

LWT Threads

1.0

Generated by Doxygen 1.8.8

Tue Mar 10 2015 01:53:40

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	lwt_chan_t Struct Reference	5
3.1.1	Field Documentation	5
3.1.1.1	blocked_receiver	5
3.1.1.2	blocked_senders_head	5
3.1.1.3	blocked_senders_tail	5
3.1.1.4	buffer	5
3.1.1.5	next_sibling	6
3.1.1.6	previous_sibling	6
3.1.1.7	receiver	6
3.1.1.8	senders	6
3.2	lwt_t Struct Reference	6
3.2.1	Detailed Description	7
3.2.2	Field Documentation	7
3.2.2.1	args	7
3.2.2.2	children	7
3.2.2.3	id	7
3.2.2.4	info	7
3.2.2.5	max_addr_thread_stack	7
3.2.2.6	min_addr_thread_stack	7
3.2.2.7	next_blocked_sender	7
3.2.2.8	next_current	7
3.2.2.9	next_runnable	7

3.2.2.10	next_sender	7
3.2.2.11	next_sibling	8
3.2.2.12	parent	8
3.2.2.13	previous_blocked_sender	8
3.2.2.14	previous_current	8
3.2.2.15	previous_runnable	8
3.2.2.16	previous_sender	8
3.2.2.17	previous_sibling	8
3.2.2.18	receiving_channels	8
3.2.2.19	return_value	8
3.2.2.20	start_routine	8
3.2.2.21	thread_sp	8
3.3	msort_args Struct Reference	9
3.3.1	Detailed Description	9
3.3.2	Field Documentation	9
3.3.2.1	begin_index	9
3.3.2.2	data	9
3.3.2.3	end_index	9
3.3.2.4	swap	9
4	File Documentation	11
4.1	lwt.c File Reference	11
4.1.1	Macro Definition Documentation	12
4.1.1.1	DEFAULT_ID	12
4.1.1.2	INIT_ID	12
4.1.2	Function Documentation	12
4.1.2.1	__attribute__	12
4.1.2.2	__attribute__	13
4.1.2.3	__cleanup_joined_thread	13
4.1.2.4	__init_lwt	13
4.1.2.5	__init_lwt_main	13
4.1.2.6	__lwt_dispatch	13
4.1.2.7	__lwt_schedule	13
4.1.2.8	__lwt_stack_get	13
4.1.2.9	__lwt_stack_return	13
4.1.2.10	__lwt_trampoline	14
4.1.2.11	lwt_chan	14

4.1.2.12	lwt_chan_deref	14
4.1.2.13	lwt_create	14
4.1.2.14	lwt_create_chan	14
4.1.2.15	lwt_current	15
4.1.2.16	lwt_die	15
4.1.2.17	lwt_id	15
4.1.2.18	lwt_info	15
4.1.2.19	lwt_join	15
4.1.2.20	lwt_rcv	16
4.1.2.21	lwt_rcv_chan	16
4.1.2.22	lwt_snd	16
4.1.2.23	lwt_snd_chan	16
4.1.2.24	lwt_yield	16
4.2	lwt.h File Reference	17
4.2.1	Macro Definition Documentation	18
4.2.1.1	LWT_NULL	18
4.2.1.2	LWT_YIELD_NO_LWT_TO_YIELD	18
4.2.1.3	NUM_PAGES	18
4.2.1.4	PAGE_SIZE	18
4.2.1.5	STACK_SIZE	18
4.2.2	Typedef Documentation	18
4.2.2.1	lwt_chan_fn_t	18
4.2.2.2	lwt_fnt_t	18
4.2.3	Enumeration Type Documentation	19
4.2.3.1	lwt_info_t	19
4.2.4	Function Documentation	19
4.2.4.1	lwt_chan	19
4.2.4.2	lwt_chan_deref	19
4.2.4.3	lwt_create	19
4.2.4.4	lwt_create_chan	19
4.2.4.5	lwt_current	20
4.2.4.6	lwt_die	20
4.2.4.7	lwt_id	20
4.2.4.8	lwt_info	20
4.2.4.9	lwt_join	20
4.2.4.10	lwt_rcv	21
4.2.4.11	lwt_rcv_chan	21

4.2.4.12	<code>lwt_snd</code>	21
4.2.4.13	<code>lwt_snd_chan</code>	21
4.2.4.14	<code>lwt_yield</code>	22
4.3	main.c File Reference	22
4.3.1	Macro Definition Documentation	23
4.3.1.1	<code>IS_RESET</code>	23
4.3.1.2	<code>ITER</code>	23
4.3.1.3	<code>rdtscll</code>	23
4.3.2	Function Documentation	23
4.3.2.1	<code>fn_bounce</code>	23
4.3.2.2	<code>fn_identity</code>	23
4.3.2.3	<code>fn_join</code>	23
4.3.2.4	<code>fn_nested_joins</code>	23
4.3.2.5	<code>fn_null</code>	23
4.3.2.6	<code>fn_sequence</code>	23
4.3.2.7	<code>main</code>	23
4.3.2.8	<code>test crt join sched</code>	23
4.3.2.9	<code>test_perf</code>	23
4.3.3	Variable Documentation	23
4.3.3.1	<code>curr</code>	23
4.3.3.2	<code>sched</code>	23
4.4	main_chan.c File Reference	23
4.4.1	Macro Definition Documentation	24
4.4.1.1	<code>ITER</code>	24
4.4.1.2	<code>MERGE_SZ</code>	24
4.4.2	Function Documentation	24
4.4.2.1	<code>child_ping</code>	24
4.4.2.2	<code>child_pong</code>	24
4.4.2.3	<code>main</code>	25
4.4.2.4	<code>merge_sort_test</code>	25
4.4.2.5	<code>msort</code>	25
4.4.2.6	<code>ping_pong_test</code>	25
	Index	26

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

lwt_chan_t	5
lwt_t	The Lightweight Thread (LWT) struct	6
msort_args	Struct for passing the args to merge sort around	9

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

lwt.c	11
lwt.h	17
main.c	22
main_chan.c	23

Chapter 3

Data Structure Documentation

3.1 lwt_chan_t Struct Reference

```
#include <lwt.h>
```

Data Fields

- lwt_t [senders](#)
- lwt_t [blocked_senders_head](#)
- lwt_t [blocked_senders_tail](#)
- lwt_t [receiver](#)
- lwt_t [blocked_receiver](#)
- void * [buffer](#)
- lwt_chan_t [previous_sibling](#)
- lwt_chan_t [next_sibling](#)

3.1.1 Field Documentation

3.1.1.1 lwt_t lwt_chan_t::blocked_receiver

The blocked receiver

3.1.1.2 lwt_t lwt_chan_t::blocked_senders_head

The head of the blocked senders

3.1.1.3 lwt_t lwt_chan_t::blocked_senders_tail

The tail of the blocked senders

3.1.1.4 void* lwt_chan_t::buffer

Buffer to be passed to channel

3.1.1.5 `lwt_chan_t lwt_chan_t::next_sibling`

Next sibling channel

3.1.1.6 `lwt_chan_t lwt_chan_t::previous_sibling`

Previous sibling channel

3.1.1.7 `lwt_t lwt_chan_t::receiver`

The receiving thread

3.1.1.8 `lwt_t lwt_chan_t::senders`

The list of senders

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

- [lwt.h](#)

3.2 `lwt_t` Struct Reference

The Lightweight Thread (LWT) struct.

```
#include <lwt.h>
```

Data Fields

- long * [max_addr_thread_stack](#)
- long * [min_addr_thread_stack](#)
- long * [thread_sp](#)
- `lwt_t` [parent](#)
- `lwt_t` [children](#)
- `lwt_t` [previous_sibling](#)
- `lwt_t` [next_sibling](#)
- `lwt_t` [previous_current](#)
- `lwt_t` [next_current](#)
- `lwt_t` [previous_runnable](#)
- `lwt_t` [next_runnable](#)
- `lwt_t` [previous_sender](#)
- `lwt_t` [next_sender](#)
- `lwt_t` [previous_blocked_sender](#)
- `lwt_t` [next_blocked_sender](#)
- `lwt_chan_t` [receiving_channels](#)
- `lwt_fnt_t` [start_routine](#)
- void * [args](#)
- void * [return_value](#)
- `lwt_info_t` [info](#)
- int [id](#)

3.2.1 Detailed Description

The Lightweight Thread (LWT) struct.

3.2.2 Field Documentation

3.2.2.1 void* lwt_t::args

The args for the start_routine

3.2.2.2 lwt_t lwt_t::children

List of children threads

3.2.2.3 int lwt_t::id

The id of the thread

3.2.2.4 lwt_info_t lwt_t::info

The current status of the thread

3.2.2.5 long* lwt_t::max_addr_thread_stack

Pointer to the max address of the stack

3.2.2.6 long* lwt_t::min_addr_thread_stack

Pointer to the min address of the stack; used for malloc and free

3.2.2.7 lwt_t lwt_t::next_blocked_sender

Next blocked sender thread

3.2.2.8 lwt_t lwt_t::next_current

Next current thread

3.2.2.9 lwt_t lwt_t::next_runnable

Next runnable thread

3.2.2.10 lwt_t lwt_t::next_sender

Next sender thread

3.2.2.11 lwt_t lwt_t::next_sibling

Next sibling

3.2.2.12 lwt_t lwt_t::parent

Parent thread

3.2.2.13 lwt_t lwt_t::previous_blocked_sender

Previous blocked sender thread

3.2.2.14 lwt_t lwt_t::previous_current

Previous current thread

3.2.2.15 lwt_t lwt_t::previous_runnable

Previous runnable thread

3.2.2.16 lwt_t lwt_t::previous_sender

Previous sender thread

3.2.2.17 lwt_t lwt_t::previous_sibling

Previous sibling

3.2.2.18 lwt_chan_t lwt_t::receiving_channels

List of receiving channels associated with the thread

3.2.2.19 void* lwt_t::return_value

The return value from the routine

3.2.2.20 lwt_fnt_t lwt_t::start_routine

The start routine for the thread to run

3.2.2.21 long* lwt_t::thread_sp

The current thread stack pointer for the thread

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

- [lwt.h](#)

3.3 msort_args Struct Reference

Struct for passing the args to merge sort around.

Data Fields

- `int * data`
The int array holding randomly generated data.
- `int * swap`
The int array for swap space.
- `int begin_index`
The begin index of the segment.
- `int end_index`
The end index of the segment.

3.3.1 Detailed Description

Struct for passing the args to merge sort around.

3.3.2 Field Documentation

3.3.2.1 `int msort_args::begin_index`

The begin index of the segment.

3.3.2.2 `int* msort_args::data`

The int array holding randomly generated data.

3.3.2.3 `int msort_args::end_index`

The end index of the segment.

3.3.2.4 `int* msort_args::swap`

The int array for swap space.

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

- [main_chan.c](#)

Chapter 4

File Documentation

4.1 lwt.c File Reference

```
#include "lwt.h"
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
```

Macros

- `#define INIT_ID 1`
The initial thread id.
- `#define DEFAULT_ID -1`
The default id provided to threads before actually generating them.

Functions

- `void __lwt_dispatch (lwt_t next, lwt_t current)`
Dispatch function for switching between threads.
- `void __lwt_schedule ()`
Schedules the next_current thread to switch to and dispatches.
- `void __lwt_trampoline ()`
Drops in from being scheduled after the initialized thread is switched to and leaps to the function pointer provided.
- `void * __lwt_stack_get ()`
Allocates the stack for a LWT and returns it.
- `void __lwt_stack_return (void *stack)`
Frees the provided stack.
- `int lwt_id (lwt_t thread)`
Gets the thread id.
- `lwt_t lwt_current ()`
Gets the current thread.
- `int lwt_info (lwt_info_t t)`
Gets the counts of the info.

- void `__init_lwt_main` (lwt_t thread)
Initializes the main thread.
- void `__init_lwt` (lwt_t thread)
Initializes the provided thread.
- void `__cleanup_joined_thread` (lwt_t lwt)
Cleans up the thread on join.
- void * `lwt_join` (lwt_t thread)
Joins the provided thread.
- void `lwt_die` (void *value)
Prepares the current thread to be cleaned up.
- int `lwt_yield` (lwt_t lwt)
Yields to the provided LWT.
- `__attribute__` ((constructor))
Initializes the LWT by wrapping the current thread as a LWT.
- `__attribute__` ((destructor))
Cleans up all remaining threads on exit.
- lwt_t `lwt_create` (lwt_fn_t fn, void *data)
Creates a LWT using the provided function pointer and the data as input for it.
- lwt_chan_t `lwt_chan` (int sz)
Creates the channel on the receiving thread.
- void `lwt_chan_deref` (lwt_chan_t c)
Deallocates the channel only if no threads still have references to the channel.
- int `lwt_snd` (lwt_chan_t c, void *data)
Sends the data over the channel to the receiver.
- void * `lwt_rcv` (lwt_chan_t c)
Receives the data from the channel and returns it.
- int `lwt_snd_chan` (lwt_chan_t c, lwt_chan_t sending)
Sends sending over the channel c.
- lwt_chan_t `lwt_rcv_chan` (lwt_chan_t c)
Receives the data over the channel.
- lwt_t `lwt_create_chan` (lwt_chan_fn_t fn, lwt_chan_t c)
Creates a lwt with the channel as an arg.

4.1.1 Macro Definition Documentation

4.1.1.1 #define DEFAULT_ID -1

The default id provided to threads before actually generating them.

4.1.1.2 #define INIT_ID 1

The initial thread id.

4.1.2 Function Documentation

4.1.2.1 __attribute__ ((constructor))

Initializes the LWT by wrapping the current thread as a LWT.

4.1.2.2 `__attribute__ ((destructor))`

Cleans up all remaining threads on exit.

4.1.2.3 `void __cleanup_joined_thread (lwt_t lwt)`

Cleans up the thread on join.

Parameters

<i>lwt</i>	The thread to join on
------------	-----------------------

4.1.2.4 `void __init_lwt (lwt_t thread)`

Initializes the provided thread.

Parameters

<i>thread</i>	The thread to init
---------------	--------------------

4.1.2.5 `void __init_lwt_main (lwt_t thread)`

Initializes the main thread.

Parameters

<i>thread</i>	The main thread
---------------	-----------------

4.1.2.6 `void __lwt_dispatch (lwt_t next, lwt_t current)`

Dispatch function for switching between threads.

Parameters

<i>next</i>	The next thread to switch to
<i>current</i>	The current thread

4.1.2.7 `void __lwt_schedule (void)`

Schedules the next_current thread to switch to and dispatches.

4.1.2.8 `void * __lwt_stack_get (void)`

Allocates the stack for a LWT and returns it.

4.1.2.9 `void __lwt_stack_return (void * stack)`

Frees the provided stack.

Parameters

<i>stack</i>	The LWT stack to free
--------------	-----------------------

4.1.2.10 `void __lwt_trampoline (void)`

Drops in from being scheduled after the initialized thread is switched to and leaps to the function pointer provided.

4.1.2.11 `lwt_chan_t lwt_chan (int sz)`

Creates the channel on the receiving thread.

Parameters

<i>sz</i>	The size of the buffer
-----------	------------------------

Returns

A pointer to the initialized channel

4.1.2.12 `void lwt_chan_deref (lwt_chan_t c)`

Deallocates the channel only if no threads still have references to the channel.

Parameters

<i>c</i>	The channel to deallocate
----------	---------------------------

4.1.2.13 `lwt_t lwt_create (lwt_fnt_t fn, void * data)`

Creates a LWT using the provided function pointer and the data as input for it.

Parameters

<i>fn</i>	The function pointer to use
<i>data</i>	The data to the function

Returns

A pointer to the initialized LWT

4.1.2.14 `lwt_t lwt_create_chan (lwt_chan_fn_t fn, lwt_chan_t c)`

Creates a lwt with the channel as an arg.

Parameters

<i>fn</i>	The function to use to create the thread
<i>c</i>	The channel to send

4.1.2.15 `lwt_t lwt_current ()`

Gets the current thread.

Returns

The current thread

4.1.2.16 `void lwt_die (void * value)`

Prepares the current thread to be cleaned up.

4.1.2.17 `int lwt_id (lwt_t thread)`

Gets the thread id.

Returns

The id of the thread

4.1.2.18 `int lwt_info (lwt_info_t t)`

Gets the counts of the info.

Parameters

<i>t</i>	The info enum to get the counts
----------	---------------------------------

Returns

The count for the info enum provided

See also

[lwt_info_t](#)

4.1.2.19 `void* lwt_join (lwt_t thread)`

Joins the provided thread.

Parameters

<i>thread</i>	The thread to join on
---------------	-----------------------

4.1.2.20 void* lwt_rcv (lwt_chan_t c)

Receives the data from the channel and returns it.

Parameters

<i>c</i>	The channel to receive from
----------	-----------------------------

Returns

The data from the channel

4.1.2.21 lwt_chan_t lwt_rcv_chan (lwt_chan_t c)

Receives the data over the channel.

Parameters

<i>c</i>	The channel to use for receiving
----------	----------------------------------

Returns

The channel being sent over c

4.1.2.22 int lwt_snd (lwt_chan_t c, void * data)

Sends the data over the channel to the receiver.

Parameters

<i>c</i>	The channel to use for sending
<i>data</i>	The data for sending

Returns

-1 if there is no receiver; 0 if successful

4.1.2.23 int lwt_snd_chan (lwt_chan_t c, lwt_chan_t sending)

Sends sending over the channel c.

Parameters

<i>c</i>	The channel to send sending across
<i>sending</i>	The channel to send

4.1.2.24 int lwt_yield (lwt_t lwt)

Yields to the provided LWT.

Parameters

<i>lwt</i>	The thread to yield to
------------	------------------------

Note

Will just schedule normally if LWT_NULL is provided

Returns

0 if successful

4.2 lwt.h File Reference

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

Data Structures

- struct [lwt_t](#)
The Lightweight Thread (LWT) struct.
- struct [lwt_chan_t](#)

Macros

- #define [PAGE_SIZE](#) 4096
- #define [NUM_PAGES](#) 5
- #define [STACK_SIZE](#) 4096*4
- #define [LWT_NULL](#) NULL
- #define [LWT_YIELD_NO_LWT_TO_YIELD](#) 1

Typedefs

- typedef void *(* [lwt_fn_t](#))(void *)
- typedef void *(* [lwt_chan_fn_t](#))(lwt_chan_t)

Enumerations

- enum [lwt_info_t](#) {
[LWT_INFO_NTHD_RUNNABLE](#), [LWT_INFO_NTHD_BLOCKED](#), [LWT_INFO_NTHD_ZOMBIES](#), [LWT_INFO_NCHAN](#),
[LWT_INFO_NSENDING](#), [LWT_INFO_NRECEIVING](#) }
The various statuses for a LWT.

Functions

- `lwt_t lwt_create (lwt_fn_t fn, void *data)`
Creates a LWT using the provided function pointer and the data as input for it.
- `void * lwt_join (lwt_t)`
Joins the provided thread.
- `void lwt_die (void *)`
Prepares the current thread to be cleaned up.
- `int lwt_yield (lwt_t)`
Yields to the provided LWT.
- `lwt_t lwt_current ()`
Gets the current thread.
- `int lwt_id (lwt_t)`
Gets the thread id.
- `int lwt_info (lwt_info_t t)`
Gets the counts of the info.
- `lwt_chan_t lwt_chan (int)`
Creates the channel on the receiving thread.
- `void lwt_chan_deref (lwt_chan_t)`
Deallocates the channel only if no threads still have references to the channel.
- `int lwt_snd (lwt_chan_t, void *)`
Sends the data over the channel to the receiver.
- `void * lwt_rcv (lwt_chan_t)`
Receives the data from the channel and returns it.
- `int lwt_snd_chan (lwt_chan_t, lwt_chan_t)`
Sends sending over the channel c.
- `lwt_chan_t lwt_rcv_chan (lwt_chan_t)`
Receives the data over the channel.
- `lwt_t lwt_create_chan (lwt_chan_fn_t, lwt_chan_t)`
Creates a lwt with the channel as an arg.

4.2.1 Macro Definition Documentation

4.2.1.1 `#define LWT_NULL NULL`

4.2.1.2 `#define LWT_YIELD_NO_LWT_TO_YIELD 1`

4.2.1.3 `#define NUM_PAGES 5`

4.2.1.4 `#define PAGE_SIZE 4096`

4.2.1.5 `#define STACK_SIZE 4096*4`

4.2.2 Typedef Documentation

4.2.2.1 `typedef void* (* lwt_chan_fn_t)(lwt_chan_t)`

4.2.2.2 `typedef void* (* lwt_fn_t)(void *)`

4.2.3 Enumeration Type Documentation

4.2.3.1 enum lwt_info_t

The various statuses for a LWT.

Enumerator

LWT_INFO_NTHD_RUNNABLE Thread state is runnable; it can be switched to
LWT_INFO_NTHD_BLOCKED Thread state is blocked; waiting for another thread to complete
LWT_INFO_NTHD_ZOMBIES Thread state is zombie; thread is dead and needs to be joined
LWT_INFO_NCHAN Number of channels that are active
LWT_INFO_NSENDING Number of threads blocked sending
LWT_INFO_NRECEIVING Number of threads blocked receiving

4.2.4 Function Documentation

4.2.4.1 lwt_chan_t lwt_chan (int sz)

Creates the channel on the receiving thread.

Parameters

sz	The size of the buffer
----	------------------------

Returns

A pointer to the initialized channel

4.2.4.2 void lwt_chan_deref (lwt_chan_t c)

Deallocates the channel only if no threads still have references to the channel.

Parameters

c	The channel to deallocate
---	---------------------------

4.2.4.3 lwt_t lwt_create (lwt_fnt_t fn, void * data)

Creates a LWT using the provided function pointer and the data as input for it.

Parameters

fn	The function pointer to use
data	The data to the function

Returns

A pointer to the initialized LWT

4.2.4.4 lwt_t lwt_create_chan (lwt_chan_fn_t fn, lwt_chan_t c)

Creates a lwt with the channel as an arg.

Parameters

<i>fn</i>	The function to use to create the thread
<i>c</i>	The channel to send

4.2.4.5 `lwt_t lwt_current ()`

Gets the current thread.

Returns

The current thread

4.2.4.6 `void lwt_die (void *)`

Prepares the current thread to be cleaned up.

4.2.4.7 `int lwt_id (lwt_t thread)`

Gets the thread id.

Returns

The id of the thread

4.2.4.8 `int lwt_info (lwt_info_t t)`

Gets the counts of the info.

Parameters

<i>t</i>	The info enum to get the counts
----------	---------------------------------

Returns

The count for the info enum provided

See also

[lwt_info_t](#)

4.2.4.9 `void* lwt_join (lwt_t thread)`

Joins the provided thread.

Parameters

<i>thread</i>	The thread to join on
---------------	-----------------------

4.2.4.10 void* lwt_rcv (lwt_chan_t c)

Receives the data from the channel and returns it.

Parameters

<i>c</i>	The channel to receive from
----------	-----------------------------

Returns

The data from the channel

4.2.4.11 lwt_chan_t lwt_rcv_chan (lwt_chan_t c)

Receives the data over the channel.

Parameters

<i>c</i>	The channel to use for receiving
----------	----------------------------------

Returns

The channel being sent over c

4.2.4.12 int lwt_snd (lwt_chan_t c, void * data)

Sends the data over the channel to the receiver.

Parameters

<i>c</i>	The channel to use for sending
<i>data</i>	The data for sending

Returns

-1 if there is no receiver; 0 if successful

4.2.4.13 int lwt_snd_chan (lwt_chan_t c, lwt_chan_t sending)

Sends sending over the channel c.

Parameters

<i>c</i>	The channel to send sending across
----------	------------------------------------

<i>sending</i>	The channel to send
----------------	---------------------

4.2.4.14 int lwt_yield (lwt_t lwt)

Yields to the provided LWT.

Parameters

<i>lwt</i>	The thread to yield to
------------	------------------------

Note

Will just schedule normally if LWT_NULL is provided

Returns

0 if successful

4.3 main.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include "lwt.h"
```

Macros

- #define `rdtscl(val)` `__asm__ __volatile__ ("rdtsc" : "=A" (val))`
- #define `ITER` 10000
- #define `IS_RESET()`

Functions

- void * `fn_bounce` (void *d)
- void * `fn_null` (void *d)
- void `test_perf` (void)
- void * `fn_identity` (void *d)
- void * `fn_nested_joins` (void *d)
- void * `fn_sequence` (void *d)
- void * `fn_join` (void *d)
- void `test_crt_join_sched` (void)
- int `main` (void)

Variables

- volatile int `sched` [2] = {0, 0}
- volatile int `curr` = 0

4.3.1 Macro Definition Documentation

4.3.1.1 #define IS_RESET()

Value:

```
assert ( lwt_info(LWT_INFO_NTHD_RUNNABLE) == 1 && \
         lwt_info(LWT_INFO_NTHD_ZOMBIES) == 0 && \
         lwt_info(LWT_INFO_NTHD_BLOCKED) == 0)
```

4.3.1.2 #define ITER 10000

4.3.1.3 #define rdtscll(val) __asm__ __volatile__("rdtsc" : "=A" (val))

4.3.2 Function Documentation

4.3.2.1 void* fn_bounce (void * d)

4.3.2.2 void* fn_identity (void * d)

4.3.2.3 void* fn_join (void * d)

4.3.2.4 void* fn_nested_joins (void * d)

4.3.2.5 void* fn_null (void * d)

4.3.2.6 void* fn_sequence (void * d)

4.3.2.7 int main (void)

4.3.2.8 void test_crt_join_sched (void)

4.3.2.9 void test_perf (void)

4.3.3 Variable Documentation

4.3.3.1 volatile int curr = 0

4.3.3.2 volatile int sched[2] = {0, 0}

4.4 main_chan.c File Reference

```
#include "lwt.h"
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include <time.h>
```

Data Structures

- struct [msort_args](#)
Struct for passing the args to merge sort around.

Macros

- #define [ITER](#) 10000
- #define [MERGE_SZ](#) 1000

Functions

- void * [msort](#) (lwt_chan_t main_channel)
Merge sort in parallel.
- void [merge_sort_test](#) ()
Runs the merge sort test Tests being able to create multiple child channels and joining them properly.
- void * [child_ping](#) (lwt_chan_t main_channel)
Ping channel test.
- void * [child_pong](#) (lwt_chan_t main_channel)
Receives a count, updates it and sends it back.
- void [ping_pong_test](#) ()
Runs the ping/pong test.
- int [main](#) ()

4.4.1 Macro Definition Documentation

4.4.1.1 #define ITER 10000

4.4.1.2 #define MERGE_SZ 1000

4.4.2 Function Documentation

4.4.2.1 void* child_ping (lwt_chan_t main_channel)

Ping channel test.

Parameters

<i>main_channel</i>	The channel from the main thread
---------------------	----------------------------------

Returns

0 if successful Sends count out to many siblings; tests that they receive and update it properly

4.4.2.2 void* child_pong (lwt_chan_t main_channel)

Receives a count, updates it and sends it back.

Parameters

<i>main_channel</i>	The channel from the main thread
---------------------	----------------------------------

Returns

0 if successful

4.4.2.3 int main (void)

Main function

4.4.2.4 void merge_sort_test ()

Runs the merge sort test Tests being able to create multiple child channels and joining them properly.

4.4.2.5 void* msort (lwt_chan_t main_channel)

Merge sort in parallel.

Parameters

<i>main_channel</i>	The channel from the main thread
---------------------	----------------------------------

Returns

0 if successful

Note

Adapted from wikipedia: http://en.wikipedia.org/wiki/Merge_sort#Parallel_merge_sort)

4.4.2.6 void ping_pong_test ()

Runs the ping/pong test.

Index

args

lwt, [7](#)

children

lwt, [7](#)

id

lwt, [7](#)

info

lwt, [7](#)

LWT_INFO_NCHAN

lwt.h, [19](#)

LWT_INFO_NRECEIVING

lwt.h, [19](#)

LWT_INFO_NSENDING

lwt.h, [19](#)

LWT_INFO_NTHD_BLOCKED

lwt.h, [19](#)

LWT_INFO_NTHD_RUNNABLE

lwt.h, [19](#)

LWT_INFO_NTHD_ZOMBIES

lwt.h, [19](#)

lwt

args, [7](#)

children, [7](#)

id, [7](#)

info, [7](#)

parent, [8](#)

lwt.h

LWT_INFO_NCHAN, [19](#)

LWT_INFO_NRECEIVING, [19](#)

LWT_INFO_NSENDING, [19](#)

LWT_INFO_NTHD_BLOCKED, [19](#)

LWT_INFO_NTHD_RUNNABLE, [19](#)

LWT_INFO_NTHD_ZOMBIES, [19](#)

parent

lwt, [8](#)