Important OpenMP Directives and Functions
Wichtige OpenMP-Direktiven und Funktionen

- **`#pragma omp parallel [clause] ...`**
  `statement_or_block`

  Create multiple threads for a parallel region.

  Important clauses:

  - `private(variable_list)`
  - `firstprivate(variable_list)`
  - `shared(variable_list)`

- **`#pragma omp for [clause] ...`**
  `for_loop`

  Distribute the iterations of following loop among threads.

  Important clauses:

  - `private(variable_list)`
  - `firstprivate(variable_list)`
  - `lastprivate(variable_list)`
  - `reduction(operator : variable_list)`
  - `ordered`
  - `schedule(kind, chunk-size)`
  - `nowait`

- **`#pragma omp sections [clause] ...`**

```
{
    #pragma omp section
        statement_or_block

    #pragma omp section
        statement_or_block

    ...
}
```

  Distribute the marked code sections among threads.

  Important clauses:

  - `private(variable_list)`
  - `firstprivate(variable_list)`
  - `lastprivate(variable_list)`
  - `nowait`

- **`#pragma omp task [clause] ...`**
  `statement_or_block`

  Create an asynchronously executed task.

  Important clauses:

  - `private(variable_list)`
  - `firstprivate(variable_list)`
  - `shared(variable_list)`

  - `depend(direction : variable_list)`
  - `if(expression)`

- **`#pragma omp barrier`**

  Barrier synchronization.

- **`#pragma omp single [clause] ...`**
  `statement_or_block`

  Only a single thread executes the following statement/block.

- **`#pragma omp master`**
  `statement_or_block`

  Only the master thread executes the following statement/block.

- **`#pragma omp critical [name]`**
  `statement_or_block`

  The following statement/block is executed under mutual exclusion.

- **`#pragma omp atomic [clause] ...`**
  `statement_or_block`

  The memory update in the following statement/block is executed atomically.

  Important clauses:

  - `read`
  - `write`
  - `update`
  - `capture`
  - `seq_cst`

- **`#pragma omp ordered`**
  `statement_or_block`

  The following statement/block will be executed in order required by the sequential loop.

- **`#pragma omp taskwait`**

  Wait for completion of all direct subtasks of the current task.

- **`#pragma omp taskgroup`**
  `block`

  Wait for completion of all subtasks creted in the block.

- **`int omp_get_num_threads()`**

  Return the current number of threads.

- **`int omp_get_thread_num()`**

  Return the thread number of the calling thread (range: 0 ... `omp_get_num_threads()`-1).