



Department of Computer Technology

Vision of the Department

To be a well-known centre for pursuing computer education through innovative pedagogy, value-based education and industry collaboration.

Mission of the Department

To establish learning ambience for ushering in computer engineering professionals in core and multidisciplinary area by developing Problem-solving skills through emerging technologies.

Session 2025-2026

Vision: Dream of where you want.	Mission: Means to achieve Vision
---	---

Program Educational Objectives of the program (PEO): (broad statements that describe the professional and career accomplishments)

PEO1	Preparation	P: Preparation	Pep-CL abbreviation pronounce as Pep-si-IL easy to recall
PEO2	Core Competence	E: Environment (Learning Environment)	
PEO3	Breadth	P: Professionalism	
PEO4	Professionalism	C: Core Competence	
PEO5	Learning Environment	L: Breadth (Learning in diverse areas)	

Program Outcomes (PO): (statements that describe what a student should be able to do and know by the end of a program)

Keywords of POs:

Engineering knowledge, Problem analysis, Design/development of solutions, Conduct Investigations of Complex Problems, Engineering Tool Usage, The Engineer and The World, Ethics, Individual and Collaborative Team work, Communication, Project Management and Finance, Life-Long Learning

PSO Keywords: Cutting edge technologies, Research

"I am an engineer, and I know how to apply engineering knowledge to investigate, analyse and design solutions to complex problems using tools for entire world following all ethics in a collaborative way with proper management skills throughout my life." to contribute to the development of cutting-edge technologies and Research.

Integrity: I will adhere to the Laboratory Code of Conduct and ethics in its entirety.

Name and Signature of Student and Date

(Signature and Date in Handwritten)



Department of Computer Technology

Vision of the Department

To be a well-known centre for pursuing computer education through innovative pedagogy, value-based education and industry collaboration.

Mission of the Department

To establish learning ambience for ushering in computer engineering professionals in core and multidisciplinary area by developing Problem-solving skills through emerging technologies.

Session	2025-26 (ODD)	Course Name	HPC Lab
Semester	7	Course Code	22ADS706
Roll No	03	Name of Student	Debasrita Chattopadhyay

Practical Number	03
Course Outcome	1. Understand and Apply Parallel Programming Concepts 2. Analyze and Improve Program Performance. 3. Demonstrate Practical Skills in HPC Tools and Environments.
Aim	Introduction to OpenMP
Problem Definition	Implement: Part I: - OpenMP (Open Multi-Processing) Part II: - Example 1: OpenMP Scheduling Clause Example 2: Using Barrier for Synchronization Example 3: Nested Parallelism
Theory (100 words)	<p>OpenMP (Open Multi-Processing) is an API (Application Programming Interface) that supports parallel programming in C, C++, and Fortran on shared-memory architectures. It provides a set of compiler directives, runtime library routines, and environment variables that enable programmers to write applications that can run on multiple processors simultaneously.</p> <p>OpenMP follows the fork-join model of parallelism:</p> <p>When a parallel region is encountered, the master thread forks into multiple threads that run concurrently.</p> <p>At the end of the parallel region, the threads join back into a single thread (the master).</p> <p>Key Features</p> <ol style="list-style-type: none"> 1. Ease of Use – Uses simple compiler directives (#pragma omp in C/C++) to parallelize code without major modifications. 2. Shared Memory Model – All threads share the same memory address space, making communication easy.



Department of Computer Technology

Vision of the Department

To be a well-known centre for pursuing computer education through innovative pedagogy, value-based education and industry collaboration.

Mission of the Department

To establish learning ambience for ushering in computer engineering professionals in core and multidisciplinary area by developing Problem-solving skills through emerging technologies.

	<p>3. Scalability – Programs can scale with the number of available processors.</p> <p>4. Portability – Supported by most modern compilers (e.g., GCC, Intel, Clang).</p> <p>5. Incremental Parallelism – Developers can parallelize code step by step.</p>
Procedure and Execution (100 Words)	<p>Algorithm:</p> <p>Step 1: Install GCC with OpenMP support Most CentOS systems have GCC preinstalled. If not: sudo yum install gcc To verify OpenMP support: gcc -fopenmp --version</p> <p>Step 2: Write the OpenMP Program Create a file named openmp_example.c. nano openmp_example.c Paste your OpenMP C code (example below).</p> <p>Step 3: Compile the Program Use -fopenmp flag to enable OpenMP: gcc -fopenmp -o openmp_example openmp_example.c</p> <p>Step 4: Set Number of Threads (Optional) You can set how many threads OpenMP should use: export OMP_NUM_THREADS=4</p> <p>Step 5: Run the Program .openmp_example</p>
	<p>Code:</p> <p>Part 1</p> <pre>// File: openmp_example.c #include <stdio.h> #include <omp.h> int main() { int i; int n = 10; int a[10]; // Parallel region</pre>



Department of Computer Technology

Vision of the Department

To be a well-known centre for pursuing computer education through innovative pedagogy, value-based education and industry collaboration.

Mission of the Department

To establish learning ambience for ushering in computer engineering professionals in core and multidisciplinary area by developing Problem-solving skills through emerging technologies.

```
#pragma omp parallel for
for (i = 0; i < n; i++) {
    a[i] = i * i;
    printf("Thread %d: a[%d] = %d\n", i, a[i]);
}

return 0;
}
```

Part II:-

Example 1: OpenMP Scheduling Clause

```
#include <stdio.h>
#include <omp.h>
int main() {
    int i;
    int n = 12;
    #pragma omp parallel for schedule(static, 3)
    for (i = 0; i < n; i++) {
        printf("Thread %d processing iteration %d\n", i);
    }
    return 0;
}
```

Example 2: Using Barrier for Synchronization

```
#include <stdio.h>
#include <omp.h>
int main() {
    #pragma omp parallel num_threads(4)
    {
        int id = omp_get_thread_num();

        printf("Thread %d before barrier\n", id);
        #pragma omp barrier
        if (id == 0) {
            printf("All threads reached the barrier. Thread %d
continuing.\n", id);
        }
    }
    return 0;
}
```

Example 3: Nested Parallelism



Department of Computer Technology

Vision of the Department

To be a well-known centre for pursuing computer education through innovative pedagogy, value-based education and industry collaboration.

Mission of the Department

To establish learning ambience for ushering in computer engineering professionals in core and multidisciplinary area by developing Problem-solving skills through emerging technologies.

```
#include <stdio.h>
#include <omp.h>
int main() {
    omp_set_nested(1); // Enable nested parallelism
    #pragma omp parallel num_threads(2)
    {
        int id = omp_get_thread_num();
        printf("Outer thread %d starting\n", id);
        #pragma omp parallel num_threads(2)
        {
            int inner_id = omp_get_thread_num();
            printf("Inner thread %d of outer thread %d\n",
                   inner_id, id);
        }
    }
    return 0;
}
```

Output:

```
[lab1@localhost ~]$ gcc --version
gcc (GCC) 11.5.0 20240719 (Red Hat 11.5.0-5)
Copyright (C) 2021 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

```
[lab1@localhost ~]$ gcc -fopenmp --version
gcc (GCC) 11.5.0 20240719 (Red Hat 11.5.0-5)
Copyright (C) 2021 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

[lab1@localhost ~]$ nano openmp_example.c
[lab1@localhost ~]$ gcc -fopenmp -o openmp_example openmp_example.c
```

**Department of Computer Technology****Vision of the Department**

To be a well-known centre for pursuing computer education through innovative pedagogy, value-based education and industry collaboration.

Mission of the Department

To establish learning ambience for ushering in computer engineering professionals in core and multidisciplinary area by developing Problem-solving skills through emerging technologies.

The figure consists of three vertically stacked screenshots of a Linux terminal window. The terminal is titled 'Activities Terminal' and shows the command-line interface of a Red Hat system (version 11.5.0-5). The first screenshot shows the output of the 'gcc -fopenmp --version' command. The second screenshot shows the source code of 'openmp_example.c' being edited in the nano text editor. The third screenshot shows the terminal after compilation with 'gcc -fopenmp -o openmp_example openmp_example.c' and execution with './openmp_example'. The terminal output indicates that the program has successfully run, printing the result of a parallel for loop to the screen.

```
[lab1@localhost ~]$ gcc -fopenmp --version
gcc (GCC) 11.5.0 20240719 (Red Hat 11.5.0-5)
Copyright (C) 2021 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

[lab1@localhost ~]$ nano openmp_example.c
[lab1@localhost ~]$ gcc -fopenmp -o openmp_example openmp_example.c
[lab1@localhost ~]$ export OMP_NUM_THREADS=4
```



```
GNU nano 5.6.1
// File: openmp_example.c
#include <stdio.h>
#include <omp.h>

int main() {
    int i;
    int n = 10;
    int a[10];

    // Parallel region with for loop
#pragma omp parallel for
    for (i = 0; i < n; i++) {
        a[i] = i * i;
        printf("Thread %d: a[%d] = %d\n", omp_get_thread_num(), i, a[i]);
    }

    return 0;
}
```



```
[lab1@localhost ~]$ gcc -fopenmp --version
gcc (GCC) 11.5.0 20240719 (Red Hat 11.5.0-5)
Copyright (C) 2021 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

[lab1@localhost ~]$ ./openmp_example
```

**Department of Computer Technology****Vision of the Department**

To be a well-known centre for pursuing computer education through innovative pedagogy, value-based education and industry collaboration.

Mission of the Department

To establish learning ambience for ushering in computer engineering professionals in core and multidisciplinary area by developing Problem-solving skills through emerging technologies.

```

Activities Terminal lab1@localhost:~ Aug
[lab1@localhost ~]$ gcc -fopenmp --version
gcc (GCC) 11.5.0 20240719 (Red Hat 11.5.0-5)
Copyright (C) 2021 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

[lab1@localhost ~]$ nano openmp_example.c
[lab1@localhost ~]$ gcc -fopenmp -o openmp_example openmp_example.c
[lab1@localhost ~]$ export OMP_NUM_THREADS=4
[lab1@localhost ~]$ ./openmp_example

Activities Terminal lab1@localhost:~ Aug
[lab1@localhost ~]$ gcc -fopenmp --version
gcc (GCC) 11.5.0 20240719 (Red Hat 11.5.0-5)
Copyright (C) 2021 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

[lab1@localhost ~]$ nano openmp_example.c
[lab1@localhost ~]$ gcc -fopenmp -o openmp_example openmp_example.c
[lab1@localhost ~]$ export OMP_NUM_THREADS=4
[lab1@localhost ~]$ ./openmp_example
Thread 2: a[6] = 36
Thread 2: a[7] = 49
Thread 3: a[8] = 64
Thread 3: a[9] = 81
Thread 0: a[0] = 0
Thread 0: a[1] = 1
Thread 0: a[2] = 4
Thread 1: a[3] = 9
Thread 1: a[4] = 16
Thread 1: a[5] = 25
[lab1@localhost ~$]

Activities Terminal lab1@localhost:~ Aug
[lab1@localhost ~]$ gcc -fopenmp --version
gcc (GCC) 11.5.0 20240719 (Red Hat 11.5.0-5)
Copyright (C) 2021 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

[lab1@localhost ~]$ nano openmp_example.c
[lab1@localhost ~]$ gcc -fopenmp -o openmp_example openmp_example.c
[lab1@localhost ~]$ export OMP_NUM_THREADS=4
[lab1@localhost ~]$ ./openmp_example
Thread 2: a[6] = 36
Thread 2: a[7] = 49
Thread 3: a[8] = 64
Thread 3: a[9] = 81
Thread 0: a[0] = 0
Thread 0: a[1] = 1
Thread 0: a[2] = 4
Thread 1: a[3] = 9
Thread 1: a[4] = 16
Thread 1: a[5] = 25
[lab1@localhost ~]$ nano example1_schedule.c

```

**Department of Computer Technology****Vision of the Department**

To be a well-known centre for pursuing computer education through innovative pedagogy, value-based education and industry collaboration.

Mission of the Department

To establish learning ambience for ushering in computer engineering professionals in core and multidisciplinary area by developing Problem-solving skills through emerging technologies.

```
GNU nano 5.6.1
#include <stdio.h>
#include <omp.h>
int main() {
    int i;
    int n = 12;
    #pragma omp parallel for schedule(static, 3)
    for (i = 0; i < n; i++) {
        printf("Thread %d processing iteration %d\n", omp_get_thread_num(), i);
    }
    return 0;
}
```

```
[lab1@localhost ~]$ gcc -fopenmp --version
gcc (GCC) 11.5.0 20240719 (Red Hat 11.5.0-5)
Copyright (C) 2021 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

[lab1@localhost ~]$ nano openmp_example.c
[lab1@localhost ~]$ gcc -fopenmp -o openmp_example openmp_example.c
[lab1@localhost ~]$ export OMP_NUM_THREADS=4
[lab1@localhost ~]$ ./openmp_example
Thread 2: a[6] = 36
Thread 2: a[7] = 49
Thread 3: a[8] = 64
Thread 3: a[9] = 81
Thread 0: a[0] = 0
Thread 0: a[1] = 1
Thread 0: a[2] = 4
Thread 1: a[3] = 9
Thread 1: a[4] = 16
Thread 1: a[5] = 25
[lab1@localhost ~]$ nano example1_schedule.c
[lab1@localhost ~]$ gcc -fopenmp -o example1_schedule example1_schedule.c
```



Hingna Road, Wanadongri, Nagpur - 441 110

NAAC A++

Ph.: 07104-237919, 234623, 329249, 329250 Fax: 07104-232376, Website: www.ycce.edu

Department of Computer Technology

Vision of the Department

To be a well-known centre for pursuing computer education through innovative pedagogy, value-based education and industry collaboration.

Mission of the Department

To establish learning ambience for ushering in computer engineering professionals in core and multidisciplinary area by developing Problem-solving skills through emerging technologies.

The image shows three terminal windows side-by-side, each displaying a different OpenMP example. The top window shows the execution of 'openmp_example.c' which prints array values from four threads. The middle window shows the execution of 'example1_schedule.c' which prints processing iterations for four threads. The bottom window shows the code for 'example2_barrier.c' in a nano editor, which includes pragmas for parallel execution and barriers.

```
gcc (GCC) 11.5.0 20240719 (Red Hat 11.5.0-5)
Copyright (C) 2021 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

[lab1@localhost ~]$ nano openmp_example.c
[lab1@localhost ~]$ gcc -fopenmp -o openmp_example openmp_example.c
[lab1@localhost ~]$ export OMP_NUM_THREADS=4
[lab1@localhost ~]$ ./openmp_example
Thread 2: a[6] = 36
Thread 2: a[7] = 49
Thread 3: a[8] = 64
Thread 3: a[9] = 81
Thread 0: a[0] = 0
Thread 0: a[1] = 1
Thread 0: a[2] = 4
Thread 1: a[3] = 9
Thread 1: a[4] = 16
Thread 1: a[5] = 25
[lab1@localhost ~]$ nano example1_schedule.c
[lab1@localhost ~]$ gcc -fopenmp -o example1_schedule example1_schedule.c
[lab1@localhost ~]$ export OMP_NUM_THREADS=4
./example1_schedule
Thread 0: a[0] = 0
Thread 0: a[1] = 1
Thread 0: a[2] = 4
Thread 1: a[3] = 9
Thread 1: a[4] = 16
Thread 1: a[5] = 25
[lab1@localhost ~]$ nano example1_schedule.c
[lab1@localhost ~]$ gcc -fopenmp -o example1_schedule example1_schedule.c
[lab1@localhost ~]$ export OMP_NUM_THREADS=4
./example1_schedule
Thread 0 processing iteration 0
Thread 0 processing iteration 1
Thread 0 processing iteration 2
Thread 3 processing iteration 9
Thread 3 processing iteration 10
Thread 3 processing iteration 11
Thread 2 processing iteration 6
Thread 2 processing iteration 7
Thread 2 processing iteration 8
Thread 1 processing iteration 3
Thread 1 processing iteration 4
Thread 1 processing iteration 5
[lab1@localhost ~]$ nano example2_barrier.c
[lab1@localhost ~]$ nano example2_barrier.c
GNU nano 5.6.1
#include <stdio.h>
#include <omp.h>
int main() {
    #pragma omp parallel num_threads(4)
    {
        int id = omp_get_thread_num();
        printf("Thread %d before barrier\n", id);
        #pragma omp barrier
        if (id == 0) {
            printf("All threads reached the barrier. Thread %d continuing.\n", id);
        }
    }
    return 0;
}
```



Department of Computer Technology

Vision of the Department

To be a well-known centre for pursuing computer education through innovative pedagogy, value-based education and industry collaboration.

Mission of the Department

To establish learning ambience for ushering in computer engineering professionals in core and multidisciplinary area by developing Problem-solving skills through emerging technologies.

The image shows three vertically stacked terminal windows from a Linux desktop environment. Each window has a title bar with 'Activities' and 'Terminal' icons, and a status bar indicating the host as 'lab1@localhost:~'. The first terminal window displays the output of a parallel algorithm using 4 threads. The second window shows the same algorithm with 5 threads. The third window shows the algorithm with 6 threads. All three windows show the same sequence of thread processing iterations, with threads 0-3 handling odd iterations and threads 4-5 handling even iterations. The code used is a variation of the OpenMP example1_schedule.c provided in the question, with the number of threads being the key difference.

```
Thread 3: a[9] = 81
Thread 0: a[0] = 0
Thread 0: a[1] = 1
Thread 0: a[2] = 4
Thread 1: a[3] = 9
Thread 1: a[4] = 16
Thread 1: a[5] = 25
[lab1@localhost ~]$ nano example1_schedule.c
[lab1@localhost ~]$ gcc -fopenmp -o example1_schedule example1_schedule.c
[lab1@localhost ~]$ export OMP_NUM_THREADS=4
./example1_schedule
Thread 0 processing iteration 0
Thread 0 processing iteration 1
Thread 0 processing iteration 2
Thread 3 processing iteration 9
Thread 3 processing iteration 10
Thread 3 processing iteration 11
Thread 2 processing iteration 6
Thread 2 processing iteration 7
Thread 2 processing iteration 8
Thread 1 processing iteration 3
Thread 1 processing iteration 4
Thread 1 processing iteration 5
[lab1@localhost ~]$
```



```
Thread 0: a[1] = 1
Thread 0: a[2] = 4
Thread 1: a[3] = 9
Thread 1: a[4] = 16
Thread 1: a[5] = 25
[lab1@localhost ~]$ nano example1_schedule.c
[lab1@localhost ~]$ gcc -fopenmp -o example1_schedule example1_schedule.c
[lab1@localhost ~]$ export OMP_NUM_THREADS=4
./example1_schedule
Thread 0 processing iteration 0
Thread 0 processing iteration 1
Thread 0 processing iteration 2
Thread 3 processing iteration 9
Thread 3 processing iteration 10
Thread 3 processing iteration 11
Thread 2 processing iteration 6
Thread 2 processing iteration 7
Thread 2 processing iteration 8
Thread 1 processing iteration 3
Thread 1 processing iteration 4
Thread 1 processing iteration 5
[lab1@localhost ~]$ nano example2_barrier.c
[lab1@localhost ~]$ gcc -fopenmp -o example2_barrier example2_barrier.c
[lab1@localhost ~]$ ./example2_barrier
```



```
Thread 0: a[2] = 4
Thread 1: a[3] = 9
Thread 1: a[4] = 16
Thread 1: a[5] = 25
[lab1@localhost ~]$ nano example1_schedule.c
[lab1@localhost ~]$ gcc -fopenmp -o example1_schedule example1_schedule.c
[lab1@localhost ~]$ export OMP_NUM_THREADS=4
./example1_schedule
Thread 0 processing iteration 0
Thread 0 processing iteration 1
Thread 0 processing iteration 2
Thread 3 processing iteration 9
Thread 3 processing iteration 10
Thread 3 processing iteration 11
Thread 2 processing iteration 6
Thread 2 processing iteration 7
Thread 2 processing iteration 8
Thread 1 processing iteration 3
Thread 1 processing iteration 4
Thread 1 processing iteration 5
[lab1@localhost ~]$ nano example2_barrier.c
[lab1@localhost ~]$ gcc -fopenmp -o example2_barrier example2_barrier.c
[lab1@localhost ~]$ ./example2_barrier
```



Hingna Road, Wanadongri, Nagpur - 441 110

NAAC A++

Ph.: 07104-237919, 234623, 329249, 329250 Fax: 07104-232376, Website: www.ycce.edu

Department of Computer Technology

Vision of the Department

To be a well-known centre for pursuing computer education through innovative pedagogy, value-based education and industry collaboration.

Mission of the Department

To establish learning ambience for ushering in computer engineering professionals in core and multidisciplinary area by developing Problem-solving skills through emerging technologies.

The screenshot displays three terminal windows side-by-side, each showing the output of a different C program demonstrating OpenMP parallelization:

- Terminal 1:** Shows the execution of `example1_schedule.c`. It prints a sequence of iterations from 0 to 11, with threads 0, 1, 2, and 3 processing them sequentially.
- Terminal 2:** Shows the execution of `example2_barrier.c`. It prints iterations from 0 to 11, with threads 0, 1, 2, and 3 processing them sequentially, separated by barrier points.
- Terminal 3:** Shows the execution of `example3_nested.c`. It prints iterations from 0 to 11, with threads 0, 1, 2, and 3 processing them sequentially, demonstrating nested parallelism.



Nagar Yuwak Shikshan Sanstha's
Yeshwantrao Chavan College of Engineering
(An Autonomous Institution affiliated to Rashtrasant Tukadoji Maharaj Nagpur University)



Hingna Road, Wanadongri, Nagpur - 441 110

NAAC A++

Ph.: 07104-237919, 234623, 329249, 329250 Fax: 07104-232376, Website: www.ycce.edu

Department of Computer Technology

Vision of the Department

To be a well-known centre for pursuing computer education through innovative pedagogy, value-based education and industry collaboration.

Mission of the Department

To establish learning ambience for ushering in computer engineering professionals in core and multidisciplinary area by developing Problem-solving skills through emerging technologies.

The image displays three vertically stacked terminal windows, each showing the output of different OpenMP programs. The top window shows threads 0-3 processing iterations 0-10 sequentially. The middle window shows threads 0-3 processing iterations 0-11 sequentially, with a barrier point indicated. The bottom window shows threads 0-3 processing iterations 0-11 sequentially, with a nested thread structure where threads 0 and 1 have inner threads 0 and 1 respectively.

```
./example1_schedule
Thread 0 processing iteration 0
Thread 0 processing iteration 1
Thread 0 processing iteration 2
Thread 0 processing iteration 3
Thread 0 processing iteration 4
Thread 0 processing iteration 5
Thread 0 processing iteration 6
Thread 0 processing iteration 7
Thread 0 processing iteration 8
Thread 0 processing iteration 9
Thread 0 processing iteration 10
[lab1@localhost ~]$ nano example2_barrier.c
[lab1@localhost ~]$ gcc -fopenmp -o example2_barrier example2_barrier.c
[lab1@localhost ~]$ ./example2_barrier
Thread 3 before barrier
Thread 0 before barrier
Thread 2 before barrier
Thread 1 before barrier
All threads reached the barrier. Thread 0 continuing.
[lab1@localhost ~]$ nano example3_nested.c
[lab1@localhost ~]$ gcc -fopenmp -o example3_nested example3_nested.c
[lab1@localhost ~]$ ./example3_nested
```



```
Thread 0 processing iteration 0
Thread 0 processing iteration 1
Thread 0 processing iteration 2
Thread 3 processing iteration 9
Thread 3 processing iteration 10
Thread 3 processing iteration 11
Thread 2 processing iteration 6
Thread 2 processing iteration 7
Thread 2 processing iteration 8
Thread 1 processing iteration 3
Thread 1 processing iteration 4
Thread 1 processing iteration 5
[lab1@localhost ~]$ nano example2_barrier.c
[lab1@localhost ~]$ gcc -fopenmp -o example2_barrier example2_barrier.c
[lab1@localhost ~]$ ./example2_barrier
Thread 3 before barrier
Thread 0 before barrier
Thread 2 before barrier
Thread 1 before barrier
All threads reached the barrier. Thread 0 continuing.
[lab1@localhost ~]$ nano example3_nested.c
[lab1@localhost ~]$ gcc -fopenmp -o example3_nested example3_nested.c
[lab1@localhost ~]$ ./example3_nested
```



```
Thread 2 processing iteration 6
Thread 2 processing iteration 7
Thread 2 processing iteration 8
Thread 1 processing iteration 3
Thread 1 processing iteration 4
Thread 1 processing iteration 5
[lab1@localhost ~]$ nano example2_barrier.c
[lab1@localhost ~]$ gcc -fopenmp -o example2_barrier example2_barrier.c
[lab1@localhost ~]$ ./example2_barrier
Thread 3 before barrier
Thread 0 before barrier
Thread 2 before barrier
Thread 1 before barrier
All threads reached the barrier. Thread 0 continuing.
[lab1@localhost ~]$ nano example3_nested.c
[lab1@localhost ~]$ gcc -fopenmp -o example3_nested example3_nested.c
[lab1@localhost ~]$ ./example3_nested
Outer thread 0 starting
Outer thread 1 starting
  Inner thread 0 of outer thread 0
  Inner thread 1 of outer thread 0
  Inner thread 0 of outer thread 1
  Inner thread 1 of outer thread 1
[lab1@localhost ~]$
```



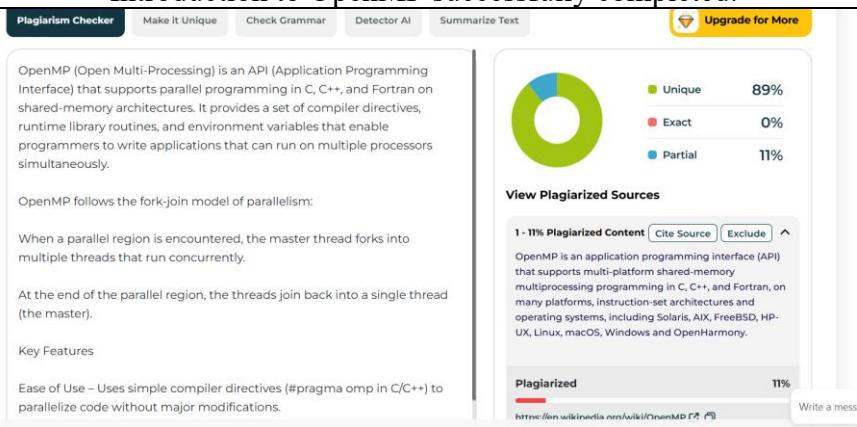
Department of Computer Technology

Vision of the Department

To be a well-known centre for pursuing computer education through innovative pedagogy, value-based education and industry collaboration.

Mission of the Department

To establish learning ambience for ushering in computer engineering professionals in core and multidisciplinary area by developing Problem-solving skills through emerging technologies.

Output Analysis	The output demonstrates how OpenMP forks multiple threads to run the same block of code concurrently. The execution order may change, but the number of threads remains consistent with the environment variable or default system settings.
Link of student Github profile where lab assignment has been uploaded	
Conclusion	Introduction to OpenMP successfully completed.
Plag Report (Similarity index < 12%)	 A screenshot of a plagiarism checker interface. At the top, there are tabs for 'Plagiarism Checker', 'Make it Unique', 'Check Grammar', 'Detector AI', and 'Summarize Text'. On the right, there is a yellow button labeled 'Upgrade for More'. Below these are three circular progress bars: 'Unique' at 89%, 'Exact' at 0%, and 'Partial' at 11%. A link 'View Plagiarized Sources' is provided. The main content area contains text about OpenMP, its features like the fork-join model, and key features. It also includes a 'Plagiarized' section with a red bar showing 1-11% plagiarized content, a link to a source, and a 'Exclude' button. <p>OpenMP (Open Multi-Processing) is an API (Application Programming Interface) that supports parallel programming in C, C++, and Fortran on shared-memory architectures. It provides a set of compiler directives, runtime library routines, and environment variables that enable programmers to write applications that can run on multiple processors simultaneously.</p> <p>OpenMP follows the fork-join model of parallelism:</p> <p>When a parallel region is encountered, the master thread forks into multiple threads that run concurrently.</p> <p>At the end of the parallel region, the threads join back into a single thread (the master).</p> <p>Key Features</p> <p>Ease of Use – Uses simple compiler directives (#pragma omp in C/C++) to parallelize code without major modifications.</p> <p>1 - 11% Plagiarized Content Cite Source Exclude</p> <p>OpenMP is an application programming interface (API) that supports multi-platform shared-memory multiprocessing programming in C, C++, and Fortran, on many platforms, instruction-set architectures and operating systems, including Solaris, AIX, FreeBSD, HP-UX, Linux, macOS, Windows and OpenHarmony.</p> <p>Plagiarized 11% Write a message</p> <p>https://en.wikipedia.org/wiki/OpenMP</p>
Date	26/ 08 /25