Chapter 18 Work sharing

Working sharing construct: a way of dividing parallelizable work over a team of thread.

They are “for/do, sections, single, task, workshare”.

1. Section

```
y = 0;
#pragma omp sections reduction (+:y)
{
#pragma omp section
 y += f (x)
#pragma omp section
y += g(x)
}
```

2. single/master

The single and master pragma limits the execution of a block to a single thread.

```
int a;
#pragma omp parallel
{
  #pragma omp single
  a = f (); // some computation
  #pragma omp sections
  // various different computations using a
}
```

Chapter 19 Controlling thread data

1. shared data

Any data declared outside of a parallel region will be shared, any thread using that variable will access the same memory location associated with that variable.

2. private data

Any variable declared in a block of following an OpenMP directive will be local to the executing thread. There is no storage association between the private and global one. The Fortran language does not have the concept of scope, you have to use a private clause: !$OMP parallel private (x)

3. Data in dynamic scope

Any variables locally defined to the function are private.

4. Temporary variables in a loop

Having a variable that is set and used in each loop iteration.

5. Default

Loop variables in an “omp for” are private;

Local variables in the parallel region are private.

6. Array data

Statically allocated data can be shared or private, depending on the clause you use.

Dynamically allocated data can only be shared.

7. First and last private

There are two cases where you may want some storage association between a private variable and a global counterpart.

8. Persistent data through threadprivate

Chapter 20 Reductions

1. Built-in reduction operators

Arithmetic reductions: +, *, -, max, min

Logical operator reductions in C: & && | || ^

2. Initial value for reductions

The treatment of initial values in reductions is slightly involved.

3. User-defined reductions

#pragma omp declare reduction (identifier: typelist: conbiner) [initializer (initializer-expression)]

where

indentifier is a name, typelist is a list of types, combiner is an expression that updates the internal variable omp_out as function of itself and omp_in

initializer sets omp_priv to the identity of the reduction.

Questions

1. What does single and master pragma mean? In 18.2.
2. What is temporary variables? Are they eliminated after the parallel region?
3. How to understand the first and last private?