Directive's Categories

Mehti Musayev
M4115c academic group

What is synchronization directive?

- OpenMP directives exploit shared memory parallelism by defining various types of parallel regions. Parallel regions can include both iterative and non-iterative segments of program code.

It’s all about magic words #pragma omp

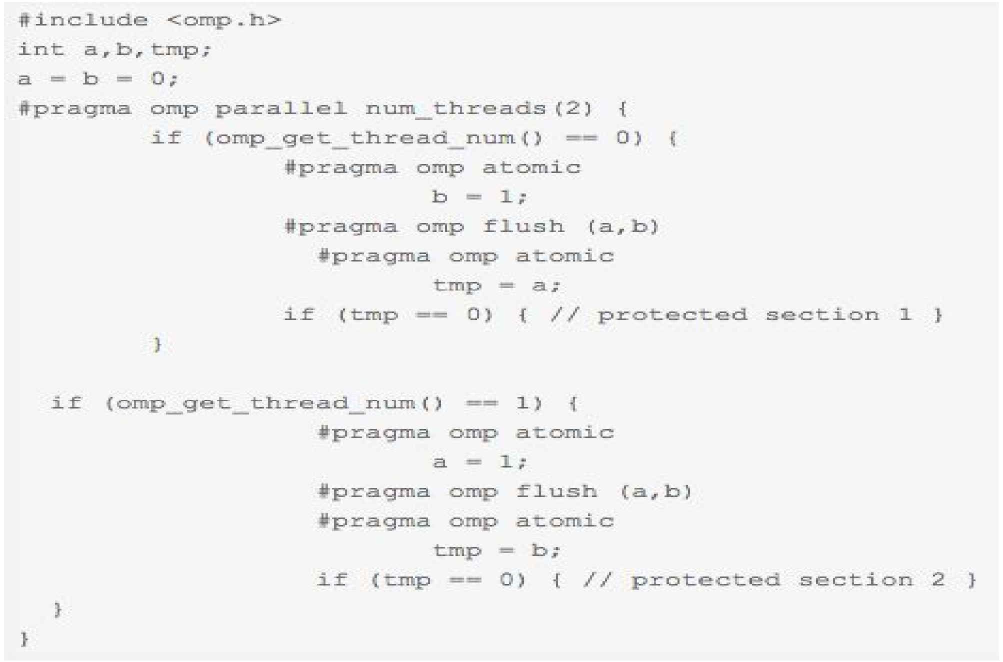
The **#pragma omp** pragmas fall into the following general categories:

- The **#pragma omp** pragmas for defining parallel regions in which work is done by threads in parallel (**#pragma omp parallel**). Most of the OpenMP directives either statically or dynamically bind to an enclosing parallel region.
- The **#pragma omp** pragmas for defining how work is distributed or shared across the threads in a parallel region (**#pragma omp sections**, **#pragma omp for**, **#pragma omp single**, **#pragma omp task**).
- The **#pragma omp** pragmas for controlling synchronization among threads (**#pragma omp atomic**, **#pragma omp master**, **#pragma omp barrier**, **#pragma omp critical**, **#pragma omp flush**, **#pragma omp ordered**) .
- The **#pragma omp** pragmas for defining the scope of data visibility across parallel regions within the same thread (**#pragma omp threadprivate**).
- The **#pragma omp** pragmas for synchronization (**#pragma omp taskwait**, **#pragma omp barrier**)

#pragma omp flush

- the flush directive is a stand-alone construct that forces a thread’s temporal local storage (view) of a variable to memory where a consistent view of the variable storage can be accesses
- the flush construct also effectively insures that no memory (load or store) operation for the variable set (list items, or default set) may be reordered across the flush directive
- Arguments: list	A comma-delimited list of one or more variables to be flushed.

#pragma omp flush example



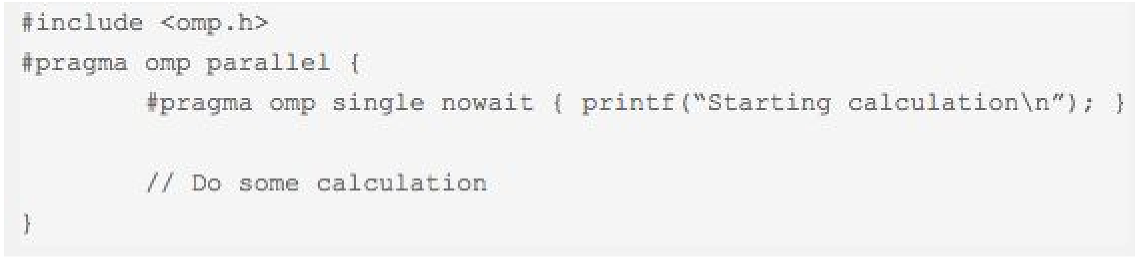
```
#include <omp.h>
int a,b,tmp;
a = b = 0;
#pragma omp parallel num_threads(2) {
        if (omp_get_thread_num() == 0) {
                #pragma omp atomic
                         b = 1;
                #pragma omp flush (a,b)
                  #pragma omp atomic
                         tmp = a;
                if (tmp == 0) { // protected section 1 }
        }

  if (omp_get_thread_num() == 1) {
                  #pragma omp atomic
                         a = 1;
                  #pragma omp flush (a,b)
                  #pragma omp atomic
                         tmp = b;
                  if (tmp == 0) { // protected section 2 }
  }
}
```

#pragma omp single

- Specifies a structured block that will be executed only once by a single thread in the team
- Arguments: clause Can be one or more of the following clauses:
    - copyprivate(list)
    - firstprivate(list)
    - nowait(integer expression)
    - private(list)

#pragma omp single example



```
#include <omp.h>
#pragma omp parallel {
        #pragma omp single nowait { printf("Starting calculation\n"); }

        // Do some calculation
}
```

#pragma omp master

- Specifies the beginning of a code block that must be executed only once by the master thread of the team
- Arguments: None

#pragma omp master example

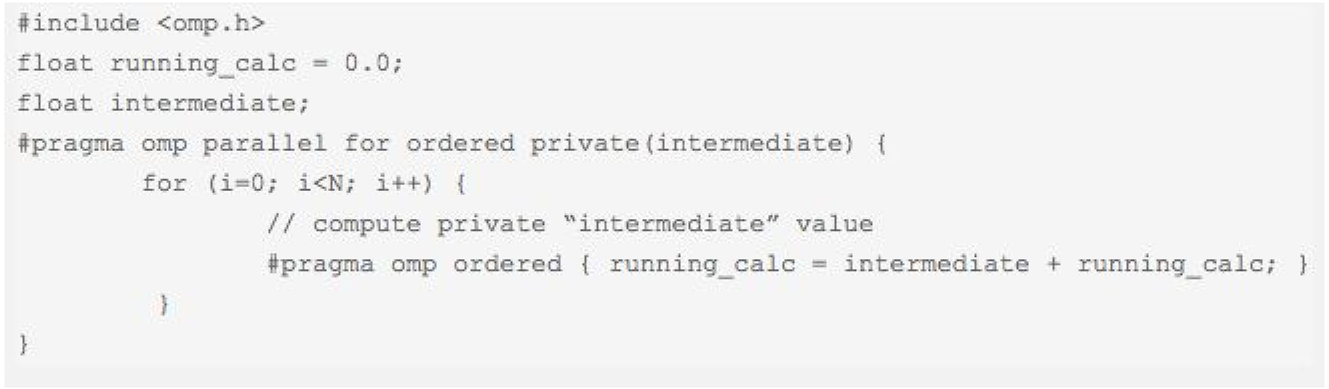
```
#include <omp.h>
double coefficient_step;

#pragma omp parallel private(coefficient_step) {
  #pragma omp master {
          coefficient_step = omp_get_wtime();
           output_timestamp(coefficient_step, "calculating coefficients");
  }
  // parallel work to calculate coefficients
}
```

#pragma omp ordered

- Specifies a code block in a worksharing loop that will be run in the order of the loop iterations
- Arguments: None

#pragma omp ordered example



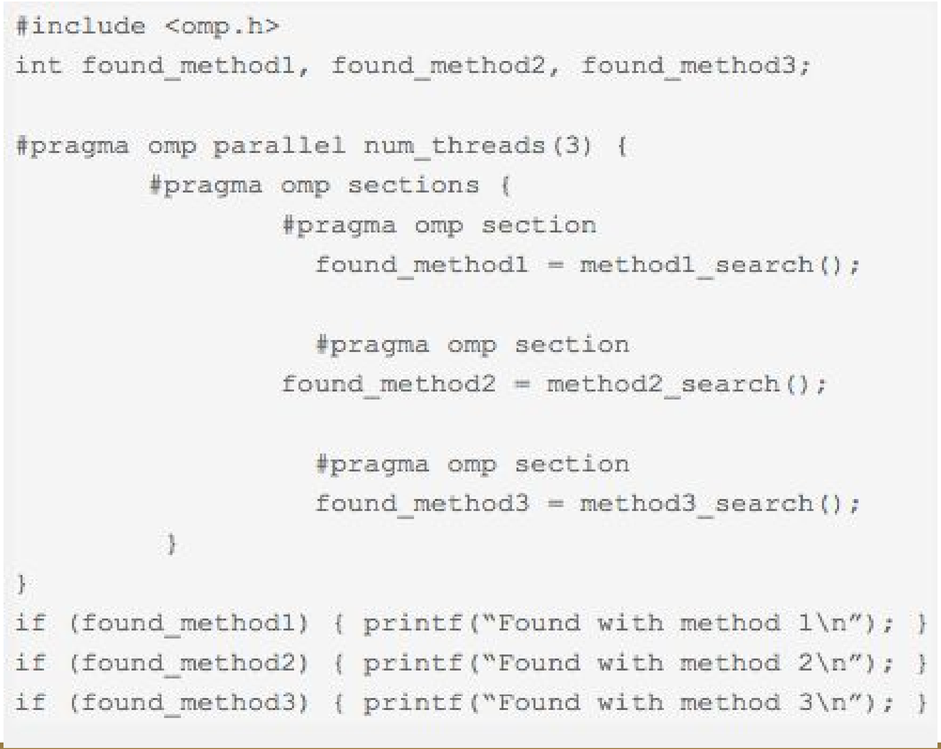
```
#include <omp.h>
float running_calc = 0.0;
float intermediate;
#pragma omp parallel for ordered private(intermediate) {
        for (i=0; i<N; i++) {
                // compute private "intermediate" value
                #pragma omp ordered { running_calc = intermediate + running_calc; }
         }
}
```

#pragma omp sections

- Defines a region of structured blocks that will be distributed among the threads in a team
- Arguments: clause

  Can be one or more of the following clauses:
  - firstprivate(list)
  - lastprivate(list)
  - nowait
  - private(list)
  - reduction(operator:list)

#pragma omp sections example



```
#include <omp.h>
int found_method1, found_method2, found_method3;

#pragma omp parallel num_threads(3) {
        #pragma omp sections {
                #pragma omp section
                  found_method1 = method1_search();

                  #pragma omp section
                found_method2 = method2_search();

                  #pragma omp section
                  found_method3 = method3_search();
         }
}
if (found_method1) { printf(“Found with method 1\n”); }
if (found_method2) { printf(“Found with method 2\n”); }
if (found_method3) { printf(“Found with method 3\n”); }
```

References

- https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.3.0/com.ibm.zos.v2r3.cbcpx01/cuppovrv2.htm
- https://msdn.microsoft.com/en-us/library/0ca2w8dk.aspx
- http://www.openmp.org/wp-content/uploads/openmp-examples-4.5.0.pdf
- https://software.intel.com/en-us/node/524494

Thank you!

- email: mehty.musaev@gmail.com
- c.p.: +7-911-039-51-35
- LinkedIn: https://www.linkedin.com/in/mehtimusayev/
- presentation: https://github.com/b5y/parallel_programing