OpenMP Functions (2A)

Young Won Lim
2/28/19

Based on

https://en.wikipedia.org/wiki/OpenMP

https://es.wikipedia.org/wiki/OpenMP

Functions (0)

The OpenMP API has a series of functions
to obtain information and configure the parallel environment.

It also allows to handle locks and take measurements of time.

In addition to the functions shown here, each OpenMP implementation has
its own environment variables and functions.

https://es.wikipedia.org/wiki/OpenMP

Functions (A)

omp_destroy_lock	Uninitializes a lock.
omp_destroy_nest_lock	Uninitializes a nestable lock.
omp_get_dynamic	Returns a value that indicates if the number of threads available in upcoming parallel regions can be adjusted by the run time.
omp_get_max_threads	Returns an integer that is equal to or greater than the number of threads that would be available if a parallel region without num_threads were defined at that point in the code.
omp_get_nested	Returns a value that indicates if nested parallelism is enabled.
omp_get_num_procs	Returns the number of processors that are available when the function is called.
omp_get_num_threads	Returns the number of threads in the parallel region.
omp_get_thread_num	Returns the thread number of the thread executing within its thread team.

https://docs.microsoft.com/en-us/cpp/parallel/openmp/reference/openmp-functions?view=vs-2017

Functions (B)

omp_get_wtick	Returns the number of seconds between processor clock ticks.
omp_get_wtime	Returns a value in seconds of the time elapsed from some point.
omp_in_parallel	Returns nonzero if called from within a parallel region.
omp_init_lock	Initializes a simple lock.
omp_init_nest_lock	Initializes a lock.
omp_set_dynamic	Indicates that the number of threads available in upcoming parallel regions can be adjusted by the run time.
omp_set_lock	Blocks thread execution until a lock is available.
omp_set_nest_lock	Blocks thread execution until a lock is available.
omp_set_nested	Enables nested parallelism.
omp_set_num_threads	Sets the number of threads in upcoming parallel regions, unless overridden by a num_threads clause.

https://docs.microsoft.com/en-us/cpp/parallel/openmp/reference/openmp-functions?view=vs-2017

Functions (C)

omp_test_lock	Attempts to set a lock but doesn't block thread execution.
omp_test_nest_lock	Attempts to set a nestable lock but doesn't block thread execution.
omp_unset_lock	Releases a lock.
omp_unset_nest_lock	Releases a nestable lock.

https://docs.microsoft.com/en-us/cpp/parallel/openmp/reference/openmp-functions?view=vs-2017

Functions (1)

omp_get_active_level() :

Returns the number of nested, active parallel regions enclosing the task that contains the call.

omp_get_ancestor_thread_num(int level) :

Returns the thread number of the ancestor of the current thread at a given nested level

omp_get_cancellation() : Does the implementation of OpenMP support in use? Controlled by the environment variable OMP_CANCELLATION .

omp_get_default_device() : Get the device by default in target regions that do not have a device clause.

omp_get_dynamic() : Is the dynamic size team creation functionality active? Value controlled by environment variable OMP_DYNAMIC or omp_set_dynamic() .

omp_get_level() : Obtain the nesting level at the moment of invoking the function.

omp_get_max_active_levels() : What is the maximum number of active parallel regions allowed?

omp_get_max_task_priority() : What is the maximum priority allowed for an explicit task?

omp_get_max_threads() : What is the maximum number of threads in the current parallel region, which does not use num_threads ?

https://es.wikipedia.org/wiki/OpenMP

Functions (2)

omp_get_nested() : Are nested parallel regions allowed? Value controlled by the **OMP_NESTED** environment **OMP_NESTED** or **omp_set_nested()** function.

omp_get_num_devices() : How many accelerator devices are there?

omp_get_num_procs() : How many processors (cores) are available in the current device?

omp_get_num_teams() : How many thread units are in the current parallel region?

omp_get_num_threads() : How many threads make up the current active thread equipment? In a sequential section, it will return 1. The default size can be initialized with the variable OMP_NUM_THREADS . During execution, you can use the num_threads clause or the omp_set_num_threads(int num_threads) . If none of the above is used and the value of OMP_DYNAMIC is deactivated, a total of threads corresponding to the total of available processors is configured.

omp_get_proc_bind() : Are the threads anchored to specific processors? The function will return the type of anchor (binding), which can be one of the following: omp_proc_bind_false , omp_proc_bind_true ,

https://es.wikipedia.org/wiki/OpenMP

Functions (3)

omp_proc_bind_master , **omp_proc_bind_close** or **omp_proc_bind_spread** . The type of anchor is established at the start of the program with the environment variable OMP_PROC_BIND .

omp_get_schedule() : What is the default planning method?

omp_get_team_num() : What is the identifier of the current thread equipment?

omp_get_team_size() : What is the size of the active thread equipment?

omp_get_thread_limit() : What is the maximum number of threads allowed?

omp_get_thread_num() : What is the identifier within the active team of the thread that invokes the function?

omp_in_parallel() : Is the thread within a parallel region?

omp_in_final() : Are we within an explicit task (task) or is the region of the explicit task running as final ?

omp_is_initial_device() : At the moment of invoking the function, are we in the host or in an accelerator?

https://es.wikipedia.org/wiki/OpenMP

Functions (4)

omp_set_default_device(int device_num) : Configures the device by default for target regions that do not have a device clause.

omp_set_dynamic(int dynamic_threads) : Activate / deactivate the dynamic regulation of the size of the wire equipment.

omp_set_max_active_levels(int max_levels) : Limits the number of active parallel regions.

omp_set_nested(int nested) : Allow or not nested parallel regions.

omp_set_num_threads(int num_threads) : Sets the maximum number of threads in a thread machine.

omp_set_schedule(omp_sched_t kind, int chunk_size) : Sets the default schedule method. The value of kind can be omp_sched_static , omp_sched_dynamic , omp_sched_guided or omp_sched_auto .

https://es.wikipedia.org/wiki/OpenMP

Functions (5)

omp_init_lock(omp_lock_t *lock) : Initialize a padlock.

omp_set_lock(omp_lock_t *lock) : Wait for the padlock to be inactive and activate it for the invoking thread.

omp_test_lock(omp_lock_t *lock) : Check if the padlock is inactive and activate it for the calling thread if the check is positive.

omp_unset_lock(omp_lock_t *lock) : Disable a simple padlock.

omp_destroy_lock(omp_lock_t *lock) : Destroy a simple padlock.

omp_init_nest_lock(omp_lock_t *lock) : Initialize a nested padlock.

omp_set_nest_lock(omp_lock_t *lock) : Wait until the nested padlock is inactive and activate it for the invoking thread.

omp_test_nest_lock(omp_lock_t *lock) : Check if the padlock is inactive and activate it for the calling thread if the check is positive.

omp_unset_nest_lock(omp_lock_t *lock) : Disable a nested padlock.

omp_destroy_nest_lock(omp_lock_t *lock) : Destroy a nested lock.

https://es.wikipedia.org/wiki/OpenMP

Functions (7)

omp_get_wtick : Get the accuracy of the stopwatch: how many seconds pass between two clock ticks.

omp_get_wtime : Get the timer time. The time corresponds to the real one, that is, program execution time, not process time. The process time would be the sum of the activity times of all the threads. The unit of measure are the seconds. There is no guarantee that two different threads will give the same time.

https://es.wikipedia.org/wiki/OpenMP

omp_set_num_threads

Affects the number of threads used for subsequent parallel regions
not specifying a num_threads clause,
by setting the value of the first element of the nthreads-var ICV
of the current task to num_threads.

void omp_set_num_threads(int num_threads);

https://www.openmp.org/wp-content/uploads/OpenMP-4.0-C.pdf

omp_get_num_threads

Returns the number of threads in the current team.

The binding region for an omp_get_num_threads region

is the innermost enclosing parallel region.

int omp_get_num_threads(void);

https://www.openmp.org/wp-content/uploads/OpenMP-4.0-C.pdf

omp_get_max_threads

Returns an upper bound on the number of threads that could be used
to form a new team if a parallel construct without a num_threads clause
were encountered after execution returns from this routine.

int omp_get_max_threads(void);

https://www.openmp.org/wp-content/uploads/OpenMP-4.0-C.pdf

omp_get_thread_num

Returns the thread number of the calling thread within the current team.

int omp_get_thread_num(void);

https://www.openmp.org/wp-content/uploads/OpenMP-4.0-C.pdf

omp_get_num_procs

Returns the number of processors that are available

to the device at the time the routine is called.

int omp_get_num_procs(void);

https://www.openmp.org/wp-content/uploads/OpenMP-4.0-C.pdf

omp_in_parallel

Returns true if the active-levels-var ICV is greater than zero;

otherwise it returns false.

int omp_in_parallel(void);

https://www.openmp.org/wp-content/uploads/OpenMP-4.0-C.pdf

omp_set_dynamic

Returns the value of the dyn-var ICV, which indicates

if dynamic adjustment of the number of threads is enabled or disabled.

void omp_set_dynamic(int dynamic_threads);

https://www.openmp.org/wp-content/uploads/OpenMP-4.0-C.pdf

omp_get_dynamic

This routine returns the value of the dyn-var ICV,

which is true if dynamic adjustment of the number of threads is enabled for the current task.

```
int omp_get_dynamic(void);
```

https://www.openmp.org/wp-content/uploads/OpenMP-4.0-C.pdf

omp_get_cancellation

Returns the value of the cancel-var ICV, which controls the behavior of cancel construct and cancellation points.

int omp_get_cancellation(void);

https://www.openmp.org/wp-content/uploads/OpenMP-4.0-C.pdf

omp_set_nested

Enables or disables nested parallelism, by setting the nest-var ICV.

void omp_set_nested(int nested);

https://www.openmp.org/wp-content/uploads/OpenMP-4.0-C.pdf

omp_get_nested

Returns the value of the nest-var ICV, which indicates if nested parallelism is enabled or disabled.

int omp_get_nested(void);

https://www.openmp.org/wp-content/uploads/OpenMP-4.0-C.pdf

omp_set_schedule

Affects the schedule that is applied when runtime is used as schedule kind.

void omp_set_schedule(omp_sched_t kind, int modifier);

kind: one of the folowing, or an implementation-defined schedule:

omp_sched_static = 1

omp_sched_dynamic = 2

omp_sched_guided = 3

omp_sched_auto = 4

https://www.openmp.org/wp-content/uploads/OpenMP-4.0-C.pdf

omp_get_schedule

Returns the value of run-sched-var ICV, which is the schedule applied when runtime schedule is used.

void omp_get_schedule(omp_sched_t *kind, int *modifier);

See kind above.

https://www.openmp.org/wp-content/uploads/OpenMP-4.0-C.pdf

omp_get_thread_limit

Returns the value of the thread-limit-var ICV, which is the maximum number of OpenMP threads available.

int omp_get_thread_limit(void);

https://www.openmp.org/wp-content/uploads/OpenMP-4.0-C.pdf

omp_set_max_active_levels

Limits the number of nested active parallel regions, by setting max-active-levels-var ICV.

void omp_set_max_active_levels(int max_levels);

https://www.openmp.org/wp-content/uploads/OpenMP-4.0-C.pdf

omp_get_max_active_levels

Returns the value of max-active-levels-varICV, which determines the maximum number of nested active parallel regions.

int omp_get_max_active_levels(void);

https://www.openmp.org/wp-content/uploads/OpenMP-4.0-C.pdf

omp_get_level

For the enclosing device region, returns the levels-vars ICV, which is the number of nested parallel regions that enclose the task containing the call.

int omp_get_level(void);

https://www.openmp.org/wp-content/uploads/OpenMP-4.0-C.pdf

omp_get_ancestor_thread_num

Returns, for a given nested level of the current thread, the thread number of the ancestor of the current thread.

int omp_get_ancestor_thread_num(int level);

https://www.openmp.org/wp-content/uploads/OpenMP-4.0-C.pdf

omp_get_team_size

Returns, for a given nested level of the current thread, the size of the thread team to which the ancestor or the current thread belongs.

```
int omp_get_team_size(int level);
```

https://www.openmp.org/wp-content/uploads/OpenMP-4.0-C.pdf

omp_get_active_level

Returns the value of the active-level-vars ICV, which determines the number of active, nested parallel regions enclosing the task that contains the call.

int omp_get_active_level(void);

https://www.openmp.org/wp-content/uploads/OpenMP-4.0-C.pdf

omp_in_final

Returns true if the routine is executed in a final task region; otherwise, it returns false.

int omp_in_final(void);

https://www.openmp.org/wp-content/uploads/OpenMP-4.0-C.pdf

omp_get_proc_bind

Returns the thread affinity policy to be used for the subsequent nested parallel regions that do not specify a proc_bind clause.

omp_proc_bind_t omp_get_proc_bind(void);

Returns one of:

omp_proc_bind_false	= 0
omp_proc_bind_true	= 1
omp_proc_bind_master	= 2
omp_proc_bind_close	= 3
omp_proc_bind_spread	= 4

https://www.openmp.org/wp-content/uploads/OpenMP-4.0-C.pdf

omp_set_default_device

Controls the default target device byassigning the value of the default-device-var ICV.

```
void omp_set_default_device(int device_num);
```

https://www.openmp.org/wp-content/uploads/OpenMP-4.0-C.pdf

omp_get_default_device

Returns the default target device.int

omp_get_default_device(void);

https://www.openmp.org/wp-content/uploads/OpenMP-4.0-C.pdf

omp_get_num_devices

Returns the number of target devices.

int omp_get_num_devices(void);

https://www.openmp.org/wp-content/uploads/OpenMP-4.0-C.pdf

omp_get_num_teams

Returns the number of teams in the current teams region, or 1 if called from outside of a teams region.

int omp_get_num_teams(void);

https://www.openmp.org/wp-content/uploads/OpenMP-4.0-C.pdf

omp_get_team_num

Returns the team number of calling thread. Theteam number is an integer between 0 and one less than the value returned by omp_get_num_teams, inclusive.

int omp_get_team_num(void);

https://www.openmp.org/wp-content/uploads/OpenMP-4.0-C.pdf

omp_is_initial_device

Returns true if the current task is executingon the host device; otherwise, it returns false.

int omp_is_initial_device(void);

https://www.openmp.org/wp-content/uploads/OpenMP-4.0-C.pdf

Initialize lock

Initialize an OpenMP lock.

```
void omp_init_lock(omp_lock_t *lock);
void omp_init_nest_lock(omp_nest_lock_t *lock);
```

https://www.openmp.org/wp-content/uploads/OpenMP-4.0-C.pdf

Destroy lock

Ensure that the OpenMP lock is uninitialized.

void omp_destroy_lock(omp_lock_t *lock);

void omp_destroy_nest_lock(omp_nest_lock_t *lock);

https://www.openmp.org/wp-content/uploads/OpenMP-4.0-C.pdf

Set lock

Sets an OpenMP lock. The calling task region is suspended

until the lock is set.

void omp_set_lock(omp_lock_t *lock);

void omp_set_nest_lock(omp_nest_lock_t *lock);

https://www.openmp.org/wp-content/uploads/OpenMP-4.0-C.pdf

Unset lock

Unsets an OpenMP lock.

```
void omp_unset_lock(omp_lock_t *lock);
void omp_unset_nest_lock(omp_nest_lock_t *lock);
```

https://www.openmp.org/wp-content/uploads/OpenMP-4.0-C.pdf

Test lock

Attempt to set an OpenMP lock but do not suspend execution of the task executing the routine.

```
int omp_test_lock(omp_lock_t *lock);
int omp_test_nest_lock(omp_nest_lock_t *lock);
```

https://www.openmp.org/wp-content/uploads/OpenMP-4.0-C.pdf

References

[1] ftp://ftp.geoinfo.tuwien.ac.at/navratil/HaskellTutorial.pdf

[2] https://www.umiacs.umd.edu/~hal/docs/daume02yaht.pdf