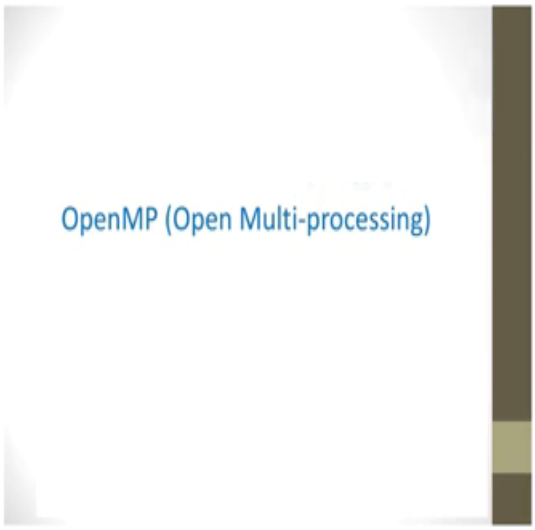
Introduction to Parallel Programming in OpenMP

Dr. Yogish Sabharwal
Department of Computer Science & Engineering
Indian Institute of Technology, Delhi

Lecture – 04
First OpenMP program and Threads

(Refer Slide Time: 00:26)



We will start with open mp now, open mp stands for open multi processing. We will go through some programs we will start with hello world, and then we will build on it to explain various features of open mp and what all you can do and different constructs.

(Refer Slide Time: 00:31)

So, let us start with a very simple program, I just want to print hello world in parallel right and parallel I mean multiple threads.

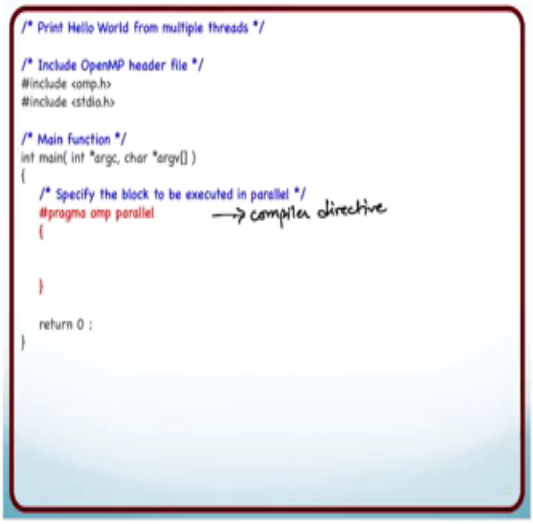
(Refer Slide Time: 00:44)

So, we will get into details of what are threatening, for now just think of a thread as something executing in parallel, right.

So, if there are four threads there are 4 codes executing in parallel. So, I want to print hello world for multiple threads right in parallel. So, how do I write an open mp program

for that? The very first thing I have to do is, I have to include the open mp header file it is called omp dot h, this contains the prototypes for various open mp functions. In open mp you will find a lot of constructs directive functions, you know containing the word omp, right is just a short form for openMP.

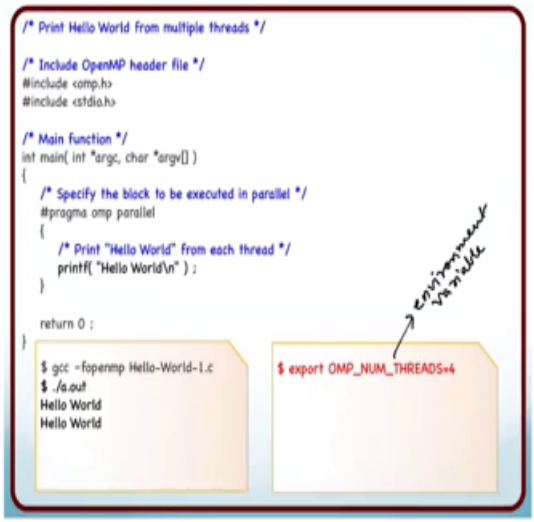
(Refer Slide Time: 01:40)



So, we include this file omp dot h, that is mandatory then I define the main function this is as usual as you define your main function for any c code. So, a as I move across slides explaining the code rate, the new code will always be in red. So, just follow the red parts of the code that will tell you what you are supposed to concentrate on what has changed from the previous slide. And now whenever I want some particular block to be executed in parallel, I have to enclose it within the construct hash pragma omp parallel. This is a compiler directive right. So, it essentially when you compile this code rate, the compiler understands this directive and knows the rate has to do something over here. So, what does it do over here it basically substitutes a lot of code.

So, typically if you have just one statement which is to be executed in parallel, you just specify that after hash pragma omp parallel, you do not need to put these curly braces and if you have a set of statements that you want to execute in parallel, which is generally the case then you put them in parentheses, right. So, that just says that this block is supposed to be executed in parallel.

(Refer Slide Time: 02:50)

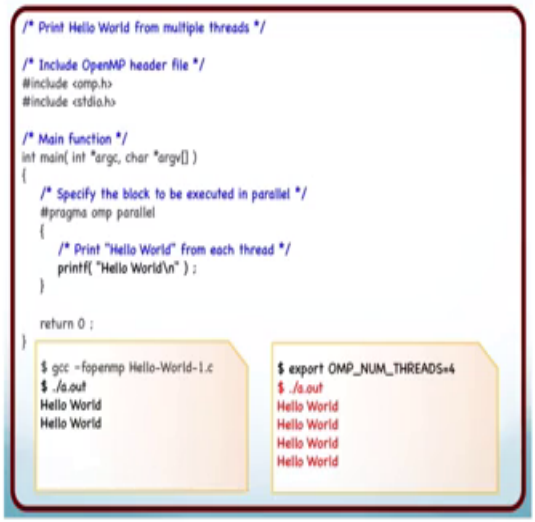


Now, what do I do? I basically print hello world from each thread right. So, this is print f hello world. So, first I have to compile this code, for the GCC compiler I just have to give a flag dash f open mp right that tells the compiler that this is an open mp code and to use the appropriate libraries and so on and after compiling I basically run the output file which the default is a dot out, and what does it print? It print hello world multiple time how many times does it print hello world this is equivalent to the number of threads that were launched right the number of threads that executed the parallel block.

So, in this case the number of threads goes to, but how do we know what is the number of threads? Here I have not specified anywhere what the number of threads is right. So, some default will be picked up. So, how do you specify the number of threads? There are multiple ways of specifying how many threads you want to execute the parallel region or this is one of them. So, you can specify it using the environment variable, omp num threads will remember the directive is something that the compiler understand. So, it substituted that code over there.

So, that code basically looks up if the environment variable exists it is going to use that information for something right and in this case it uses this particular environment variable omp num threads, looks it up if it is defined it uses that to determine how many threads should be launched, how many threads should execute the parallel region.

(Refer Slide Time: 04:29)



So, the way to do it is you say export omp number is equal to 4 right. So, this defines the environment variable set it to 4 and now if I execute the code again I again type a dot out this time I see that there are 4 prints right hello world is printed 4 times. So, now, I know how to control the number of threads that execute the parallel region right.

So, this is doing it dynamically, at the time that the code was compiled it had no information of how many threads are supposed to be launched right. So, the compiler had no way of knowing. So, it has to substitute some dynamic code over there, lookup this particular environment variable and if it exists then launch that many number of threads right and this is done dynamically at run right. So, if I just change this environment variable and without compiling the code I run a dot out again, let us say I set it to 3 then it will only be printed thrice right.

So, all of this is being determined dynamically.