**Introduction to OpenMP**

**What is OpenMP?**

An application Program Interface(API) that may be used to explicitly direct multi-threaded, shared memory parallelism is called OpenMP.

* Comprises three primary API components
* Compile Directives
* Runtime Library Routines
* Environment Variables
* Portable
* The API is specified for C/C++ and Fortran.
* Has been implemented for most major platforms including Unix/Linux platforms and WindowsNT.

**What is OpenMP?(Construct)**

* Standardized
* Jointly defined and endorsed by a group of major computer hardware and software vendors.
* Expected to becomes and ANSI standard later???
* What does OpenMP stand for?
* Short version: Open Multi-Processing
* Long version: Open specifications for Multi-Processing via collaborative work between interested parties from the hardware and software industry, government and academia.

**Why OpenMP?**

* More efficient
* Hides the low-level details.

**OpenMP has directives that allow the programmer to:**

* Specify the parallel region
* Specify whether the variables in the parallel section are private or shared
* Specify how/if the thread are synchronized
* Specify how to parallelize loops
* Specify how the works is divided between threads(scheduling)

**Shared memory, thread-based parallelism:**

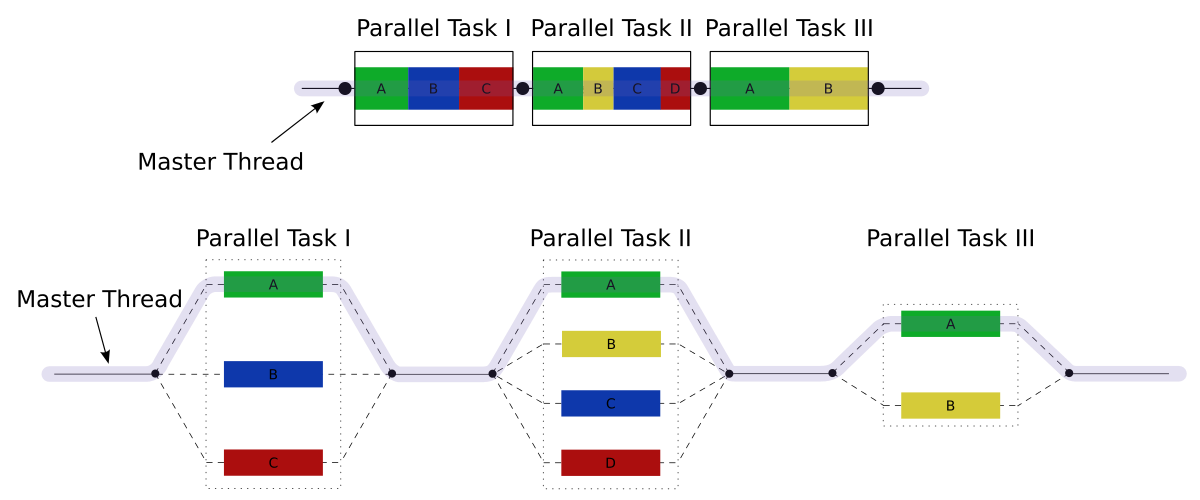
* OpenMP is based upon the existence of multiple threads in the shared memory programming paradigm.
* A shared memory process consists of multiple threads.

**Explicit Parallelism**

* OpenMP is an explicit (Not automatic) programming model, offering the programmer full control over parallelism.
* OpenMP uses the fork-join model of parallel execution.

**Compiler directive based**

* Most OpenMP parallelism is specified through the use of compiler directives which are imbedded in C/C++ or Fortran source code.

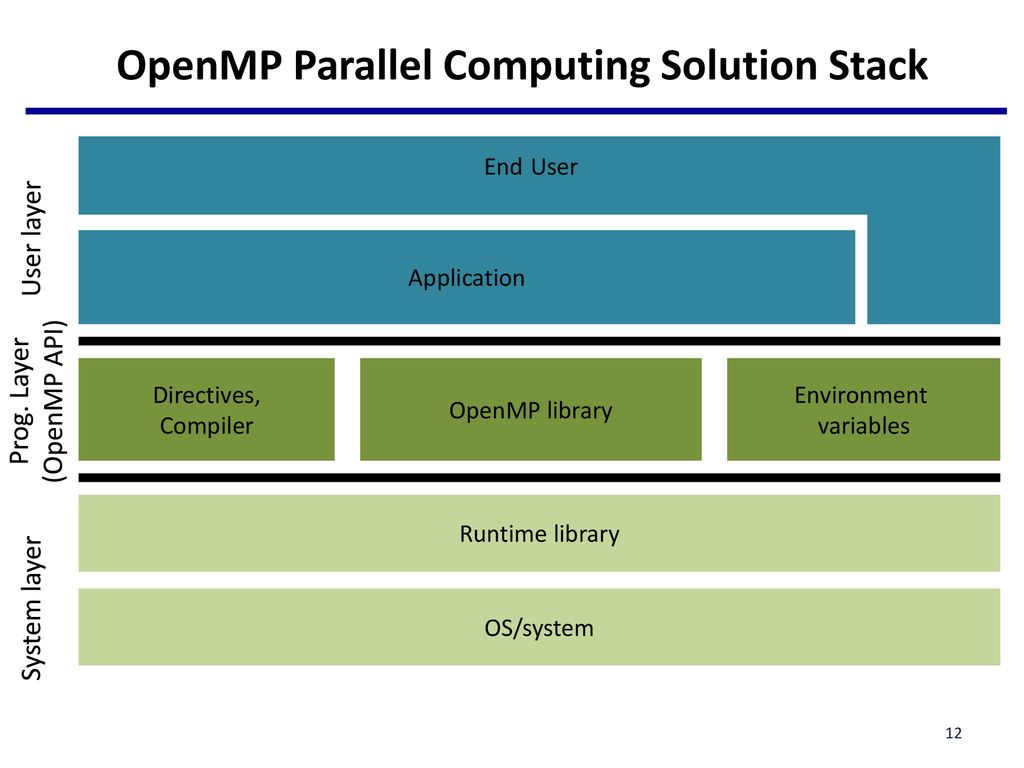


All OpenMP programs begin as a single process: the master thread. The master thread executes sequentially until the first parallel region construct is encountered.

**FORK:** The master thread then creates a team of parallel threads

* The statements in the program that are enclosed by the parallel region construct are then executed in parallel among the team threads.

**JOIN:** When the team threads complete the statements in the parallel region construct, the synchronize and terminate, leaving only the master thread.



**Basic Syntax**

* Function prototype and type in the file:

#include<omp.h>

* Most of the construct in OpenMP are compile directives.

#pragma\_omp\_construct[clause [clause]…]

{

//..Do some work here

}

**Example:**

#pragma omp parallel num\_threads(5)

**OpenMP Components**

* **Directives**
* Parallel region
* Worksharing construct
* Tasking
* Offloading
* Affinity
* Error Handling
* SIMD
* Synchronization
* Data-sharing attributes
* **Runtime Environment**
* Number of threads
* Thread ID
* Dynamic thread adjustment
* Nested parallelism
* Schedule
* Thread limit
* Wallclock timer
* Team size
* Nesting level
* **Runtime Variables**
* Number of threads
* Scheduling type
* Dynamic thread djusting
* Nested parallelism
* Thread limit