Name: Nitesh Sharma
Reg No: 23361
MTCS-103(P)
Parallel Processing Practicals
Assignment 12

==========

Environment Variables:

Environment variables in OpenMP are configuration settings that influence the behavior of OpenMP runtime and library functions. These variables have uppercase names and are case insensitive, allowing for flexibility in specifying their values. Leading and trailing white spaces in these values are also accommodated. By using environment variables, developers can adjust runtime behavior without modifying code, making it easier to control aspects like thread behavior, parallelism levels, and debugging options during program execution.

OMP_SCHEDULE type[,chunk]:

The OMP_SCHEDULE environment variable plays a crucial role in controlling the behavior of OpenMP parallel loops with regards to their scheduling. This variable is used to set the run-sched-var internal control variable (ICV) for determining the runtime schedule type and the optional chunk size for parallel loop iterations. The schedule type can be specified as static, dynamic, guided, or auto, each affecting how loop iterations are distributed among threads. Additionally, a positive integer chunk value can be provided to regulate the size of chunks that are distributed to threads. By manipulating these settings through the OMP_SCHEDULE environment variable, developers can optimize loop execution and achieve better workload distribution among threads, thereby improving parallelism and performance in OpenMP programs.

```
#include <stdio.h>
#include <omp.h>
int main() {
  int array_size = 10;
  int array[array_size];

  for (int i = 0; i < array_size; ++i) {
     array[i] = i + 1;
  }
  omp_set_schedule(omp_sched_dynamic, 2);
  int sum = 0;
  #pragma omp parallel for reduction(+:sum)
  for (int i = 0; i < array_size; ++i) {
     sum += array[i];
```

```
    printf("Thread %d: Array[%d] = %d\n", omp_get_thread_num(), i, array[i]);
  }
  printf("Total Sum: %d\n", sum);

  return 0;
}
```

Output:

```
et2023@dmacs13-OptiPlex-9020:~/OPENMP/ r( $ gcc -fopenmp omp_schedule.c
et2023@dmacs13-OptiPlex-9020:~/OPENMP/ r( $ time ./a.out
Thread 2: Array[6] = 7
Thread 2: Array[7] = 8
Thread 3: Array[8] = 9
Thread 3: Array[9] = 10
Thread 1: Array[3] = 4
Thread 1: Array[4] = 5
Thread 1: Array[5] = 6
Thread 0: Array[0] = 1
Thread 0: Array[1] = 2
Thread 0: Array[2] = 3
Total Sum: 55
```

Explanation:

In the provided code, the `omp_set_schedule` function configures the scheduling behavior for the subsequent parallel loop construct. Specifically, the code sets the scheduling policy to `omp_sched_dynamic` with a modifier of 2. This means that the loop iterations will be dynamically allocated to threads, and each thread will process chunks of two consecutive loop iterations. As a result, the workload is distributed dynamically among the threads, potentially achieving better load balancing and reduced overhead compared to finer-grained scheduling. The use of this scheduling policy aims to optimize thread utilization and efficiency in the parallel loop.

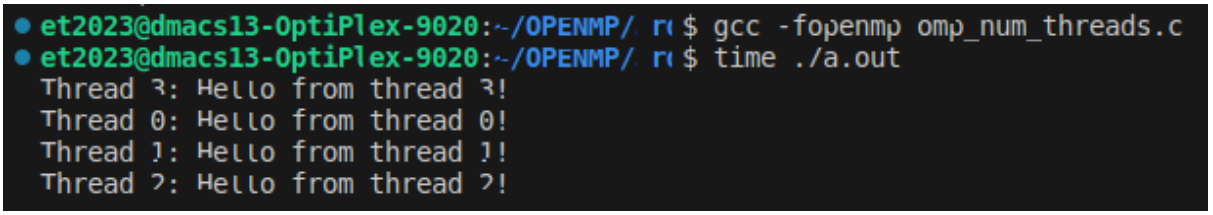
OMP_NUM_THREADS list

The OMP_NUM_THREADS environment variable is used to control the number of threads that OpenMP should use for parallel regions. By setting this variable, you can specify the desired number of threads to be employed for parallel execution within your program. This allows you to explicitly manage thread allocation according to your application's requirements, enabling you to optimize parallel performance and resource utilization. The value provided to OMP_NUM_THREADS determines the number of threads that will be created and utilized by OpenMP when executing parallel sections of code.

```
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
int main() {
  putenv("OMP_NUM_THREADS=3");
  #pragma omp parallel
```

```
    {
        int thread_id = omp_get_thread_num();
        printf("Thread %d: Hello from thread %d!\n", thread_id, thread_id);
    }
    return 0;
}
```

Output:



Explanation:

The above code demonstrates the use of the OMP_NUM_THREADS environment variable to control the number of threads used in parallel regions within an OpenMP program. In the code, the putenv function is used to set the value of OMP_NUM_THREADS to 3, indicating that the program should use three threads for parallel execution. Within the parallel region, each thread identifies its thread ID using omp_get_thread_num() and then prints a message indicating its thread ID. As a result, when the program is executed, three threads are spawned, each printing its unique thread ID, showcasing the influence of the OMP_NUM_THREADS environment variable on the number of threads utilized for parallel execution.