**Intro to Shared Memory Parallel**

**Computing with OpenMP**

OpenMP Directives Sheet for C/C++

# Parallel Region

| #include "omp.h" . . ... serial code ... . . #pragma omp parallel {  ... code to be executed ...  ... in parallel ... } . . ... serial code ... . . |
| --- |

# Parallel Region with Parallel Loop

| #include "omp.h" . . ... serial code ... . . #pragma omp parallel {  #pragma omp for  for (i = 0; i < N; i++)  {  ... task in parallel ...  } } . . ... serial code ... . . |
| --- |

## 

## 

# Parallel Region with Parallel Loop Combined

| #include "omp.h" . . ... serial code ... . . #pragma omp parallel for for(i = 0; i < N; i++) {  ... task to be executed ...  ... inside parallel loop ... } . . ... serial code ... . . |
| --- |

# Loop Scheduling Syntax

| #pragma omp for schedule(scheduleOption) |
| --- |

scheduleOption can be:

* **static**: this is the default if you don’t specify a schedule
* **dynamic**: default chunk size is 1, you can specify a different one, e.g.
  + schedule(dynamic, 100)
* **guided:** similar to dynamic, default minimum chuck size is 1, but you can specify a different one as well

# Data Contexts Syntax

| #pragma omp parallel private(var1, var2) shared(var3, var4) |
| --- |

* **private**: each thread has its own copy of these variables
* **shared**: threads share these variables

These options can also be used along with parallel loops.Other options are:

* **firstprivate**: each thread will have its own copy that is initialized with the value that the variable had before entering the parallel region.
* **lastprivate**: the value of the variable that is passed back to the main code after the parallel region is completed is the one that the thread that made the last operation in the region had.

# Reduction Clauses Syntax

| #pragma omp parallel reduction(reductionType:var1) |
| --- |

and/or

| #pragma omp for reduction(reductionType:var1) |
| --- |

or

| #pragma omp parallel for reduction(reductionType:var1) |
| --- |

The reduced variable (var1) is **always** a private variable (no need to specify that). Some options for reductionType are:

* +: sum the local values of each thread
* min: get the minimum value of all threads
* max: get the maximum value of all threads

# Measuring Execution Time

| #include "omp.h" . . double startT, stopT; . . ... serial stuff ... . . startT = omp\_get\_wtime(); #pragma omp parallel {  ... parallel stuff ... } stopT = omp\_get\_wtime(); . . ... more serial stuff ... . . |
| --- |

(stopT - startT) will give you the time, in seconds, that it took to run the parallel region.