Generic Lightweight Thread (GLT) Library

2.5

Generated by Doxygen 1.8.6

Wed Jul 20 2016 12:36:28

Contents

Chapter 1

GLT_documentation

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Module Documentation

3.1 Library functions

Functions

- void __attribute__ ((constructor)) glt_start(void)

  Entry point for the GLT dynamic library.
- void __attribute__ ((destructor)) glt_end(void)

  Ending point for the GLT dynamic library.
- void glt_init (int argc, char *argv[])

  GLT initialization function.
- void glt_finalize ()

  GLT finalization function.

3.1.1 Detailed Description

These functions start/stop and open/close the underlying GLT libraries. They are used in dynamic and static implementations.

3.1.2 Function Documentation

3.1.2.1 void __attribute__ ((constructor))

Entry point for the GLT dynamic library.

`glt_start()` is the first called function when the GLT dynamic library is loaded

3.1.2.2 void __attribute__ ((destructor))

Ending point for the GLT dynamic library.

`glt_end()` is the last called function when the GLT dynamic library is unloaded

3.1.2.3 void glt_finalize ( )

GLT finalization function.

`glt_finalize()` destroys the GLT environment. It is not mandatory and should be the last GLT function call.

3.1.2.4 void glt_init (int *argc*, char * *argv[]*)

GLT initialization function.

`glt_init()` sets the GLT environment up. It is mandatory and needs to be the first GLT function call.

Parameters

in	*argc*	
in	*argv*	

3.2 Barrier functions

Functions

- void glt_barrier_create (int num_waiters, GLT_barrier *barrier)

  Creates a barrier.
- void glt_barrier_free (GLT_barrier *barrier)

  Destroys a barrier.
- void glt_barrier_wait (GLT_barrier *barrier)

  Waits into a barrier.

3.2.1 Detailed Description

These functions manage the GLT barriers for the ULTs.

3.2.2 Function Documentation

3.2.2.1 void glt_barrier_create (int *num_waiters*, GLT_barrier * *barrier*)

Creates a barrier.

`glt_barrier_create()` creates a barrier for ULTs.

Parameters

in	*num_waiters*	Indicates the number of ULTs requested to continue
in,out	*barrier*	Hande to newly created `GLT_barrier`

3.2.2.2 void glt_barrier_free (GLT_barrier * *barrier*)

Destroys a barrier.

`glt_barrier_free()` destroys a barier for ULTs.

Parameters

in	*barrier*	Handle to the target `GLT_barrier`.

3.2.2.3 void glt_barrier_wait (GLT_barrier * *barrier*)

Waits into a barrier.

`glt_barrier_wait()` Executed by a ULT, it waits until the number of waiters is achieved.

Parameters

in	*barrier*	Handle to the target `GLT_barrier`.

3.3 Condition functions

Functions

- void glt_cond_create (GLT_cond *cond)
  Creates a condition.
- void glt_cond_free (GLT_cond *cond)
  Destroys a condition.
- void glt_cond_signal (GLT_cond cond)
  Sends a signal for a condition.
- void glt_cond_wait (GLT_cond cond, GLT_mutex mutex)
  A ULT waits in this point for a signal.
- void glt_cond_broadcast (GLT_cond cond)
  Broadcast a signal for a condition.

3.3.1 Detailed Description

These functions manage the GLT conditions for the ULTs.

3.3.2 Function Documentation

3.3.2.1 void glt_cond_broadcast (GLT_cond *cond*)

Broadcast a signal for a condition.

`glt_cond_broadcast()` broadcasts a signal for ULTs.

Parameters

in	*cond*	Handle to the target `GLT_condition`.

3.3.2.2 void glt_cond_create (GLT_cond * *cond*)

Creates a condition.

`glt_cond_create()` creates a condition for ULTs.

Parameters

in,out	*cond*	Hande to newly created `GLT_condition`

3.3.2.3 void glt_cond_free (GLT_cond * *cond*)

Destroys a condition.

`glt_cond_free()` destroys a condition for ULTs.

Parameters

in	*cond*	Handle to the target `GLT_condition`.

3.3.2.4 void glt_cond_signal (GLT_cond *cond*)

Sends a signal for a condition.

`glt_cond_signal()` sends a signal for ULTs.

Parameters

in	*cond*	Handle to the target GLT_condition.

3.3.2.5 void glt_cond_wait (GLT_cond *cond*, GLT_mutex *mutex*)

A ULT waits in this point for a signal.

glt_cond_wait() a ULT waits at this point for a signal to access the mutex.

Parameters

in	*cond*	Handle to the target GLT_condition.
in	*mutex*	Handle to the target GLT_mutex.

3.4 Mutex functions

Functions

- void glt_mutex_create (GLT_mutex *mutex)
  Creates a mutex.
- void glt_mutex_lock (GLT_mutex mutex)
  Locks a mutex.
- void glt_mutex_unlock (GLT_mutex mutex)
  Unlocks a mutex.
- void glt_mutex_free (GLT_mutex *mutex)
  Destroys a mutex.
- void glt_mutex_trylock (GLT_bool *locked, GLT_mutex mutex)
  Tries to lock a mutex.

3.4.1 Detailed Description

These functions manage the GLT mutexes for the ULTs.

3.4.2 Function Documentation

3.4.2.1 void glt_mutex_create (GLT_mutex * *mutex*)

Creates a mutex.

`glt_mutex_create()` creates a mutex for ULTs.

Parameters

`in,out`	*mutex*	Hande to newly created `GLT_mutex`

3.4.2.2 void glt_mutex_free (GLT_mutex * *mutex*)

Destroys a mutex.

`glt_mutex_free()` destroys a mutex for ULTs.

Parameters

`in`	*mutex*	Handle to the target `GLT_mutex`.

3.4.2.3 void glt_mutex_lock (GLT_mutex *mutex*)

Locks a mutex.

`glt_mutex_lock()` locks (if possible) a mutex.

Parameters

`in`	*mutex*	Handle to the target `GLT_mutex`.

3.4.2.4 void glt_mutex_trylock (GLT_bool * *locked*, GLT_mutex *mutex*)

Tries to lock a mutex.

`glt_mutex_trylock()` tries to lock a mutex.

Parameters

in	*mutex*	Handle to the target GLT_mutex.
out	*locked*	GLT_bool with the value 1 if the mutex has been locked or 0 if it was not possible.

3.4.2.5 void glt_mutex_unlock (GLT_mutex *mutex*)

Unlocks a mutex.

glt_mutex_unlock() unlocks a mutex.

Parameters

in	*mutex*	Handle to the target GLT_mutex.

3.5 Work-units functions

Functions

- GLT_ult * glt_ult_malloc (int number_of_ult)
  ULT allocation.
- GLT_tasklet * glt_tasklet_malloc (int number_of_tasklets)
  ULT allocation.
- void glt_ult_create (void(*thread_func)(void *), void *arg, GLT_ult *new_ult)
  ULT creation.
- void glt_ult_create_to (void(*thread_func)(void *), void *arg, GLT_ult *new_ult, int dest)
  ULT creation in a given destination.
- void glt_tasklet_create (void(*thread_func)(void *), void *arg, GLT_tasklet *new_ult)
  Tasklet creation.
- void glt_tasklet_create_to (void(*thread_func)(void *), void *arg, GLT_tasklet *new_ult, int dest)
  Tasklet creation.
- void glt_yield ()
  ULT yields to another ready ULT.
- void glt_yield_to (GLT_ult ult)
  ULT yields to an specific ULT.
- void glt_ult_join (GLT_ult *ult)
  Joins an specific ULT.
- void glt_tasklet_join (GLT_tasklet *tasklet)
  Joins an specific Tasklet.
- void glt_ult_get_id (GLT_ult_id *id, GLT_ult ult)
  Return the unique id of a ULT.
- void glt_workunit_get_thread_id (GLT_thread_id *id)
  Return the unique id of a thread.
- void glt_ult_migrate_self_to (GLT_thread_id id)
  Migrates the current ULT to another thread ready queue.
- void glt_ult_self (GLT_ult *ult)
  Obtains the current ULT handle.
- void glt_tasklet_self (GLT_tasklet *tasklet)
  Obtains the current Tasklet handle.
- void glt_ult_cancel (GLT_ult ult)
  Cancels an specific ULT.
- void glt_tasklet_cancel (GLT_tasklet tasklet)
  Cancels an specific Tasklet.
- void glt_ult_exit ()
  Exits the current ULT.

3.5.1 Detailed Description

These functions create, map, schedule, join, and execute work-units.

3.5.2 Function Documentation

3.5.2.1 void glt_tasklet_cancel (GLT_tasklet *tasklet*)

Cancels an specific Tasklet.

`glt_tasklet_cancel()` cancels a given `GLT_tasklet`.

Parameters

in	*tasklet*	Handle to the target GLT_tasklet.

3.5.2.2 void glt_tasklet_create (void(*)(void *) *thread_func*, void * *arg*, GLT_tasklet * *new_ult*)

Tasklet creation.

glt_tasklet_create() creates a GLT_tasklet.

Parameters

in	*thread_func*	Is the function pointer to be executed by the GLT_tasklet.
in	*arg*	Are the arguments for thread_func.
out	*new_ult*	Handle to a newly created GLT_tasklet.

3.5.2.3 void glt_tasklet_create_to (void(*)(void *) *thread_func*, void * *arg*, GLT_tasklet * *new_ult*, int *dest*)

Tasklet creation.

glt_tasklet_create() creates a GLT_tasklet.

Parameters

in	*thread_func*	Is the function pointer to be executed by the GLT_tasklet.
in	*arg*	Are the arguments for thread_func.
out	*new_ult*	Handle to a newly created GLT_tasklet.
in	*dest*	Number of the GLT_thread that should execute the newly created GLT_tasklet.

3.5.2.4 void glt_tasklet_join (GLT_tasklet * *tasklet*)

Joins an specific Tasklet.

glt_tasklet_join() joins a given GLT_tasklet.

Parameters

in	*tasklet*	Handle to the target GLT_tasklet.

3.5.2.5 GLT_tasklet* glt_tasklet_malloc (int *number_of_tasklets*)

ULT allocation.

glt_tasklet_malloc() allocates memory for a given number of GLT_tasklet.

Parameters

in	*number_of_tasklets*	Indicates the total number of GLT_tasklets that needs to be allocated.

Returns

The pointer to the allocated memory.

3.5.2.6 void glt_tasklet_self (GLT_tasklet * *tasklet*)

Obtains the current Tasklet handle.

`glt_tasklet_self()` returns the current `GLT_tasklet` handler.

Parameters

out	*tasklet*	Handler of the the current GLT_tasklet.

3.5.2.7 void glt_ult_cancel (GLT_ult *ult*)

Cancels an specific ULT.

glt_ult_cancel() cancels a given GLT_ult.

Parameters

in	*ult*	Handle to the target GLT_ult.

3.5.2.8 void glt_ult_create (void(∗)(void ∗) *thread_func,* void ∗ *arg,* GLT_ult ∗ *new_ult*)

ULT creation.

glt_ult_create() creates a GLT_ult.

Parameters

in	*thread_func*	Is the function pointer to be executed by the GLT_ult.
in	*arg*	Are the arguments for thread_func.
out	*new_ult*	Handle to a newly created GLT_ult.

3.5.2.9 void glt_ult_create_to (void(∗)(void ∗) *thread_func,* void ∗ *arg,* GLT_ult ∗ *new_ult,* int *dest*)

ULT creation in a given destination.

glt_ult_create_to() creates a GLT_ult in a particular destination.

Parameters

in	*thread_func*	Is the function pointer to be executed by the GLT_ult.
in	*arg*	Are the arguments for thread_func.
out	*new_ult*	Handle to a newly created GLT_ult.
in	*dest*	Number of the GLT_thread that should execute the newly created GLT_ult.

3.5.2.10 void glt_ult_exit ( )

Exits the current ULT.

glt_ult_exit() cancels the current GLT_ult.

3.5.2.11 void glt_ult_get_id (GLT_ult_id ∗ *id,* GLT_ult *ult*)

Return the unique id of a ULT.

glt_ult_get_id() returns the id of a given GLT_ult.

Parameters

in	*ult*	Handle to the target GLT_ult.
out	*id*	Identifier if the the target GLT_ult.

3.5.2.12 void glt_ult_join (GLT_ult * *ult*)

Joins an specific ULT.

glt_ult_join() joins a given GLT_ult.

Parameters

in	*ult*	Handle to the target GLT_ult.

3.5.2.13 GLT_ult* glt_ult_malloc (int *number_of_ult*)

ULT allocation.

glt_ult_malloc() allocates memory for a given number of GLT_ult.

Parameters

in	*number_of_ult*	Indicates the total number of GLT_ult that needs to be allocated.

Returns

The pointer to the allocated memory.

3.5.2.14 void glt_ult_migrate_self_to (GLT_thread_id *id*)

Migrates the current ULT to another thread ready queue.

glt_ult_migrate_self_to() moves the current GLT_ult to another GLT_thread ready queue.

Parameters

in	*The*	identifier of the the GLT_thread destination.

3.5.2.15 void glt_ult_self (GLT_ult * *ult*)

Obtains the current ULT handle.

glt_ult_self() returns the current GLT_ult handler.

Parameters

out	*ult*	Handler of the the current GLT_ult.

3.5.2.16 void glt_workunit_get_thread_id (GLT_thread_id * *id*)

Return the unique id of a thread.

glt_workunit_get_thread_id() returns the id of the current GLT_thread.

Parameters

out	*id*	Identifier of the the current GLT_thread.

3.5.2.17 void glt_yield ( )

ULT yields to another ready ULT.

glt_yield() puts the current GLT_ult in the ready queue and allows another ready GLT_ult to be executed.

3.5.2.18 void glt_yield_to (GLT_ult *ult*)

ULT yields to an specific ULT.

glt_yield_to() puts the current GLT_ult in the ready queue and allows an specific ready GLT_ult to be executed.

Parameters

in	*ult*	Handle to the target GLT_ult.

3.6 Timer functions

Functions

- double glt_get_wtime ()

  Returns the current time.
- void glt_timer_create (GLT_timer *timer)

  Creates a timer.
- void glt_timer_free (GLT_timer *timer)

  Destroys a timer.
- void glt_timer_start (GLT_timer timer)

  Inits a timer.
- void glt_timer_stop (GLT_timer timer)

  Stops a timer.
- void glt_timer_get_secs (GLT_timer timer, double *secs)

  Obtains the elapsed time.

3.6.1 Detailed Description

These functions simplify the use of timers.

3.6.2 Function Documentation

3.6.2.1 double glt_get_wtime ( )

Returns the current time.

`glt_get_wtime()` returns the time.

Returns

The time in seconds.

3.6.2.2 void glt_timer_create (GLT_timer * *timer*)

Creates a timer.

`glt_timer_create()` creates a timer.

Parameters

`in,out`	*timer*	Hande to newly created `GLT_timer`.

3.6.2.3 void glt_timer_free (GLT_timer * *timer*)

Destroys a timer.

`glt_timer_free()` destroys a timer.

Parameters

in	*timer*	Handle to the target GLT_timer.

3.6.2.4 void glt_timer_get_secs (GLT_timer *timer*, double * *secs*)

Obtains the elapsed time.

glt_timer_get_secs() given a timer. It calculates the elapsed time in seconds.

Parameters

in	*timer*	Handle to the target GLT_timer.
out	*secs*	Seconds.

3.6.2.5 void glt_timer_start (GLT_timer *timer*)

Inits a timer.

glt_timer_start() inits a timer.

Parameters

in	*timer*	Handle to the target GLT_timer.

3.6.2.6 void glt_timer_stop (GLT_timer *timer*)

Stops a timer.

glt_timer_stop() stops a timer.

Parameters

in	*timer*	Handle to the target GLT_timer.

3.7 Util functions

Functions

- int glt_get_thread_num ()

  Obtains the number of the current thread.
- int glt_get_num_threads ()

  Returns the total number of threads.

3.7.1 Detailed Description

These functions return values from the environment set up.

3.7.2 Function Documentation

3.7.2.1 int glt_get_num_threads ( )

Returns the total number of threads.

`glt_get_num_threads()` returns the number threads.

Returns

The number of `GLT_threads`.

3.7.2.2 int glt_get_thread_num ( )

Obtains the number of the current thread.

`glt_get_thread_num()` returns the number of the current thread.

Returns

The number of the current `GLT_thread`.

3.8 Scheduler functions

Functions

- void glt_scheduler_config_free (GLT_sched_config ∗config)

  Destroys the scheduler configuration.
- void glt_scheduler_create (GLT_sched_def ∗def, int num_threads, int ∗threads_id, GLT_sched_config config, GLT_sched ∗newsched)

  Creates a new scheduler.
- void glt_schededuler_create_basic (GLT_sched_predef predef, int num_threads, int ∗threads_id, GLT_sched_config config, GLT_sched ∗newsched)

  Creates a new scheduler with predefined behaviour.
- void glt_scheduler_free (GLT_sched ∗sched)

  Destroys a scheduler.
- void glt_scheduler_finish (GLT_sched sched)

  Finalizes a scheduler.
- void glt_scheduler_exit (GLT_sched sched)

  Stops a scheduler.
- void glt_scheduler_has_to_stop (GLT_sched sched, GLT_bool ∗stop)

  Requires to a scheduler to stop.
- void glt_scheduler_set_data (GLT_sched sched, void ∗data)

  Sets new data to a scheduler.
- void glt_scheduler_get_data (GLT_sched sched, void ∗∗data)

  gets data from a scheduler.
- void glt_scheduler_get_size (GLT_sched sched, size_t ∗size)

  gets the current size from the scheduler.
- void glt_scheduler_get_total_size (GLT_sched sched, size_t ∗size)

  gets the total size from the scheduler.

3.8.1 Detailed Description

These functions manages the configurable scheduler (just with Argobots).

3.8.2 Function Documentation

3.8.2.1 void glt_schededuler_create_basic (GLT_sched_predef *predef,* int *num_threads,* int ∗ *threads_id,* GLT_sched_config *config,* GLT_sched ∗ *newsched*)

Creates a new scheduler with predefined behaviour.

`glt_schededuler_create_basic()` creates a new scheduler for some threads with a predefined behaviour.

Parameters

in	*def*	Handle of the target `GLT_sched_predef`.
in	*num_threads*	number of `GLT_thread` affected by this scheduler.
in	*threads_id*	pointer to an array of `GLT_threads_id`.
in	*config*	Handle of the target `GLT_sched_config`.

out	*newsched*	Handle of new GLT_sched.

3.8.2.2 void glt_scheduler_config_free (GLT_sched_config * *config*)

Destroys the scheduler configuration.

glt_scheduler_config_free() deletes the scheduler configuration.

Parameters

in	*config*	Handle of the target GLT_sched_config.

3.8.2.3 void glt_scheduler_create (GLT_sched_def * *def*, int *num_threads*, int * *threads_id*, GLT_sched_config *config*, GLT_sched * *newsched*)

Creates a new scheduler.

glt_scheduler_create() creates a new scheduler for some threads.

Parameters

in	*def*	Handle of the target GLT_sched_def.
in	*num_threads*	number of GLT_thread affected by this scheduler.
in	*threads_id*	pointer to an array of GLT_threads_id.
in	*config*	Handle of the target GLT_sched_config.
out	*newsched*	Handle of new GLT_sched.

3.8.2.4 void glt_scheduler_exit (GLT_sched *sched*)

Stops a scheduler.

glt_scheduler_exit() Stops a scheduler.

Parameters

in	*sched*	Handle of the target GLT_sched.

3.8.2.5 void glt_scheduler_finish (GLT_sched *sched*)

Finalizes a scheduler.

glt_scheduler_finish() finalizes a scheduler.

Parameters

in	*sched*	Handle of the target GLT_sched.

3.8.2.6 void glt_scheduler_free (GLT_sched * *sched*)

Destroys a scheduler.

glt_scheduler_free() destroys a scheduler.

Parameters

in	*sched*	Handle of the target GLT_sched.

3.8.2.7 void glt_scheduler_get_data (GLT_sched *sched*, void ** *data*)

gets data from a scheduler.

glt_scheduler_get_data() gets data from a scheduler.

Parameters

in	*sched*	Handle of the target GLT_sched.
out	*data*	obtained.

3.8.2.8 void glt_scheduler_get_size (GLT_sched *sched*, size_t * *size*)

gets the current size from the scheduler.

glt_scheduler_get_size() gets size from a scheduler.

Parameters

in	*sched*	Handle of the target GLT_sched.
out	*size*	obtained.

3.8.2.9 void glt_scheduler_get_total_size (GLT_sched *sched*, size_t * *size*)

gets the total size from the scheduler.

glt_scheduler_get_total_size() gets the total size from a scheduler.

Parameters

in	*sched*	Handle of the target GLT_sched.
out	*size*	obtained.

3.8.2.10 void glt_scheduler_has_to_stop (GLT_sched *sched*, GLT_bool * *stop*)

Requires to a scheduler to stop.

glt_scheduler_has_to_stop() Requires a scheduler to stop.

Parameters

in	*sched*	Handle of the target GLT_sched.
out	*stop*	shows the answer of the scheduler.

3.8.2.11 void glt_scheduler_set_data (GLT_sched *sched*, void * *data*)

Sets new data to a scheduler.

glt_scheduler_set_data() Sets data to a scheduler.

Parameters

in	*sched*	Handle of the target GLT_sched.
in	*data*	to be set.

3.9 Key functions

Functions

- void glt_key_create (void(*destructor)(void *value), GLT_key *newkey)
  Creates a key.
- void glt_key_free (GLT_key *key)
  Destroys a key.
- void glt_key_set (GLT_key key, void *value)
  Sets new value to a key.
- void glt_key_get (GLT_key key, void **value)
  Gets value from a key.

3.9.1 Detailed Description

These functions manage the GLT keys for the ULTs.

3.9.2 Function Documentation

3.9.2.1 void glt_key_create (void(*)(void *value) *destructor,* GLT_key * *newkey*)

Creates a key.

`glt_key_create()` creates a key.

Parameters

in	*destructor*	Hande to newly created `GLT_ult`.
out	*newkey*	Hande to newly created `GLT_key`.

3.9.2.2 void glt_key_free (GLT_key * *key*)

Destroys a key.

`glt_key_free()` destroys a key for ULTs.

Parameters

in	*key*	Handle to the target `GLT_key`.

3.9.2.3 void glt_key_get (GLT_key *key,* void ** *value*)

Gets value from a key.

`glt_key_get()` Gets value from a key.

Parameters

in	*key*	Handle of the target `GLT_key`.
out	*value*	obtained value.

3.9.2.4 void glt_key_set (GLT_key *key,* void * *value*)

Sets new value to a key.

`glt_key_set()` Sets value to a key.

Parameters

in	*key*	Handle of the target GLT_key.
in	*value*	to be set.

Index