2.1.2.4 template < class T > int Mailbox < T >::PutMessage ( T message )

Add a message to the mailbox its smart to run IsFull() first

**Parameters** 

message | the message that has to be mailt to the other thread

Returns

0 on success else >0

2.1.2.5 template < class T > int Mailbox < T >::SetSize (int size)

Set the size of the mailbox

2.2 Mutex Class Reference 5

#### **Parameters**

size the maximum size of the mailbox

Returns

0 on success else >0

#### 2.1.3 Member Data Documentation

2.1.3.1 template < class T > std::vector < T > Mailbox < T >::m\_mailBox [protected]

We use a vector to store the messages

**2.1.3.2** template < class T > int Mailbox < T >::m\_maxSize [protected]

The maximum size of the mailbox

2.1.3.3 template < class T > Mutex Mailbox < T >::mutex [protected]

The mutex to make it threadsafe

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

· mailbox.h

## 2.2 Mutex Class Reference

**Public Member Functions** 

- int Lock ()
- int Unlock ()

### 2.2.1 Member Function Documentation

2.2.1.1 int Mutex::Lock ( )

Lock the mutex

Returns

0 on success else >0

2.2.1.2 int Mutex::Unlock ( )

Lock the mutex

Returns

0 on success else >0

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

· mutex.h

6 Class Documentation

## 2.3 Thread Class Reference

```
Public Member Functions
```

• int GetThreadId ()

```
int SetThreadFunction (int(*p_function)(void *))
int SetParam (void *p_param)
int Start ()
int Stop ()
```

### **Protected Attributes**

```
void * mp_paramint(* mp_function )(void *)
```

### 2.3.1 Member Function Documentation

```
2.3.1.1 int Thread::GetThreadId ( )
```

Warning

this function is not implemented

Returns

The thread ID

```
2.3.1.2 int Thread::SetParam (void * p_param)
```

Set the parameters that have to be send to the thread

Returns

0 on success else >07-2014

2.3.1.3 int Thread::SetThreadFunction (  $int(*)(void *) p\_function$  )

Set the function that has to run and tell if it has to run in a while (true) loop

**Parameters** 

*p\_function* the pointer to the function that has to run as a thread format (int FunctionName (void\*))

Returns

0 on success else >0

2.3.1.4 int Thread::Start ( )

Start the thread

Returns

0 on success else >0

2.3 Thread Class Reference 7

```
2.3.1.5 int Thread::Stop ( )
```

Stop the thread

Returns

0 on success else >0

### 2.3.2 Member Data Documentation

```
2.3.2.1 int(* Thread::mp_function)(void *) [protected]
```

A pointer to the function that has to run as a thread

```
2.3.2.2 void* Thread::mp_param [protected]
```

A pointer to the param that has to be send to the thread  $% \left( x\right) =\left( x\right) +\left( x\right)$ 

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

- thread.h
- thread\_general.cpp

# Index

Lock	
M	lutex, 5
Mailbo	X
M	lailbox, 3, 4
m	nutex, 5
Mailbo	x < T >, 3
Mutex	, 5
L	ock, 5
U	Inlock, 5
mutex	
N	lailbox, 5
Start	
Т	hread, 6
Stop	
Т	hread, 6
Thread	d, 6
S	tart, 6
S	top, 6
Unlock	(
M	lutex, 5