OpenMP Directives , Functions , and Environment Variables

October 16, 2020

1 OpenMP Directives

Directives are based on #*pragma* directives defined in the C and C++ standards. Compilers that support the OpenMP C and C++ API will include a command-line option that activates and allows interpretation of all OpenMP compiler directives. The directives are classified into parallel construct, work-sharing construct, synchronization construct and data environment construct. The different openMP directives supported are shown in the Table 1.

2 OpenMP Functions

The OpenMP functions are included in a header file called omp.h. The Visual C++ implementation of the OpenMP standard includes the functions (for environment execution and lock) and data types (for lock) shown in Tables 2 and 3.

3 Environment Variables

The Visual C++ implementation of the OpenMP standard includes the environment variables in Table 4. These environment variables are read at program startup and modifications to their values are ignored at runtime.

Name	**Parallel**	**Worksharing**	**Synchronization**	**Data environment**	**Description**
parallel	Yes	No	No	No	Defines a parallel region, which is code that will be executed by multiple threads in parallel
for	No	Yes	No	No	Causes the work done in a for loop inside a parallel region to be divided among threads
sections	No	Yes	No	No	Identifies code sections to be divided among all threads
single	No	Yes	No	No	Lets you specify that a section of code should be executed on a single thread, not necessarily the master thread
master	No	No	Yes	No	Specifies that only the master thread should execute a section of the program
critical	No	No	Yes	No	Specifies that code is only executed on one thread at a time
barrier	No	No	Yes	No	Synchronizes all threads in a team; all threads pause at the barrier, until all threads execute the barrier
atomic	No	No	Yes	No	Specifies that a memory location that will be updated atomically
flush	No	No	Yes	No	Specifies that all threads have the same view of memory for all shared objects
ordered	No	No	Yes	No	Specifies that code under a parallelized for loop should be executed like a sequential loop
threadprivate	No	No	No	Yes	Specifies that a variable is private to a thread

Table 1: OpenMP directives

Name	**Environment Execution**	**Lock**	**Description**
omp_set_num_threads	Yes	No	Sets the number of threads in upcoming parallel regions, unless overridden by a num_threads clause.
omp_get_num_threads	Yes	No	Returns the number of threads in the parallel region.
omp_get_max_threads	Yes	No	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_thread_num	Yes	No	Returns the thread number of the thread executing within its thread team.
omp_get_num_procs	Yes	No	Returns the number of processors that are available when the function is called.
omp_in_parallel	Yes	No	Returns nonzero if called from within a parallel region.
omp_set_dynamic	Yes	No	Indicates that the number of threads available in upcoming parallel regions can be adjusted by the run time.
omp_get_dynamic	Yes	No	Returns a value that indicates if the number of threads available in upcoming parallel regions can be adjusted by the run time.
omp_set_nested	Yes	No	Enables nested parallelism.
omp_get_nested	Yes	No	Returns a value that indicates if nested parallelism is enabled.
omp_init_lock	No	Yes	Initializes a simple lock.
omp_init_nest_lock	No	Yes	Initializes a lock.
omp_destroy_lock	No	Yes	Uninitializes a lock.
omp_destroy_nest_lock	No	Yes	Uninitializes a nestable lock.
omp_set_lock	No	Yes	Blocks thread execution until a lock is available.
omp_set_nest_lock	No	Yes	Blocks thread execution until a lock is available.
omp_unset_lock	No	Yes	Releases a lock.
omp_unset_nest_lock	No	Yes	Releases a nestable lock.
omp_test_lock	No	Yes	Attempts to set a lock but doesn't block thread execution.
omp_test_nest_lock	No	Yes	Attempts to set a nestable lock but doesn't block thread execution.

Table 2: OpenMP functions

Name	**Environment Execution**	**Lock**	**Description**
omp_lock_t	No	Yes	A type that holds the status of a lock, whether the lock is available or if a thread owns a lock.
omp_nest_lock_t	No	Yes	A type that holds one of the following pieces of information about a lock: whether the lock is available, and the identity of the thread that owns the lock and a nesting count.

Table 3: OpenMP datatypes

Environment Variable	**Description**
OMP_SCHEDULE	Modifies the behavior of the schedule clause when *schedule(runtime)* is specified in a *for* or *parallel for* directive.
OMP_NUM_THREADS	Sets the maximum number of threads in the parallel region, unless overridden by omp_set_num_threads or num_threads.
OMP_DYNAMIC	Specifies whether the OpenMP run time can adjust the number of threads in a parallel region.
OMP_NESTED	Specifies whether nested parallelism is enabled, unless nested parallelism is enabled or disabled with omp_set_nested.

Table 4: Environmental variables