LWT Threads

1.0

Generated by Doxygen 1.8.8

Mon May 4 2015 13:33:59

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 files with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 http_req Struct Reference

```
#include <simple_http.h>
```

Data Fields

- int fd
- char ∗ request
- int req_len
- char ∗ path
- char ∗ resp_head
- char ∗ response
- int resp_hd_len
- int resp_len

3.1.1 Detailed Description

Redistribution of this file is permitted under the GNU General Public License v2.

Copyright 2012 by Gabriel Parmer. Author: Gabriel Parmer, gparmer@gwu.edu, 2012

3.1.2 Field Documentation

3.1.2.1 int http_req::fd

3.1.2.2 char∗ http_req::path

3.1.2.3 int http_req::req_len

3.1.2.4 char∗ http_req::request

3.1.2.5 int http_req::resp_hd_len

3.1.2.6 char∗ http_req::resp_head

3.1.2.7 int http_req::resp_len

3.1.2.8 char * http_req::response

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

- simple_http.h

3.2 kthd_event Struct Reference

```
#include <objects.h>
```

Data Fields

- lwt_t originator
- lwt_t lwt
- lwt_chan_t channel
- lwt_cgrp_t group
- lwt_kthd_t kthd
- int is_done
- int block
- lwt_remote_op_t op

3.2.1 Field Documentation

3.2.1.1 int kthd_event::block

3.2.1.2 lwt_chan_t kthd_event::channel

3.2.1.3 lwt_cgrp_t kthd_event::group

3.2.1.4 int kthd_event::is_done

3.2.1.5 lwt_kthd_t kthd_event::kthd

3.2.1.6 lwt_t kthd_event::lwt

3.2.1.7 lwt_remote_op_t kthd_event::op

3.2.1.8 lwt_t kthd_event::originator

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

- objects.h

3.3 lwt_cgrp_t Struct Reference

Channel group for handling events within a group.

`#include <objects.h>`

Public Member Functions

- LIST_HEAD (head_channels_in_group, lwt_channel) head_channels_in_group
- TAILQ_HEAD (head_event, lwt_channel) head_event

Data Fields

- lwt_t waiting_thread
- lwt_t creator_thread

3.3.1 Detailed Description

Channel group for handling events within a group.

3.3.2 Member Function Documentation

3.3.2.1 lwt_cgrp_t::LIST_HEAD (head_channels_in_group , lwt_channel)

Definition of the head in the channels

3.3.2.2 lwt_cgrp_t::TAILQ_HEAD (head_event , lwt_channel)

Definition of the head node for the event queue

3.3.3 Field Documentation

3.3.3.1 lwt_t lwt_cgrp_t::creator_thread

Creator thread

3.3.3.2 lwt_t lwt_cgrp_t::waiting_thread

Waiting thread

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

- objects.h

3.4 lwt_chan_t Struct Reference

The channel for synchronous and asynchronous communication.

`#include <objects.h>`

Public Member Functions

- LIST_HEAD (head_senders, lwt) head_senders
- TAILQ_HEAD (head_blocked_senders, lwt) head_blocked_senders
- LIST_ENTRY (lwt_channel) receiver_channels
- LIST_ENTRY (lwt_channel) channels_in_group
- TAILQ_ENTRY (lwt_channel) events

Data Fields

- int snd_cnt
- lwt_t receiver
- void ∗ sync_buffer
- void ∗∗ async_buffer
- unsigned volatile int start_index
- unsigned volatile int end_index
- unsigned int num_entries
- unsigned int buffer_size
- lwt_cgrp_t channel_group
- void ∗ mark
- lwt_kthd_t kthd

3.4.1 Detailed Description

The channel for synchronous and asynchronous communication.

3.4.2 Member Function Documentation

3.4.2.1 lwt_chan_t::LIST_ENTRY (**lwt_channel**)

List of receiver channels in a lwt

3.4.2.2 lwt_chan_t::LIST_ENTRY (**lwt_channel**)

Channels in group entries

3.4.2.3 lwt_chan_t::LIST_HEAD (head_senders , lwt)

Definition of the senders head pointer

3.4.2.4 lwt_chan_t::TAILQ_ENTRY (lwt_channel)

Channels in event

3.4.2.5 lwt_chan_t::TAILQ_HEAD (head_blocked_senders , lwt)

Definition of the blocked senders head pointer

3.4.3 Field Documentation

3.4.3.1 void** lwt_chan_t::async_buffer

Async Buffer to be passed to the channel

3.4.3.2 unsigned int lwt_chan_t::buffer_size

Size of the buffer

3.4.3.3 lwt_cgrp_t lwt_chan_t::channel_group

Channel group

3.4.3.4 unsigned volatile int lwt_chan_t::end_index

End index of the buffer

3.4.3.5 lwt_kthd_t lwt_chan_t::kthd

Kthd of the receiver

3.4.3.6 void* lwt_chan_t::mark

Mark for channel

3.4.3.7 unsigned int lwt_chan_t::num_entries

Num entries

3.4.3.8 lwt_t lwt_chan_t::receiver

The receiving thread

3.4.3.9 int lwt_chan_t::snd_cnt

The number of senders

3.4.3.10 unsigned volatile int lwt_chan_t::start_index

Start index of the buffer

3.4.3.11 void* lwt_chan_t::sync_buffer

Sync buffer to be passed to the channel

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

- objects.h

3.5 lwt_kthd_data Struct Reference

```
#include <objects.h>
```

Data Fields

- lwt_chan_fn_t channel_fn
- lwt_chan_t channel
- lwt_flags_t flags
- lwt_t parent
- int ready

3.5.1 Field Documentation

3.5.1.1 lwt_chan_t lwt_kthd_data::channel

3.5.1.2 lwt_chan_fn_t lwt_kthd_data::channel_fn

3.5.1.3 lwt_flags_t lwt_kthd_data::flags

3.5.1.4 lwt_t lwt_kthd_data::parent

3.5.1.5 int lwt_kthd_data::ready

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

- objects.h

3.6 lwt_kthd_t Struct Reference

```
#include <objects.h>
```

Public Member Functions

- LIST_HEAD (head_lwts_in_kthd, lwt) head_lwts_in_kthd
- TAILQ_HEAD (head_runnable_threads, lwt) head_runnable_threads

Data Fields

- pthread_t pthread
- int is_blocked
- pthread_mutex_t blocked_mutex
- pthread_cond_t blocked_cv
- lwt_t buffer_thread
- struct kthd_event ∗ event_buffer [EVENT_BUFFER_SIZE]
- volatile unsigned int buffer_head
- volatile unsigned int buffer_tail

3.6.1 Member Function Documentation

3.6.1.1 lwt_kthd_t::LIST_HEAD (head_lwts_in_kthd , lwt)

Point to the head of the list of lwts associated with a kthd

3.6.1.2 lwt_kthd_t::TAILQ_HEAD (head_runnable_threads , lwt)

Pointer to the head of the run queue

3.6.2 Field Documentation

3.6.2.1 pthread_cond_t lwt_kthd_t::blocked_cv

Condition variable for the lwt buffer thread

3.6.2.2 pthread_mutex_t lwt_kthd_t::blocked_mutex

Mutex for the blocked lwt buffer thread

3.6.2.3 volatile unsigned int lwt_kthd_t::buffer_head

Head of the buffer

3.6.2.4 volatile unsigned int lwt_kthd_t::buffer_tail

Tail of the buffer

3.6.2.5 lwt_t lwt_kthd_t::buffer_thread

Buffer thread for the lwt

3.6.2.6 struct kthd_event∗ lwt_kthd_t::event_buffer[EVENT_BUFFER_SIZE]

Event buffer for remote communication

3.6.2.7 int lwt_kthd_t::is_blocked

Status flag for if the current remote thread is blocked

3.6.2.8 pthread_t lwt_kthd_t::pthread

The Pthread belonging to the kthd

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

- objects.h

3.7 lwt_t Struct Reference

The Lightweight Thread (LWT) struct.

`#include <objects.h>`

Public Member Functions

- LIST_HEAD (head_children, lwt) head_children
- LIST_ENTRY (lwt) siblings
- LIST_ENTRY (lwt) current_threads
- TAILQ_ENTRY (lwt) runnable_threads
- LIST_ENTRY (lwt) ready_pool_threads
- LIST_ENTRY (lwt) senders
- TAILQ_ENTRY (lwt) blocked_senders
- LIST_HEAD (head_receiver_channel, lwt_channel) head_receiver_channel
- LIST_ENTRY (lwt) lwts_in_kthd

Data Fields

- long ∗ max_addr_thread_stack
- long ∗ min_addr_thread_stack
- long ∗ thread_sp
- lwt_flags_t flags
- lwt_t parent
- void ∗ sync_buffer
- lwt_fnt_t start_routine
- void ∗ args
- void ∗ return_value
- lwt_info_t info
- int id
- lwt_kthd_t kthd

3.7.1 Detailed Description

The Lightweight Thread (LWT) struct.

3.7.2 Member Function Documentation

3.7.2.1 lwt_t::LIST_ENTRY (lwt)

Pointers to sibling threads

3.7.2.2 lwt_t::LIST_ENTRY (lwt)

Pointers to the current threads

3.7.2.3 lwt_t::LIST_ENTRY (lwt)

List of runnable pool threads

3.7.2.4 lwt_t::LIST_ENTRY (lwt)

List of senders

3.7.2.5 lwt_t::LIST_ENTRY (lwt)

List of lwts in the kthd

3.7.2.6 lwt_t::LIST_HEAD (head_children , lwt)

Head of the list of children lwt's associated with the lwt

3.7.2.7 lwt_t::LIST_HEAD (head_receiver_channel , lwt_channel)

Head of the receiver channels associated with the lwt

3.7.2.8 lwt_t::TAILQ_ENTRY (lwt)

List of runnable threads

3.7.2.9 lwt_t::TAILQ_ENTRY (lwt)

List of blocked senders

3.7.3 Field Documentation

3.7.3.1 void∗ lwt_t::args

The args for the start_routine

3.7.3.2 lwt_flags_t lwt_t::flags

The flags associated with the lwt

3.7.3.3 int lwt_t::id

The id of the thread

3.7.3.4 lwt_info_t lwt_t::info

The current status of the thread

3.7.3.5 lwt_kthd_t lwt_t::kthd

Pointer to kthd

3.7.3.6 long∗ lwt_t::max_addr_thread_stack

Pointer to the max address of the stack

3.7.3.7 long∗ lwt_t::min_addr_thread_stack

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

3.7.3.8 lwt_t lwt_t::parent

Parent thread

3.7.3.9 void∗ lwt_t::return_value

The return value from the routine

3.7.3.10 lwt_fnt_t lwt_t::start_routine

The start routine for the thread to run

3.7.3.11 void∗ lwt_t::sync_buffer

Sync buffer

3.7.3.12 long∗ lwt_t::thread_sp

The current thread stack pointer for the thread

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

- objects.h

3.8 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.

3.8.1 Detailed Description

Struct for passing the args to merge sort around.

3.8.2 Field Documentation

3.8.2.1 int msort_args::begin_index

The begin index of the segment.

3.8.2.2 int∗ msort_args::data

The int array holding randomly generated data.

3.8.2.3 int msort_args::end_index

THe end index of the segment.

3.8.2.4 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 cas.h File Reference

4.2 content.c File Reference

```
#include <unistd.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
```

Macros

- #define MAX_CONTENT_SZ (1024∗1024∗10)

Functions

- char ∗ error_resp (char ∗path, int ∗len)
- int sanity_check (char ∗path)
- char ∗ content_get (char ∗path, int ∗content_len)

4.2.1 Macro Definition Documentation

4.2.1.1 #define MAX_CONTENT_SZ (1024∗1024∗10)

4.2.2 Function Documentation

4.2.2.1 char∗ content_get (char ∗ *path,* int ∗ *content_len*)

Redistribution of this file is permitted under the GNU General Public License v2.

Copyright 2012 by Gabriel Parmer. Author: Gabriel Parmer, gparmer@gwu.edu, 2012

4.2.2.2 char∗ error_resp (char ∗ *path,* int ∗ *len*)

4.2.2.3 int sanity_check (char ∗ *path*)

4.3 content.h File Reference

Functions

- char ∗ content_get (char ∗path, int ∗content_len)

4.3.1 Function Documentation

4.3.1.1 char∗ content_get (char ∗ *path,* int ∗ *content_len*)

Redistribution of this file is permitted under the GNU General Public License v2.

Copyright 2012 by Gabriel Parmer. Author: Gabriel Parmer, gparmer@gwu.edu, 2012

4.4 enums.h File Reference

Enumerations

- enum lwt_info_t {
  LWT_INFO_NTHD_RUNNABLE, LWT_INFO_NTHD_BLOCKED, LWT_INFO_NTHD_ZOMBIES, LWT_INFO_↵
  NTHD_READY_POOL,
  LWT_INFO_NCHAN, LWT_INFO_NSENDING, LWT_INFO_NRECEIVING, LWT_INFO_REAPER_READY }

  The various statuses for a LWT.
- enum lwt_flags_t { LWT_JOIN = 0, LWT_NOJOIN = 1 }
- enum lwt_remote_op_t {
  LWT_REMOTE_ADD_SENDER_TO_CHANNEL, LWT_REMOTE_REMOVE_SENDER_FROM_CHANNEL, L↵
  WT_REMOTE_ADD_BLOCKED_SENDER_TO_CHANNEL, LWT_REMOTE_REMOVE_BLOCKED_SENDER↵
  _FROM_CHANNEL,
  LWT_REMOTE_ADD_CHANNEL_TO_GROUP, LWT_REMOTE_REMOVE_CHANNEL_FROM_GROUP, LW↵
  T_REMOTE_ADD_EVENT_TO_GROUP, LWT_REMOTE_REMOVE_EVENT_FROM_GROUP,
  LWT_REMOTE_SIGNAL }

4.4.1 Enumeration Type Documentation

4.4.1.1 enum **lwt_flags_t**

flags for determining if the lwt is joinable

Enumerator

LWT_JOIN	lwt is joinable
LWT_NOJOIN	lwt is not joinable

4.4.1.2 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_NTHD_READY_POOL	Number of ready pool threads
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
LWT_INFO_REAPER_READY	Reaper is ready to consume

4.4.1.3 enum lwt_remote_op_t

Enumerator

LWT_REMOTE_ADD_SENDER_TO_CHANNEL	Add a lwt sender to a channel
LWT_REMOTE_REMOVE_SENDER_FROM_CHANNEL	Remove a lwt sender to a channel
LWT_REMOTE_ADD_BLOCKED_SENDER_TO_CHANNEL	Add a blocked lwt to a channel
LWT_REMOTE_REMOVE_BLOCKED_SENDER_FROM_CHANNEL	Remove a blocked sender from a channel
LWT_REMOTE_ADD_CHANNEL_TO_GROUP	Add a channel to a group
LWT_REMOTE_REMOVE_CHANNEL_FROM_GROUP	Remove a channel from the group
LWT_REMOTE_ADD_EVENT_TO_GROUP	Add an event to remote group
LWT_REMOTE_REMOVE_EVENT_FROM_GROUP	Remove an event from a remote group
LWT_REMOTE_SIGNAL	Signal

4.5 faa.c File Reference

Functions

- int fetch_and_add (volatile unsigned int ∗variable, int value)

  Implementation of fetch and add.

4.5.1 Function Documentation

4.5.1.1 int fetch_and_add (volatile unsigned int ∗ *variable,* int *value*) `[inline]`

Implementation of fetch and add.

See also

Taken from wikipedia: `https://www.en.wikipedia.org/wiki/Fetch-and-add`

Parameters

variable	The variable to modify
value	The value to modify

Returns

The updated variabled

4.6 faa.h File Reference

Functions

- int fetch_and_add (volatile unsigned int ∗, int)

  Implementation of fetch and add.

4.6.1 Function Documentation

4.6.1.1 int fetch_and_add (volatile unsigned int ∗ *variable,* int *value*) `[inline]`

Implementation of fetch and add.

See also

Taken from wikipedia: `https://www.en.wikipedia.org/wiki/Fetch-and-add`

Parameters

variable	The variable to modify
value	The value to modify

Returns

The updated variabled

4.7 kthd_server.c File Reference

```
#include "kthd_server.h"
#include "search.h"
#include "lwt_cgrp.h"
#include "lwt_chan.h"
#include "lwt_kthd.h"
#include "objects.h"
#include "simple_http.h"
#include "content.h"
#include "server.h"
#include "stdio.h"
#include "stdlib.h"
#include "unistd.h"
#include "assert.h"
#include "string.h"
```

Macros

- #define MAX_CACHE_ENTRIES 10
- #define POOL_SIZE 2
- #define MAX_ACCEPTORS 2
- #define LWT_CACHE 3
- #define MAX_REQ_SZ 1024

Functions

- void respond_and_free_req_kthd (struct http_req ∗r, char ∗response, int len)
- struct http_req ∗ newfd_create_req_kthd (int new_fd)

  Helper function for creating an http request.
- void ∗ read_fs (lwt_chan_t cache_channel)

  Processes the file system request; used for thread pool.
- void ∗ spawn_fs_workers (lwt_chan_t main_channel)

  Wrapper for the the file system workers; used for thread pool.
- void ∗ read_cache (lwt_chan_t kthd_channel)

  LWT function for caching; checks if the path has been cached; if so, return it; else hit fs threads.
- void ∗ read_cache_kthd (lwt_chan_t main_channel)

  Function for running on cache to manage lwt thread pool.
- void ∗ accept_worker (lwt_chan_t main_channel)

  Accept worker kthd; accepts the new httd request.
- void process_kthd_server (int accept_fd)

  Main function for the server; sets up channels and then passes data from cache to kthd modules.

4.7.1 Macro Definition Documentation

4.7.1.1 #define LWT_CACHE 3

4.7.1.2 #define MAX_ACCEPTORS 2

4.7.1.3 #define MAX_CACHE_ENTRIES 10

4.7.1.4 #define MAX_REQ_SZ 1024

4.7.1.5 #define POOL_SIZE 2

4.7.2 Function Documentation

4.7.2.1 void∗ accept_worker (lwt_chan_t *main_channel*)

Accept worker kthd; accepts the new httd request.

Parameters

main_channel	The channel to send data across

Returns

NULL

4.7.2.2 struct http_req∗ newfd_create_req_kthd (int *new_fd*)

Helper function for creating an http request.

Parameters

new_fd	The file descriptor to open

Returns

The Http request received from the file descriptor

4.7.2.3 void process_kthd_server (int *accept_fd*)

Main function for the server; sets up channels and then passes data from cache to kthd modules.

Parameters

accept_fd	The file descriptor for the http port being used

4.7.2.4 void∗ read_cache (lwt_chan_t *kthd_channel*)

LWT function for caching; checks if the path has been cached; if so, return it; else hit fs threads.

Parameters

kthd_channel	The channel for the spawner

Returns

NULL

4.7.2.5 void∗ read_cache_kthd (lwt_chan_t *main_channel*)

Function for running on cache to manage lwt thread pool.

Parameters

main_channel	The channel from main used for passing data to other kthds

Returns

NULL

4.7.2.6 void∗ read_fs (lwt_chan_t *cache_channel*)

Processes the file system request; used for thread pool.

Parameters

cache_channel	The channel to receive

Returns

NULL

4.7.2.7 void respond_and_free_req_kthd (struct **http_req** ∗ *r,* char ∗ *response,* int *len*)

4.7.2.8 void∗ spawn_fs_workers (lwt_chan_t *main_channel*)

Wrapper for the the file system workers; used for thread pool.

Parameters

main_channel	The channel for sending the fs channel to

4.8 kthd_server.h File Reference

Functions

- void process_kthd_server (int accept_fd)

  Main function for the server; sets up channels and then passes data from cache to kthd modules.

4.8.1 Function Documentation

4.8.1.1 void process_kthd_server (int *accept_fd*)

Main function for the server; sets up channels and then passes data from cache to kthd modules.

Parameters

accept_fd	The file descriptor for the http port being used

4.9 lwt.c File Reference

```
#include "lwt.h"
#include "lwt_chan.h"
#include "lwt_cgrp.h"
#include "lwt_kthd.h"
#include "pthread.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.
- #define POOL_SIZE 100

  The size of the pool.

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.
- __thread LIST_HEAD (head_current, lwt)

  List of all active threads created.
- void __insert_runnable_tail (lwt_t thread)

  Inserts the given thread to the tail of the runnable thread list.
- 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_new_lwt (lwt_t thread)

  Initializes the provided thread.
- void __reinit_lwt (lwt_t thread)

  Reinitializes the given 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.
- void lwt_block (lwt_info_t info)

  Blocks the current thread.
- void lwt_signal (lwt_t thread)

  Signals the non-running thread to run.
- 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_fnt_t fn, void ∗data, lwt_flags_t flags)

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

Variables

- __thread lwt_t current_thread = NULL

  Pointer to the current thread.
- __thread lwt_t original_thread = NULL

  Pointer to the original/main thread.

4.9.1 Macro Definition Documentation

4.9.1.1 #define DEFAULT_ID -1

The default id provided to threads before actually generating them.

4.9.1.2 #define INIT_ID 1

The initial thread id.

4.9.1.3 #define POOL_SIZE 100

The size of the pool.

4.9.2 Function Documentation

4.9.2.1 __attribute__ ((constructor))

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

4.9.2.2 __attribute__ ((destructor))

Cleans up all remaining threads on exit.

4.9.2.3 void __cleanup_joined_thread (lwt_t *lwt*)

Cleans up the thread on join.

Parameters

lwt	The thread to join on

4.9.2.4 void __init_lwt_main (lwt_t *thread*)

Initializes the main thread.

Parameters

thread	The main thread

4.9.2.5 void __init_new_lwt (lwt_t *thread*)

Initializes the provided thread.

Parameters

thread	The thread to init

4.9.2.6 void __insert_runnable_tail (lwt_t *thread*)

Inserts the given thread to the tail of the runnable thread list.

Parameters

thread	The new thread to be inserted in the list of runnable threads

4.9.2.7 void __lwt_dispatch (lwt_t *next,* lwt_t *current*)

Dispatch function for switching between threads.

Parameters

next	The next thread to switch to
current	The current thread

4.9.2.8 void __lwt_schedule (void)

Schedules the next_current thread to switch to and dispatches.

4.9.2.9 void ∗ __lwt_stack_get (void)

Allocates the stack for a LWT and returns it.

4.9.2.10 void __lwt_stack_return (void ∗ *stack*)

Frees the provided stack.

Parameters

stack	The LWT stack to free

4.9.2.11 void __lwt_trampoline (void)

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

4.9.2.12 void __reinit_lwt (lwt_t *thread*)

Reinitializes the given thread.

Parameters

thread	The thread to reinitialize

4.9.2.13 __thread LIST_HEAD (*head_current,* lwt)

List of all active threads created.

Counter for the id

Returns

The next id to use

4.9.2.14 void lwt_block (lwt_info_t *info*)

Blocks the current thread.

Parameters

info	The state to set the thread

4.9.2.15 lwt_t lwt_create (lwt_fnt_t *fn*, void ∗ *data*, lwt_flags_t *flags*)

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
flags	The flags to be associated with the thread

Returns

A pointer to the initialized LWT

4.9.2.16 lwt_t lwt_current ( ) [inline]

Gets the current thread.

Returns

The current thread

4.9.2.17 void lwt_die (void ∗ *value*)

Prepares the current thread to be cleaned up.

4.9.2.18 int lwt_id (lwt_t *thread*) [inline]

Gets the thread id.

Returns

The id of the thread

4.9.2.19 int lwt_info (lwt_info_t *t*)

Gets the counts of the info.

Parameters

t	The info enum to get the counts

Returns

The count for the info enum provided

See also

lwt_info_t

4.9.2.20 void∗ lwt_join (lwt_t *thread*)

Joins the provided thread.

Parameters

thread	The thread to join on

4.9.2.21 void lwt_signal (lwt_t *thread*)

Signals the non-running thread to run.

Parameters

thread	The thread to be worken

4.9.2.22 int lwt_yield (lwt_t *lwt*)

Yields to the provided LWT.

Parameters

lwt	The thread to yield to

Note

Will just schedule normally if LWT_NULL is provided

Returns

0 if successful

4.9.3 Variable Documentation

4.9.3.1 __thread lwt_t current_thread = NULL

Pointer to the current thread.

4.9.3.2 __thread lwt_t original_thread = NULL

Pointer to the original/main thread.

4.10 lwt.h File Reference

```
#include "objects.h"
```

Functions

- lwt_t lwt_create (lwt_fnt_t, void ∗, lwt_flags_t)

  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)

  Gets the counts of the info.
- void lwt_block (lwt_info_t)

  Blocks the current thread.
- void lwt_signal (lwt_t)

  Signals the non-running thread to run.
- void __init__ ()
- void __destroy__ ()

4.10.1 Function Documentation

4.10.1.1 void __destroy__ ( )

4.10.1.2 void __init__ ( )

4.10.1.3 void lwt_block (**lwt_info_t** *info*)

Blocks the current thread.

Parameters

info	The state to set the thread

4.10.1.4 lwt_t lwt_create (**lwt_fnt_t** *fn,* void ∗ *data,* **lwt_flags_t** *flags*)

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
flags	The flags to be associated with the thread

Returns

A pointer to the initialized LWT

4.10.1.5 lwt_t lwt_current ( ) [inline]

Gets the current thread.

Returns

The current thread

4.10.1.6 void lwt_die (void *)

Prepares the current thread to be cleaned up.

4.10.1.7 int lwt_id (lwt_t *thread*) [inline]

Gets the thread id.

Returns

The id of the thread

4.10.1.8 int lwt_info (lwt_info_t *t*)

Gets the counts of the info.

Parameters

t	The info enum to get the counts

Returns

The count for the info enum provided

See also

lwt_info_t

4.10.1.9 void* lwt_join (lwt_t *thread*)

Joins the provided thread.

Parameters

thread	The thread to join on

4.10.1.10 void lwt_signal (lwt_t *thread*)

Signals the non-running thread to run.

Parameters

thread	The thread to be worken

4.10.1.11 int lwt_yield (lwt_t *lwt*)

Yields to the provided LWT.

Parameters

lwt	The thread to yield to

Note

Will just schedule normally if LWT_NULL is provided

Returns

0 if successful

4.11 lwt_cgrp.c File Reference

```
#include "lwt_cgrp.h"
#include "lwt.h"
#include "lwt_chan.h"
#include "stdlib.h"
#include "assert.h"
#include "stdio.h"
#include "sys/queue.h"
```

Functions

- void __init_event (lwt_chan_t channel)
  Initializes the event for when data is added to the channel.
- void __remove_event (lwt_chan_t channel, lwt_cgrp_t group)
  Removes an event from the group.
- lwt_cgrp_t lwt_cgrp ()
  Creates a group of channels.
- int lwt_cgrp_free (lwt_cgrp_t group)
  Frees the group if there are no pending events.
- int lwt_cgrp_add (lwt_cgrp_t group, lwt_chan_t channel)
  Adds the channel to the group if the channel hasn't already been added to a group.
- int lwt_cgrp_rem (lwt_cgrp_t group, lwt_chan_t channel)
  Removes the channel from the group.
- lwt_chan_t lwt_cgrp_wait (lwt_cgrp_t group)
  Waits until there is a pending event in the queue.
- void lwt_chan_mark_set (lwt_chan_t channel, void ∗mark)
  Marks the channel.
- void ∗ lwt_chan_mark_get (lwt_chan_t channel)
  Grabs the mark from the channel.

4.11.1 Function Documentation

4.11.1.1 void __init_event (lwt_chan_t *channel*)

Initializes the event for when data is added to the channel.

Parameters

channel	The channel with the new data
sender	The sender lwt

4.11.1.2 void __remove_event (lwt_chan_t *channel,* lwt_cgrp_t *group*)

Removes an event from the group.

Parameters

channel	The channel (i.e. event) to remove
group	The group to remove the event from

4.11.1.3 lwt_cgrp_t lwt_cgrp ( )

Creates a group of channels.

Returns

The group of channels

Note

By default, the group is empty

4.11.1.4 int lwt_cgrp_add (lwt_cgrp_t *group,* lwt_chan_t *channel*)

Adds the channel to the group if the channel hasn't already been added to a group.

Parameters

group	The group to add the channel to
channel	The channel to add

Returns

0 if successful; -1 if the channel is already part of a group

4.11.1.5 int lwt_cgrp_free (lwt_cgrp_t *group*)

Frees the group if there are no pending events.

Parameters

group	The channel group to free

Returns

0 if successful; -1 if there are pending events

4.11.1.6 int lwt_cgrp_rem (lwt_cgrp_t *group*, lwt_chan_t *channel*)

Removes the channel from the group.

Parameters

group	The group to remove the channel from
channel	The channel to remove

Returns

0 if successful; -1 if the channel isn't part of the group; 1 if the group has a pending event

4.11.1.7 lwt_chan_t lwt_cgrp_wait (lwt_cgrp_t *group*)

Waits until there is a pending event in the queue.

Parameters

group	The group to wait for

Returns

The event in the queue

4.11.1.8 void* lwt_chan_mark_get (lwt_chan_t *channel*)

Grabs the mark from the channel.

Parameters

channel	The channel to read

4.11.1.9 void lwt_chan_mark_set (lwt_chan_t *channel*, void * *mark*)

Marks the channel.

Parameters

channel	The channel to mark

mark	The marker to set

4.12 lwt_cgrp.h File Reference

```
#include "objects.h"
```

Functions

- lwt_cgrp_t lwt_cgrp ()

  Creates a group of channels.
- int lwt_cgrp_free (lwt_cgrp_t)

  Frees the group if there are no pending events.
- int lwt_cgrp_add (lwt_cgrp_t, lwt_chan_t)

  Adds the channel to the group if the channel hasn't already been added to a group.
- int lwt_cgrp_rem (lwt_cgrp_t, lwt_chan_t)

  Removes the channel from the group.
- lwt_chan_t lwt_cgrp_wait (lwt_cgrp_t)

  Waits until there is a pending event in the queue.
- void lwt_chan_mark_set (lwt_chan_t, void ∗)

  Marks the channel.
- void ∗ lwt_chan_mark_get (lwt_chan_t)

  Grabs the mark from the channel.
- void __init_event (lwt_chan_t)

  Initializes the event for when data is added to the channel.
- void __remove_event (lwt_chan_t, lwt_cgrp_t)

  Removes an event from the group.

4.12.1 Function Documentation

4.12.1.1 void __init_event (lwt_chan_t *channel*)

Initializes the event for when data is added to the channel.

Parameters

channel	The channel with the new data
sender	The sender lwt

4.12.1.2 void __remove_event (lwt_chan_t *channel,* lwt_cgrp_t *group*)

Removes an event from the group.

Parameters

channel	The channel (i.e. event) to remove
group	The group to remove the event from

4.12.1.3 lwt_cgrp_t lwt_cgrp ( )

Creates a group of channels.

Returns

The group of channels

Note

By default, the group is empty

4.12.1.4 int lwt_cgrp_add (lwt_cgrp_t *group*, lwt_chan_t *channel*)

Adds the channel to the group if the channel hasn't already been added to a group.

Parameters

group	The group to add the channel to
channel	The channel to add

Returns

0 if successful; -1 if the channel is already part of a group

4.12.1.5 int lwt_cgrp_free (lwt_cgrp_t *group*)

Frees the group if there are no pending events.

Parameters

group	The channel group to free

Returns

0 if successful; -1 if there are pending events

4.12.1.6 int lwt_cgrp_rem (lwt_cgrp_t *group*, lwt_chan_t *channel*)

Removes the channel from the group.

Parameters

group	The group to remove the channel from
channel	The channel to remove

Returns

0 if successful; -1 if the channel isn't part of the group; 1 if the group has a pending event

4.12.1.7 lwt_chan_t lwt_cgrp_wait (lwt_cgrp_t *group*)

Waits until there is a pending event in the queue.

Parameters

group	The group to wait for

Returns

The event in the queue

4.12.1.8 void∗ lwt_chan_mark_get (lwt_chan_t *channel*)

Grabs the mark from the channel.

Parameters

channel	The channel to read

4.12.1.9 void lwt_chan_mark_set (lwt_chan_t *channel,* void ∗ *mark*)

Marks the channel.

Parameters

channel	The channel to mark
mark	The marker to set

4.13 lwt_chan.c File Reference

```
#include "lwt_chan.h"
#include "lwt.h"
#include "lwt_cgrp.h"
#include "lwt_kthd.h"
#include "objects.h"
#include "stdio.h"
#include "stdlib.h"
#include "assert.h"
#include "faa.h"
```

Functions

- void __insert_sender_to_chan (lwt_chan_t chan, lwt_t lwt)

  Inserts the sender into the channel.
- void __remove_sender_from_chan (lwt_chan_t chan, lwt_t lwt)

  Removes the sender from the channel.
- void __insert_blocked_sender_to_chan (lwt_chan_t chan, lwt_t lwt)

  Inserts the sender onto the blocked queue.
- void __remove_blocked_sender_from_chan (lwt_chan_t chan, lwt_t lwt)

  Removes the sender from the channel's blocked queue.
- void ∗ __pop_data_from_async_buffer (lwt_chan_t c)

  Pops the data into the buffer.
- lwt_chan_t lwt_chan (int sz)

  Creates the channel on the receiving thread.
- int lwt_snd (lwt_chan_t c, void ∗data)

  Sends the data over the channel to the receiver.
- 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.
- void lwt_chan_deref (lwt_chan_t c)

  Deallocates the channel only if no threads still have references to the channel.
- void ∗ lwt_rcv (lwt_chan_t c)

  Receives the data from the channel and returns it.
- lwt_t lwt_create_chan (lwt_chan_fn_t fn, lwt_chan_t c, lwt_flags_t flags)

  Creates a lwt with the channel as an arg.

4.13.1 Function Documentation

4.13.1.1 void __insert_blocked_sender_to_chan (lwt_chan_t *chan*, lwt_t *lwt*)

Inserts the sender onto the blocked queue.

Parameters

chan	The channel owning the queue
lwt	The sender to add to the queue

4.13.1.2 void __insert_sender_to_chan (lwt_chan_t *chan*, lwt_t *lwt*)

Inserts the sender into the channel.

Parameters

chan	The channel to insert the sender

lwt	The sender lwt

4.13.1.3 void∗ __pop_data_from_async_buffer (lwt_chan_t *c*)

Pops the data into the buffer.

Parameters

c	The channel to remove the data from
data	The data to remove If the buffer is empty, it will block until there is something to read

4.13.1.4 void __remove_blocked_sender_from_chan (lwt_chan_t *chan,* lwt_t *lwt*)

Removes the sender from the channel's blocked queue.

Parameters

chan	The channel owning the queue
lwt	The sender to remove from the queue

4.13.1.5 void __remove_sender_from_chan (lwt_chan_t *chan,* lwt_t *lwt*)

Removes the sender from the channel.

Parameters

chan	The channel to remove the sender from
lwt	The sender to remove

4.13.1.6 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

4.13.1.7 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

4.13.1.8 lwt_t lwt_create_chan (lwt_chan_fn_t *fn*, lwt_chan_t *c*, lwt_flags_t *flags*)

Creates a lwt with the channel as an arg.

Parameters

fn	The function to use to create the thread
c	The channel to send
flags	The flags for the thread

Returns

The thread to return

4.13.1.9 void∗ lwt_rcv (lwt_chan_t *c*)

Receives the data from the channel and returns it.

Parameters

c	The channel to receive from

Returns

The data from the channel

4.13.1.10 lwt_chan_t lwt_rcv_chan (lwt_chan_t *c*)

Receives the data over the channel.

Parameters

c	The channel to use for receiving

Returns

The channel being sent over c

4.13.1.11 int lwt_snd (lwt_chan_t *c*, void ∗ *data*)

Sends the data over the channel to the receiver.

Parameters

c	The channel to use for sending

data	The data for sending

Returns

-1 if there is no receiver; 0 if successful

4.13.1.12 int lwt_snd_chan (lwt_chan_t *c*, lwt_chan_t *sending*)

Sends sending over the channel c.

Parameters

c	The channel to send sending across
sending	The channel to send

4.14 lwt_chan.h File Reference

```
#include "objects.h"
```

Functions

- 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, lwt_flags_t)

  Creates a lwt with the channel as an arg.
- void __insert_sender_to_chan (lwt_chan_t, lwt_t)

  Inserts the sender into the channel.
- void __remove_sender_from_chan (lwt_chan_t, lwt_t)

  Removes the sender from the channel.
- void __insert_blocked_sender_to_chan (lwt_chan_t, lwt_t)

  Inserts the sender onto the blocked queue.
- void __remove_blocked_sender_from_chan (lwt_chan_t, lwt_t)

  Removes the sender from the channel's blocked queue.

4.14.1 Function Documentation

4.14.1.1 void __insert_blocked_sender_to_chan (lwt_chan_t *chan*, lwt_t *lwt*)

Inserts the sender onto the blocked queue.

Parameters

chan	The channel owning the queue
lwt	The sender to add to the queue

4.14.1.2 void __insert_sender_to_chan (lwt_chan_t *chan,* lwt_t *lwt*)

Inserts the sender into the channel.

Parameters

chan	The channel to insert the sender
lwt	The sender lwt

4.14.1.3 void __remove_blocked_sender_from_chan (lwt_chan_t *chan,* lwt_t *lwt*)

Removes the sender from the channel's blocked queue.

Parameters

chan	The channel owning the queue
lwt	The sender to remove from the queue

4.14.1.4 void __remove_sender_from_chan (lwt_chan_t *chan,* lwt_t *lwt*)

Removes the sender from the channel.

Parameters

chan	The channel to remove the sender from
lwt	The sender to remove

4.14.1.5 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

4.14.1.6 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

4.14.1.7 lwt_t lwt_create_chan (lwt_chan_fn_t *fn,* lwt_chan_t *c,* lwt_flags_t *flags*)

Creates a lwt with the channel as an arg.

Parameters

fn	The function to use to create the thread
c	The channel to send
flags	The flags for the thread

Returns

The thread to return

4.14.1.8 void∗ lwt_rcv (lwt_chan_t *c*)

Receives the data from the channel and returns it.

Parameters

c	The channel to receive from

Returns

The data from the channel

4.14.1.9 lwt_chan_t lwt_rcv_chan (lwt_chan_t *c*)

Receives the data over the channel.

Parameters

c	The channel to use for receiving

Returns

The channel being sent over c

4.14.1.10 int lwt_snd (lwt_chan_t *c,* void ∗ *data*)

Sends the data over the channel to the receiver.

Parameters

c	The channel to use for sending
data	The data for sending

Returns

-1 if there is no receiver; 0 if successful

4.14.1.11 int lwt_snd_chan (lwt_chan_t *c*, lwt_chan_t *sending*)

Sends sending over the channel c.

Parameters

c	The channel to send sending across
sending	The channel to send

4.15 lwt_kthd.c File Reference

```
#include "lwt_kthd.h"
#include "lwt.h"
#include "lwt_chan.h"
#include "assert.h"
#include "pthread.h"
#include "faa.h"
#include "stdio.h"
```

Functions

- void ∗ pthread_function (void ∗data)

  Function for the kthd (i.e. pthread) LWT wrapper to perform.
- int lwt_kthd_create (lwt_chan_fn_t fn, lwt_chan_t c, lwt_flags_t flags)

  Creates an N:M kthd.
- struct kthd_event ∗ __pop_from_buffer (lwt_kthd_t kthd)

  Pops a kthd event from the buffer.
- int __push_to_buffer (lwt_kthd_t kthd, struct kthd_event ∗data)

  Pushes a kthd event into the event buffer.
- void __init_kthd (lwt_t lwt)

  Initializes a kthd.
- void ∗ __lwt_buffer (void ∗d)

  Function for the reaper lwt; when all other lwts are blocked, processes events for the kthd.
- lwt_kthd_t __get_kthd ()

  Helper method for returning the current kthd.
- void __init_kthd_event (lwt_t remote_lwt, lwt_chan_t remote_chan, lwt_cgrp_t remote_group, lwt_kthd_t kthd, lwt_remote_op_t remote_op, int block)

  Initializes a kthd event.

Variables

- __thread lwt_kthd_t pthread_kthd
  Pointer to the kthd for the pthread.

4.15.1 Function Documentation

4.15.1.1 lwt_kthd_t __get_kthd ( )

Helper method for returning the current kthd.

Returns

The current kthd

4.15.1.2 void __init_kthd (lwt_t *lwt*)

Initializes a kthd.

Parameters

lwt	The lwt for the kthd

4.15.1.3 void __init_kthd_event (lwt_t *remote_lwt,* lwt_chan_t *remote_chan,* lwt_cgrp_t *remote_group,* lwt_kthd_t *kthd,* lwt_remote_op_t *remote_op,* int *block*)

Initializes a kthd event.

Parameters

remote_lwt	The lwt to modify
remote_chan	The channel to modify
remote_group	The group to modify
kthd	The kthd to modify
remote_op	The operation to perform
block	Is the operation blocking (generally yes; signal is not)

4.15.1.4 void∗ __lwt_buffer (void ∗ *d*)

Function for the reaper lwt; when all other lwts are blocked, processes events for the kthd.

Parameters

d	Data; unused; needed to match file signature

Returns

NULL

4.15.1.5 struct kthd_event∗ __pop_from_buffer (lwt_kthd_t *kthd*)

Pops a kthd event from the buffer.

Parameters

kthd	The kthd to pop

Returns

The kthd event for the action to perform in the reaper function

4.15.1.6 int __push_to_buffer (lwt_kthd_t *kthd,* struct kthd_event ∗ *data*)

Pushes a kthd event into the event buffer.

Parameters

kthd	The kthd to modify
data	The data to insert

Returns

0 if successful; -1 if not

4.15.1.7 int lwt_kthd_create (lwt_chan_fn_t *fn,* lwt_chan_t *c,* lwt_flags_t *flags*)

Creates an N:M kthd.

Parameters

fn	The channel function to run on the remote kthd
c	The channel used as input to that function
flags	The flags for the function

4.15.1.8 void∗ pthread_function (void ∗ *data*)

Function for the kthd (i.e. pthread) LWT wrapper to perform.

Parameters

data	The kthd data used for storing the params for the create chan call

Returns

NULL

4.15.2 Variable Documentation

4.15.2.1 __thread lwt_kthd_t pthread_kthd

Pointer to the kthd for the pthread.

4.16 lwt_kthd.h File Reference

```
#include "objects.h"
```

Functions

- int lwt_kthd_create (lwt_chan_fn_t, lwt_chan_t, lwt_flags_t)

  Creates an N:M kthd.
- void __init_kthd (lwt_t)

  Initializes a kthd.
- void __insert_lwt_into_tail (lwt_kthd_t, lwt_t)
- void __remove_lwt_from_kthd (lwt_kthd_t, lwt_t)
- lwt_kthd_t __get_kthd ()

  Helper method for returning the current kthd.
- void ∗ __lwt_buffer (void ∗)

  Function for the reaper lwt; when all other lwts are blocked, processes events for the kthd.
- void __init_kthd_event (lwt_t, lwt_chan_t, lwt_cgrp_t, lwt_kthd_t, lwt_remote_op_t, int)

  Initializes a kthd event.

4.16.1 Function Documentation

4.16.1.1 lwt_kthd_t __get_kthd ( )

Helper method for returning the current kthd.

Returns

The current kthd

4.16.1.2 void __init_kthd (lwt_t *lwt*)

Initializes a kthd.

Parameters

lwt	The lwt for the kthd

4.16.1.3 void __init_kthd_event (lwt_t *remote_lwt,* lwt_chan_t *remote_chan,* lwt_cgrp_t *remote_group,* lwt_kthd_t *kthd,* lwt_remote_op_t *remote_op,* int *block*)

Initializes a kthd event.

Parameters

remote_lwt	The lwt to modify
remote_chan	The channel to modify
remote_group	The group to modify
kthd	The kthd to modify
remote_op	The operation to perform
block	Is the operation blocking (generally yes; signal is not)

4.16.1.4 void __insert_lwt_into_tail (lwt_kthd_t , lwt_t)

4.16.1.5 void∗ __lwt_buffer (void ∗ *d*)

Function for the reaper lwt; when all other lwts are blocked, processes events for the kthd.

Parameters

d	Data; unused; needed to match file signature

Returns

NULL

4.16.1.6 void __remove_lwt_from_kthd (lwt_kthd_t , lwt_t)

4.16.1.7 int lwt_kthd_create (lwt_chan_fn_t *fn,* lwt_chan_t *c,* lwt_flags_t *flags*)

Creates an N:M kthd.

Parameters

fn	The channel function to run on the remote kthd
c	The channel used as input to that function
flags	The flags for the function

4.17 main.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <assert.h>
#include <sys/wait.h>
#include <pthread.h>
#include "util.h"
#include "server.h"
#include "kthd_server.h"
#include "cas.h"
```

Macros

- #define MAX_DATA_SZ 1024

- #define MAX_CONCURRENCY 4
- #define BUFFER_LENGTH 256

Enumerations

- enum server_type_t { SERVER_TYPE_ONE = 0, SERVER_TYPE_TWO = 1 }

Functions

- void server_single_request (int accept_fd)
- int main (int argc, char ∗argv[ ])

4.17.1 Macro Definition Documentation

4.17.1.1 #define BUFFER_LENGTH 256

4.17.1.2 #define MAX_CONCURRENCY 4

4.17.1.3 #define MAX_DATA_SZ 1024

Redistribution of this file is permitted under the GNU General Public License v2.

Copyright 2012 by Gabriel Parmer. Author: Gabriel Parmer, gparmer@gwu.edu, 2012

4.17.2 Enumeration Type Documentation

4.17.2.1 enum server_type_t

Enumerator

SERVER_TYPE_ONE

SERVER_TYPE_TWO

4.17.3 Function Documentation

4.17.3.1 int main (int *argc,* char ∗ *argv[ ]*)

4.17.3.2 void server_single_request (int *accept_fd*)

4.18 main3.c File Reference

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

Macros

- #define rdtscll(val) __asm__ __volatile__("rdtsc" : "=A" (val))
- #define ITER 10000
- #define IS_RESET()
- #define GRPSZ 3

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)
- void ∗ fn_chan (lwt_chan_t to)
- void test_perf_channels (int chsz)
- void ∗ fn_snder (lwt_chan_t c, int v)
- void ∗ fn_snder_1 (lwt_chan_t c)
- void ∗ fn_snder_2 (lwt_chan_t c)
- void test_multisend (int chsz)
- void ∗ fn_async_steam (lwt_chan_t to)
- void test_perf_async_steam (int chsz)
- void ∗ fn_grpwait (lwt_chan_t c)
- void test_grpwait (int chsz, int grpsz)
- int main (void)

Variables

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

4.18.1 Macro Definition Documentation

4.18.1.1 #define GRPSZ 3

4.18.1.2 #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)
```

4.18.1.3 #define ITER 10000

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

4.18.2 Function Documentation

4.18.2.1 void∗ fn_async_steam (lwt_chan_t *to*)

4.18.2.2 void∗ fn_bounce (void ∗ *d*)

4.18.2.3 void∗ fn_chan (lwt_chan_t *to*)

4.18.2.4 void∗ fn_grpwait (lwt_chan_t *c*)

4.18.2.5 void∗ fn_identity (void ∗ *d*)

4.18.2.6 void∗ fn_join (void ∗ *d*)

4.18.2.7 void∗ fn_nested_joins (void ∗ *d*)

4.18.2.8 void∗ fn_null (void ∗ *d*)

4.18.2.9 void∗ fn_sequence (void ∗ *d*)

4.18.2.10 void∗ fn_snder (lwt_chan_t *c,* int *v*)

4.18.2.11 void∗ fn_snder_1 (lwt_chan_t *c*)

4.18.2.12 void∗ fn_snder_2 (lwt_chan_t *c*)

4.18.2.13 int main (void)

4.18.2.14 void test_crt_join_sched (void)

4.18.2.15 void test_grpwait (int *chsz,* int *grpsz*)

Q: why don't we iterate through all of the data here?

A: We need to fix 1) cevt_wait to be level triggered, or 2) provide a function to detect if there is data available on a channel. Either of these would allows us to iterate on a channel while there is more data pending.

4.18.2.16 void test_multisend (int *chsz*)

4.18.2.17 void test_perf (void)

4.18.2.18 void test_perf_async_steam (int *chsz*)

4.18.2.19 void test_perf_channels (int *chsz*)

4.18.3 Variable Documentation

4.18.3.1 volatile int curr = 0

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

4.19 main_chan.c File Reference

```
#include "lwt.h"
#include "lwt_chan.h"
#include "lwt_cgrp.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 80
- #define MERGE_SZ 80

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.
- void ∗ child_multiple_channels (lwt_chan_t main_channel)
- void multiple_channels_test ()
- void multiple_channels_test_v2 ()
- void multiple_channels_test_v3 ()
- int main ()

4.19.1 Macro Definition Documentation

4.19.1.1 #define ITER 80

4.19.1.2 #define MERGE_SZ 80

4.19.2 Function Documentation

4.19.2.1 void∗ child_multiple_channels (lwt_chan_t *main_channel*)

4.19.2.2 void∗ child_ping (lwt_chan_t *main_channel*)

Ping channel test.

Parameters

main_channel	The channel from the main thread

Returns

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

4.19.2.3 void∗ child_pong (lwt_chan_t *main_channel*)

Receives a count, updates it and sends it back.

Parameters

main_channel	The channel from the main thread

Returns

0 if successful

4.19.2.4 int main (void)

Main function

4.19.2.5 void merge_sort_test ( )

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

4.19.2.6 void∗ msort (lwt_chan_t *main_channel*)

Merge sort in parallel.

Parameters

main_channel	The channel from the main thread

Returns

0 if successful

Note

Adapted from wikipedia: http://en.wikipedia.org/wiki/Merge_sort#Parallel_merge_↩
sort)

4.19.2.7 void multiple_channels_test ( )

4.19.2.8 void multiple_channels_test_v2 ( )

4.19.2.9 void multiple_channels_test_v3 ( )

4.19.2.10 void ping_pong_test ( )

Runs the ping/pong test.

4.20 main_kthd.c File Reference

```
#include "lwt_kthd.h"
#include "lwt_chan.h"
#include "lwt.h"
#include "stdio.h"
#include "assert.h"
```

Macros

- #define MAX_PING_PONG_VALUE 100
- #define ITER 10000
- #define GRPSZ 3

Functions

- void ∗ kthd_ping (lwt_chan_t ping_channel)
- void kthd_ping_pong_sync ()
- void ∗ fn_grpwait (lwt_chan_t c)
- void test_grpwait (int chsz, int grpsz)
- int main ()

4.20.1 Macro Definition Documentation

4.20.1.1 #define GRPSZ 3

4.20.1.2 #define ITER 10000

4.20.1.3 #define MAX_PING_PONG_VALUE 100

4.20.2 Function Documentation

4.20.2.1 void∗ fn_grpwait (lwt_chan_t *c*)

4.20.2.2 void∗ kthd_ping (lwt_chan_t *ping_channel*)

4.20.2.3 void kthd_ping_pong_sync ( )

4.20.2.4 int main (void)

4.20.2.5 void test_grpwait (int *chsz,* int *grpsz*)

Q: why don't we iterate through all of the data here?

A: We need to fix 1) cevt_wait to be level triggered, or 2) provide a function to detect if there is data available on a channel. Either of these would allows us to iterate on a channel while there is more data pending.

4.21 main_orig.c File Reference

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

Macros

- #define rdtscll(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

4.21.1 Macro Definition Documentation

4.21.1.1 #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)
```

4.21.1.2 #define ITER 10000

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

4.21.2 Function Documentation

4.21.2.1 void∗ fn_bounce (void ∗ *d*)

4.21.2.2 void∗ fn_identity (void ∗ *d*)

4.21.2.3 void∗ fn_join (void ∗ *d*)

4.21.2.4 void∗ fn_nested_joins (void ∗ *d*)

4.21.2.5 void∗ fn_null (void ∗ *d*)

4.21.2.6 void∗ fn_sequence (void ∗ *d*)

4.21.2.7 int main (void)

4.21.2.8 void test_crt_join_sched (void)

4.21.2.9 void test_perf (void)

4.21.3 Variable Documentation

4.21.3.1 volatile int curr = 0

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

4.22 objects.h File Reference

```
#include "pthread.h"
#include "stdlib.h"
#include <sys/queue.h>
#include "enums.h"
```

Data Structures

- struct lwt_cgrp_t

  Channel group for handling events within a group.
- struct lwt_chan_t

  The channel for synchronous and asynchronous communication.
- struct kthd_event
- struct lwt_kthd_t
- struct lwt_kthd_data
- struct lwt_t

  The Lightweight Thread (LWT) struct.

Macros

- #define EVENT_BUFFER_SIZE 10000
- #define PAGE_SIZE 4096
- #define NUM_PAGES 5
- #define STACK_SIZE PAGE_SIZE∗NUM_PAGES
- #define DEBUG 1
- #define LWT_NULL NULL

Typedefs

- typedef void ∗(∗ lwt_chan_fn_t)(lwt_chan_t)
- typedef void ∗(∗ lwt_fnt_t)(void ∗)

4.22.1 Macro Definition Documentation

4.22.1.1 #define DEBUG 1

4.22.1.2 #define EVENT_BUFFER_SIZE 10000

Size of the event buffer

4.22.1.3 #define LWT_NULL NULL

Null id for yields

4.22.1.4 #define NUM_PAGES 5

Number of pages to allocate to the stack

4.22.1.5 #define PAGE_SIZE 4096

Size of the a page in the OS -> 4K

4.22.1.6 #define STACK_SIZE PAGE_SIZE∗NUM_PAGES

Size of the stack

4.22.2 Typedef Documentation

4.22.2.1 typedef void∗(∗ lwt_chan_fn_t)(lwt_chan_t)

4.22.2.2 typedef void∗(∗ lwt_fnt_t)(void ∗)

4.23 server.c File Reference

```
#include <sys/types.h>
#include <sys/socket.h>
#include <stdio.h>
#include <errno.h>
#include <netinet/in.h>
#include <fcntl.h>
#include <arpa/inet.h>
#include <stdlib.h>
#include <unistd.h>
```

Functions

- int server_create (short int port)
- int server_accept (int fd)

4.23.1 Function Documentation

4.23.1.1 int server_accept (int *fd*)

4.23.1.2 int server_create (short int *port*)

4.24 server.h File Reference

Functions

- int server_create (short int port)
- int server_accept (int fd)

4.24.1 Function Documentation

4.24.1.1 int server_accept (int *fd*)

4.24.1.2 int server_create (short int *port*)

4.25 simple_http.c File Reference

```
#include <string.h>
```

```
#include <assert.h>
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include "simple_http.h"
```

Macros

- #define MAX_DIGITS 128

Functions

- struct http_req ∗ shttp_alloc_req (int fd, char ∗request)
- void shttp_free_req (struct http_req ∗r)
- int shttp_get_path (struct http_req ∗r)
- int shttp_alloc_response_head (struct http_req ∗r, char ∗data, int dlen)

4.25.1 Macro Definition Documentation

4.25.1.1 #define MAX_DIGITS 128

4.25.2 Function Documentation

4.25.2.1 struct **http_req**∗ shttp_alloc_req (int *fd,* char ∗ *request*)

4.25.2.2 int shttp_alloc_response_head (struct **http_req** ∗ *r,* char ∗ *data,* int *dlen*)

4.25.2.3 void shttp_free_req (struct **http_req** ∗ *r*)

4.25.2.4 int shttp_get_path (struct **http_req** ∗ *r*)

4.26 simple_http.h File Reference

Data Structures

- struct http_req

Functions

- struct http_req ∗ shttp_alloc_req (int fd, char ∗request)
- void shttp_free_req (struct http_req ∗r)
- int shttp_get_path (struct http_req ∗r)
- int shttp_alloc_response_head (struct http_req ∗r, char ∗resp, int rlen)

4.26.1 Function Documentation

4.26.1.1 struct **http_req**∗ shttp_alloc_req (int *fd,* char ∗ *request*)

Redistribution of this file is permitted under the GNU General Public License v2.

Copyright 2012 by Gabriel Parmer. Author: Gabriel Parmer, gparmer@gwu.edu, 2012

4.26.1.2 int shttp_alloc_response_head (struct **http_req** ∗ *r,* char ∗ *resp,* int *rlen*)

4.26.1.3 void shttp_free_req (struct **http_req** ∗ *r*)

4.26.1.4 int shttp_get_path (struct **http_req** ∗ *r*)

4.27 util.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <assert.h>
#include "server.h"
#include "simple_http.h"
#include "content.h"
```

Macros

- #define MAX_REQ_SZ 1024

Functions

- struct http_req ∗ newfd_create_req (int new_fd)
- void respond_and_free_req (struct http_req ∗r, char ∗response, int len)
- void client_process (int fd)

4.27.1 Macro Definition Documentation

4.27.1.1 #define MAX_REQ_SZ 1024

Redistribution of this file is permitted under the GNU General Public License v2.

Copyright 2012 by Gabriel Parmer. Author: Gabriel Parmer, gparmer@gwu.edu, 2012

4.27.2 Function Documentation

4.27.2.1 void client_process (int *fd*)

Redistribution of this file is permitted under the GNU General Public License v2.

Copyright 2012 by Gabriel Parmer. Author: Gabriel Parmer, gparmer@gwu.edu, 2012

4.27.2.2 struct **http_req**∗ newfd_create_req (int *new_fd*)

4.27.2.3 void respond_and_free_req (struct **http_req** ∗ *r,* char ∗ *response,* int *len*)

4.28 util.h File Reference

Functions

- void client_process (int fd)

4.28.1 Function Documentation

4.28.1.1 void client_process (int *fd*)

Index