NAME: ASHISH RANJAN RoLL:2023BCS0027

CSS311 Parallel and Distributed Computing

Lab 4

Task 1: Explore openmp parallel compiler directives and openmp environmental variables.

1. OpenMP Parallel Compiler Directives

OpenMP (Open Multi-Processing) is an API that supports multi-platform shared memory multiprocessing programming in C, C++, and Fortran. The key feature of OpenMP is the use of **compiler directives** (pragmas) to instruct the compiler how to parallelize code.

What are Compiler Directives?

Compiler directives are special instructions embedded in source code that tell the compiler how to process the code for parallel execution.

Common OpenMP Parallel Compiler Directives

1. **#pragma omp parallel**
   - Starts a parallel region where the code block is executed by multiple threads in parallel.
   - Syntax:
   - #pragma omp parallel
   - {
   -   // Code here runs in parallel
   - }
   - Example:
   - #pragma omp parallel
   - {

    - int thread_id = omp_get_thread_num();
    - printf("Hello from thread %d\n", thread_id);
    - }

2. **#pragma omp for**
    - Distributes iterations of a for loop among threads.
    - Often combined with parallel or parallel for.
    - Syntax:
    - #pragma omp parallel for
    - for (int i = 0; i < N; i++) {
    - // loop iterations run in parallel
    - }
3. **#pragma omp sections**
    - Splits the work into separate sections that can be executed by threads in parallel.
    - Syntax:
    - #pragma omp parallel sections
    - {
    - #pragma omp section
    - {
    - // code for section 1
    - }
    - #pragma omp section
    - {
    - // code for section 2
    - }
    - }

4. **#pragma omp critical**
    - Defines a critical section where only one thread can execute the block at a time to avoid race conditions.
    - Syntax:
    - #pragma omp critical
    - {
    -   // code executed by one thread at a time
    - }
5. **#pragma omp barrier**
    - Synchronizes threads; threads wait until all reach this barrier point.
    - Syntax:
    - #pragma omp barrier
6. **pragma omp single**
    - Only one thread executes the block; others skip it.
    - Syntax:
    - #pragma omp single
    - {
    -   // code executed by one thread only
    - }

---

2. OpenMP Environmental Variables

OpenMP environmental variables control runtime behavior of OpenMP programs without modifying the code. They let you control the number of threads, scheduling, nested parallelism, etc.

Common OpenMP Environmental Variables

1. **OMP_NUM_THREADS**
    - Sets the default number of threads to use in parallel regions.

- Example: export OMP_NUM_THREADS=4 (Linux/macOS)
- If not set, OpenMP runtime picks a default value based on the hardware.

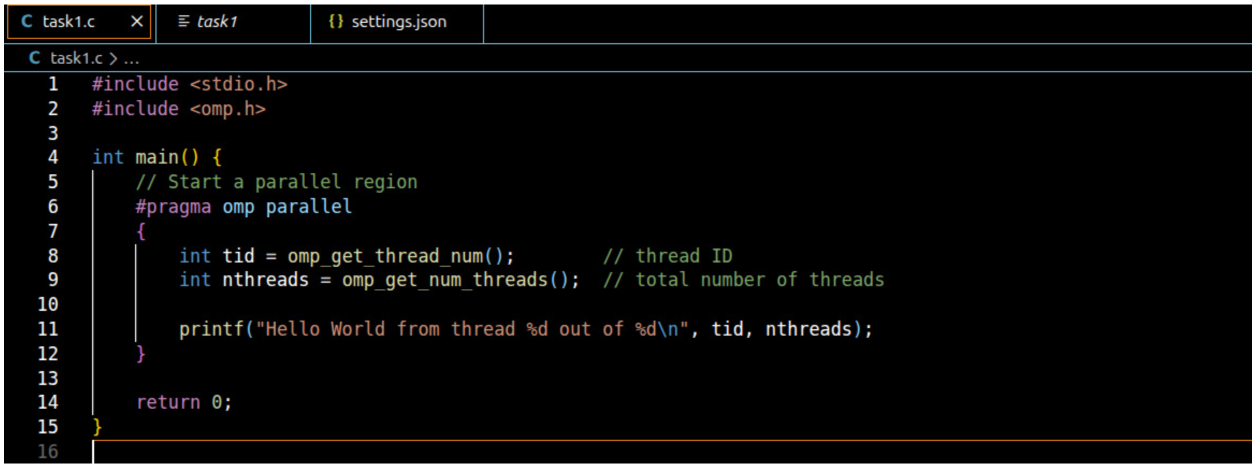
2. **OMP_SCHEDULE**
    - Controls how iterations are divided among threads in a parallel loop.
    - Syntax: OMP_SCHEDULE="type[,chunk]" where type can be:
        - static — fixed chunk size assigned in round-robin
        - dynamic — threads grab chunks dynamically
        - guided — chunks start large then get smaller
    - Example: export OMP_SCHEDULE="dynamic,4"
3. **OMP_DYNAMIC**
    - Enables/disables dynamic adjustment of the number of threads.
    - Values: TRUE or FALSE
    - Example: export OMP_DYNAMIC=TRUE
4. **OMP_NESTED**
    - Enables/disables nested parallel regions (parallel regions inside parallel regions).
    - Values: TRUE or FALSE
    - Example: export OMP_NESTED=TRUE
5. **OMP_WAIT_POLICY**
    - Controls how threads wait for work (spin or yield).
    - Values: active (spins) or passive (yields CPU).
    - Example: export OMP_WAIT_POLICY=passive
6. **OMP_PROC_BIND**
    - Controls thread affinity to processors.
    - Values: true, false, or spread, close, master
    - Example: export OMP_PROC_BIND=true

Summary Table

Directive / Variable	**Purpose**	**Example**
#pragma omp parallel	Create a parallel region	#pragma omp parallel { ... }
#pragma omp for	Parallelize loop iterations	#pragma omp parallel for for(...)
#pragma omp critical	Protect critical section	#pragma omp critical { ... }
OMP_NUM_THREADS	Set number of threads	export OMP_NUM_THREADS=8
OMP_SCHEDULE	Set loop scheduling strategy	export OMP_SCHEDULE="dynamic,2"
OMP_DYNAMIC	Enable dynamic thread adjustment	export OMP_DYNAMIC=TRUE
OMP_NESTED	Enable nested parallelism	export OMP_NESTED=TRUE

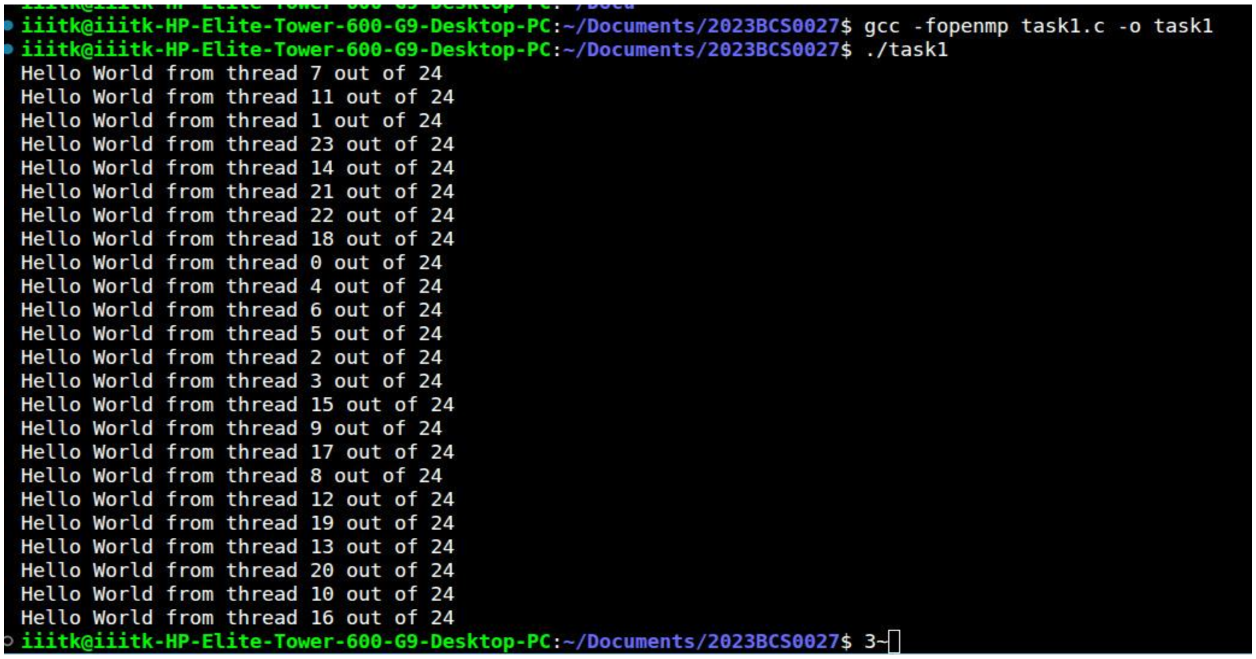
Task 2: Create 4 OpenMP threads using omp constructs and display the thread numbers using OpenMP library functions.

SIMPLE OPENMP CODE:



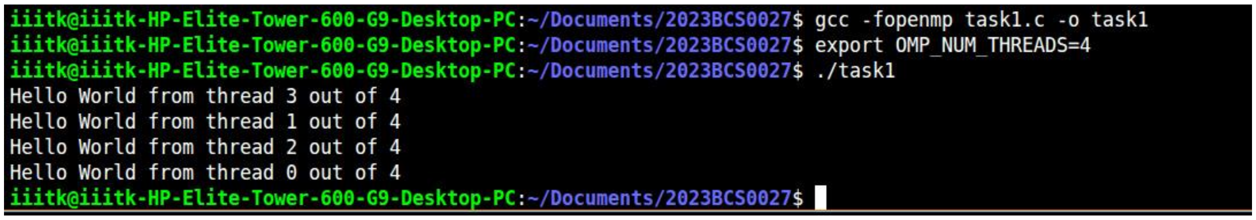
```c
#include <stdio.h>
#include <omp.h>

int main() {
    // Start a parallel region
    #pragma omp parallel
    {
        int tid = omp_get_thread_num();        // thread ID
        int nthreads = omp_get_num_threads();  // total number of threads

        printf("Hello World from thread %d out of %d\n", tid, nthreads);
    }

    return 0;
}
```

WHEN THREAD ID DEFAULT



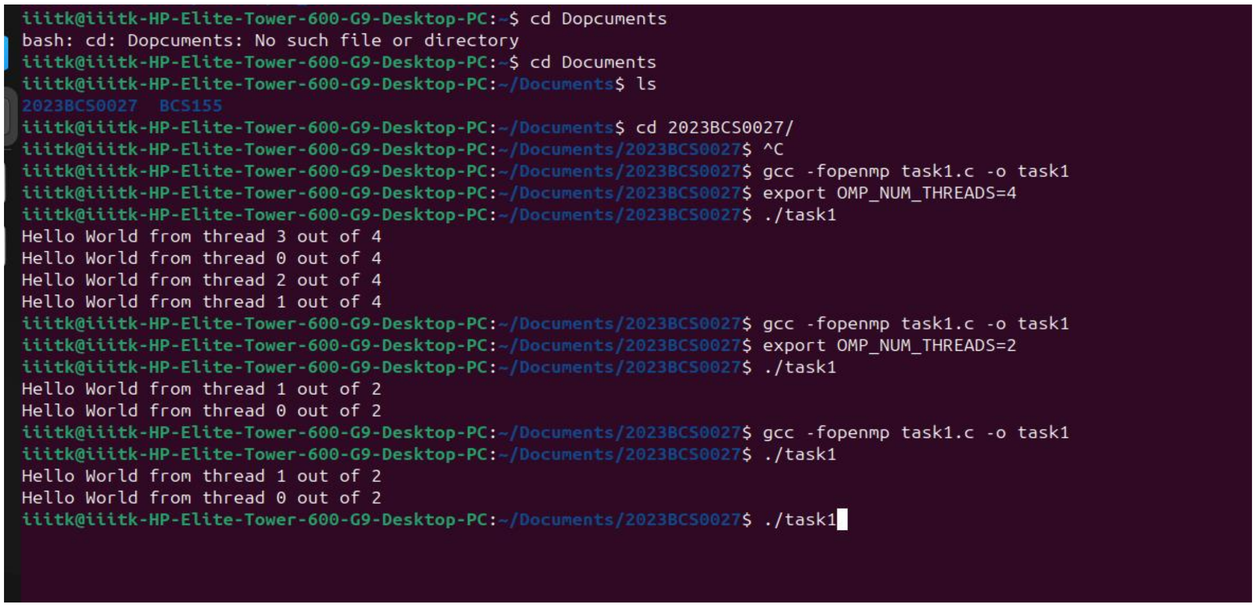
```
iiitk@iiitk-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents/2023BCS0027$ gcc -fopenmp task1.c -o task1
iiitk@iiitk-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents/2023BCS0027$ ./task1
Hello World from thread 7 out of 24
Hello World from thread 11 out of 24
Hello World from thread 1 out of 24
Hello World from thread 23 out of 24
Hello World from thread 14 out of 24
Hello World from thread 21 out of 24
Hello World from thread 22 out of 24
Hello World from thread 18 out of 24
Hello World from thread 0 out of 24
Hello World from thread 4 out of 24
Hello World from thread 6 out of 24
Hello World from thread 5 out of 24
Hello World from thread 2 out of 24
Hello World from thread 3 out of 24
Hello World from thread 15 out of 24
Hello World from thread 9 out of 24
Hello World from thread 17 out of 24
Hello World from thread 8 out of 24
Hello World from thread 12 out of 24
Hello World from thread 19 out of 24
Hello World from thread 13 out of 24
Hello World from thread 20 out of 24
Hello World from thread 10 out of 24
Hello World from thread 16 out of 24
iiitk@iiitk-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents/2023BCS0027$ 3~
```

WHEN WE USE {export OMP NUM THREADS=4}



```
iiitk@iiitk-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents/2023BCS0027$ gcc -fopenmp task1.c -o task1
iiitk@iiitk-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents/2023BCS0027$ export OMP_NUM_THREADS=4
iiitk@iiitk-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents/2023BCS0027$ ./task1
Hello World from thread 3 out of 4
Hello World from thread 1 out of 4
Hello World from thread 2 out of 4
Hello World from thread 0 out of 4
iiitk@iiitk-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents/2023BCS0027$
```

SOME VARIATION IN LINUX TERMINAL



```
iiitk@iiitk-HP-Elite-Tower-600-G9-Desktop-PC:~$ cd Dopcuments
bash: cd: Dopcuments: No such file or directory
iiitk@iiitk-HP-Elite-Tower-600-G9-Desktop-PC:~$ cd Documents
iiitk@iiitk-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ ls
2023BCS0027  BCS155
iiitk@iiitk-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents$ cd 2023BCS0027/
iiitk@iiitk-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents/2023BCS0027$ ^C
iiitk@iiitk-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents/2023BCS0027$ gcc -fopenmp task1.c -o task1
iiitk@iiitk-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents/2023BCS0027$ export OMP_NUM_THREADS=4
iiitk@iiitk-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents/2023BCS0027$ ./task1
Hello World from thread 3 out of 4
Hello World from thread 0 out of 4
Hello World from thread 2 out of 4
Hello World from thread 1 out of 4
iiitk@iiitk-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents/2023BCS0027$ gcc -fopenmp task1.c -o task1
iiitk@iiitk-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents/2023BCS0027$ export OMP_NUM_THREADS=2
iiitk@iiitk-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents/2023BCS0027$ ./task1
Hello World from thread 1 out of 2
Hello World from thread 0 out of 2
iiitk@iiitk-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents/2023BCS0027$ gcc -fopenmp task1.c -o task1
iiitk@iiitk-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents/2023BCS0027$ ./task1
Hello World from thread 1 out of 2
Hello World from thread 0 out of 2
iiitk@iiitk-HP-Elite-Tower-600-G9-Desktop-PC:~/Documents/2023BCS0027$ ./task1
```