OpenMP
openmp.org

Common Core

OpenMP API Reference Guide: Common Core

OpenMP® is an API for writing parallel applications in C/C++ and Fortran. OpenMP is suitable for programming multicore chips, NUMA systems, and devices attached to a CPU (such as a GPU).

The OpenMP Common Core consists of the 19 most commonly used items from the full OpenMP API. It is often best to master the Common Core and then proceed to the full OpenMP API as needed.

C/C++ C/C++ content | FOR Fortran content

[n.n.n] 5.0 spec. [n.n.n] 4.5 spec.

Directives and Constructs

parallel construct

parallel [2.9] [2.5]

Forms a team of threads and starts parallel execution.

C/C++	**#pragma omp parallel** *[clause[[,]clause] ...]* *structured-block*
FOR	**!$omp parallel** *[clause[[,]clause] ...]* *structured-block* **!$omp end parallel**

clause: **reduction**(*op* **:** *list*), **private**(*list*), **firstprivate**(*list*), **shared**(*list*)

Worksharing constructs

single [2.11.2] [2.7.3]

Specifies that the associated structured block is executed by only one of the threads in the team. Has implied barrier unless turned off with **nowait**.

C/C++	**#pragma omp single** *[clause[[,]clause] ...]* *structured-block*
FOR	**!$omp single** *[clause[[,]clause] ...]* *structured-block* **!$omp end single** *[end_clause[[,]end_clause] ...]*

clause: **private**(*list*), **firstprivate**(*list*), **nowait**

for [2.12.2] [2.7.1]

Specifies that the iterations of associated loops will be executed in parallel by threads in the team. Has implied barrier unless turned off with **nowait**.

C/C++	**#pragma omp for** *[clause[[,]clause] ...]* *structured-block*
FOR	**!$omp do** *[clause[[,]clause] ...]* *do-loops* *[***!$omp end do** *[***nowait***]]*

clause: **nowait**, **reduction**(*op* **:** *list*), **schedule**(*dynamic [, chunk]*), **schedule**(*static [, chunk]*), **private**(*list*), **firstprivate**(*list*)

static: Iterations are divided into chunks of size *chunk_size* and assigned to threads in the team in round-robin fashion in order of thread number.

dynamic: Each thread executes a chunk of iterations then requests another chunk until none remain.

Combined construct

Parallel Worksharing Loop [2.16.1] [2.11.1]

Specifies a **parallel** construct containing one worksharing-loop construct with one or more associated loops.

C/C++	**#pragma omp parallel for** *[clause[[,]clause] ...]* *structured-block*
FOR	**!$omp parallel do** *[clause[[,]clause] ...]* *do-loops* *[***!$omp end parallel do***]*

clause: Any accepted by the **parallel** or **for/do** directives, except the **nowait** clause.

Tasking constructs

task [2.13.1] [2.9.1]

Defines an explicit task.

C/C++	**#pragma omp task** *[clause[[,]clause] ...]* *structured-block*
FOR	**!$omp task** *[clause[[,]clause] ...]* *structured-block* **!$omp end task**

clause: **private**(*list*), **firstprivate**(*list*), **shared**(*list*)

Synchronization constructs

critical [2.20.1] [2.13.2]

Restricts execution of the associated structured block to a single thread at a time.

C/C++	**#pragma omp critical** *structured-block*
FOR	**!$omp critical** *structured-block* **!$omp end critical**

barrier [2.20.2] [2.13.3]

Specifies an explicit barrier at the point at which the construct appears.

C/C++	**#pragma omp barrier**
FOR	**!$omp barrier**

taskwait [2.20.5] [2.13.4]

Specifies a wait on the completion of child tasks of the current task.

C/C++	**#pragma omp taskwait**
FOR	**!$omp taskwait**

Memory consistency

Rules that define which values are allowed to be observed when a variable shared between one or more threads is read. A thread uses a flush to make its variables consistent with memory. A flush is implied at the following locations:

- Entry to and exit from critical
- Exit from explicit and implied barriers

Clauses

Data Sharing Clauses [2.22.4] [2.15.3]

Data-sharing attribute clauses apply only to variables whose names are visible in the construct on which the clause appears.

shared(*list*)
The items in the list are shared between threads or explicit tasks executing the construct.

private(*list*)
Creates a new variable for each item in the list that is private to each thread or explicit task. The private variable is not given an initial value.

firstprivate(*list*)
Declares list items to be private to each thread or explicit task and assigns them the value the original variable has at the time the construct is encountered.

Reduction Clause [2.22.5]

reduction(*op* : *list*)

Specifies one or more list items.

Operators for reduction (initialization values)			
+	(0)	|	(0)
*	(1)	^	(0)
-	(0)	&&	(1)
&	(~0)	||	(0)
max (Least representable number in **reduction** list item type)			
min (Largest representable number in **reduction** list item type)			

Runtime Library Routines

omp_set_num_threads [3.2.1] [3.2.1]

Sets default number of threads to request for parallel regions

C/C++	**void omp_set_num_threads(**int *num_threads***);**
FOR	**subroutine omp_set_num_threads(***num_threads***)** **integer** *num_threads*

omp_get_num_threads [3.2.2] [3.2.2]

Returns the number of threads in the current team.

C/C++	**int omp_get_num_threads(**void**);**
FOR	**integer function omp_get_num_threads()**

omp_get_thread_num [3.2.4] [3.2.4]

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

C/C++	**int omp_get_thread_num(**void**);**
FOR	**integer function omp_get_thread_num()**

omp_get_wtime [3.4.1] [3.4.1]

Returns elapsed wall clock time in seconds. Not guaranteed to be globally consistent across all the threads.

C/C++	**double omp_get_wtime(**void**);**
FOR	**double precision function omp_get_wtime()**

Environment Variable

[5.2] [4.2] OMP_NUM_THREADS *list*
Sets default number of threads to request for parallel regions

Notes

Learn More About OpenMP

OpenMPCon Developer's Conference

Held back-to-back with IWOMP, the annual OpenMPCon conference is organized by and for the OpenMP community to provide both novice and experienced developers tutorials and new insights into using OpenMP and other directive-based APIs.

openmpcon.org

IWOMP Interntional OpenMP Workshop

The annual International Workshop on OpenMP (IWOMP) is dedicated to the promotion and advancement of all aspects of parallel programming with OpenMP, covering issues, trends, recent research ideas, and results related to parallel programming with OpenMP.

iwomp.org

ISC and Supercomputing Conference Series

The annual ISC and SC conferences provide the high-performance computing community with technical programs that makes them yearly must-attend forums. OpenMP has a booth or holds sessions at one or more of these events every year.

supercomputing.org
isc-hpc.com

UK OpenMP Users Conference

The annual UK OpenMP Users Conference provides two days of talks and workshops aimed at furthering collaboration and knowledge sharing among the UK community of expert and novice high-performance computing specialists using the OpenMP API.

ukopenmpusers.co.uk