Shared-Memory Programming with OpenMP

OpenMP is an API for shared-memory parallel programming.

“MP” in OpenMP stands for “multiprocessing,”

Thus, OpenMP is designed for systems in which each thread or process can potentially have access to all available memory, and, when we’re programming with OpenMP, we view our system as a collection of cores or CPUs, all of which have access to main memory

OpenMP: allows the programmer to simply state that a block of code should be executed in parallel, and the precise determination of the tasks and which thread should execute them is left to the compiler and the run-time system

OpenMP requires compiler support for some operations, and hence it’s entirely possible that you may run across a C compiler that can’t compile OpenMP programs into parallel programs

2.1 The program

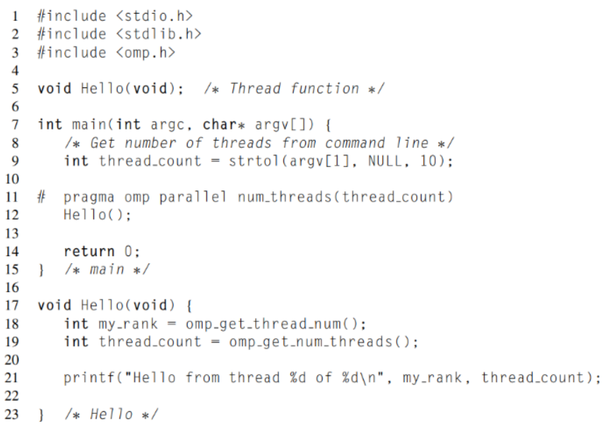
Pra gmas in C and C++ start with

#pragma

When we start the program from the command line, the operating system starts a single-threaded process, and the process executes the code in the main function

things get **interesting in Line 11.** This is our first OpenMP directive, and we’re using it to specify that the program should start some threads.

Each thread that’s forked should execute the Hello function, and when the threads return from the call to Hello, they should be terminated, and the process should then terminate when it executes the return statement



```
1   #include <stdio.h>
2   #include <stdlib.h>
3   #include <omp.h>
4
5   void Hello(void);  /* Thread function */
6
7   int main(int argc, char* argv[]) {
8      /* Get number of threads from command line */
9      int thread_count = strtol(argv[1], NULL, 10);
10
11  #  pragma omp parallel num_threads(thread_count)
12     Hello();
13
14     return 0;
15  }  /* main */
16
17  void Hello(void) {
18     int my_rank = omp_get_thread_num();
19     int thread_count = omp_get_num_threads();
20
21     printf("Hello from thread %d of %d\n", my_rank, thread_count);
22
23  }  /* Hello */
```

Compiling and running OpenMP programs

To compile this with gcc we need to include the -fopenmp option

[redacted]

to run the program, we specify the number of threads on the command line for example,

we might run the program with four threads and type

[redacted]

If we do this, the output might be

[redacted]

```
long strtol(
      const char*  number p      /* in  */,
      char**       end p         /* out */,
      int          base          /* in  */);
```

To get the number of threads

2.2 The Trapezoidal Rule

2.3 Scope of Variables

2.4 The Reduction Clause