

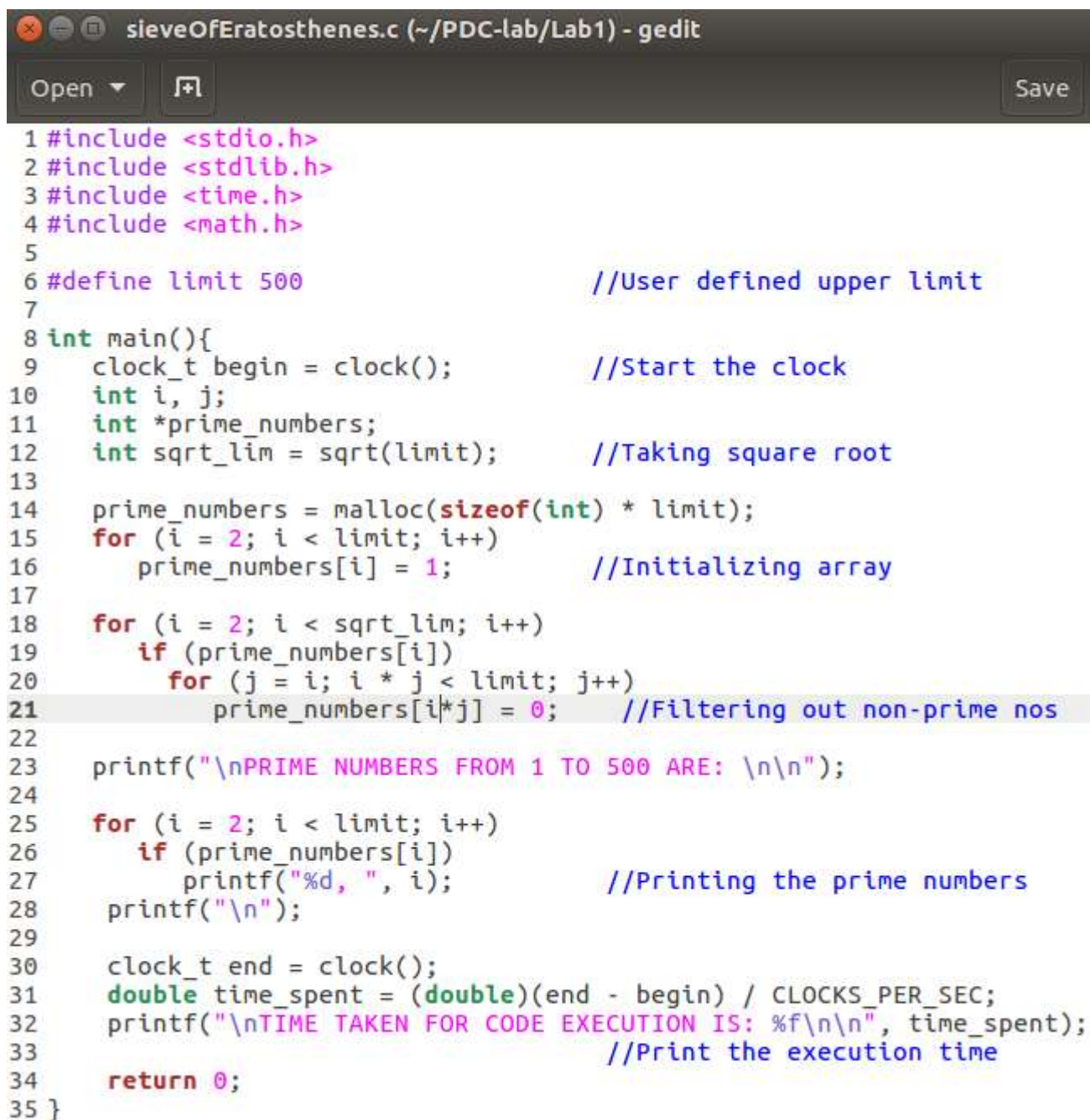
## OPENMP – SIEVE OF ERATOSTHENES

## LAB 1

**Aim:** To understand and implement firstprivate, lastprivate, parallel for and section constructs in OpenMP, for Sieve of Eratosthenes.

CODE AND OUTPUTS

## 1. Sieve of Eratosthenes – CODE WITHOUT OpenMP



```
sieveOfEratosthenes.c (~/PDC-lab/Lab1) - gedit
Open [icon] Save

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <time.h>
4 #include <math.h>
5
6 #define limit 500 //User defined upper limit
7
8 int main(){
9     clock_t begin = clock(); //Start the clock
10    int i, j;
11    int *prime_numbers;
12    int sqrt_lim = sqrt(limit); //Taking square root
13
14    prime_numbers = malloc(sizeof(int) * limit);
15    for (i = 2; i < limit; i++)
16        prime_numbers[i] = 1; //Initializing array
17
18    for (i = 2; i < sqrt_lim; i++)
19        if (prime_numbers[i])
20            for (j = i; i * j < limit; j++)
21                prime_numbers[i*j] = 0; //Filtering out non-prime nos
22
23    printf("\nPRIME NUMBERS FROM 1 TO 500 ARE: \n\n");
24
25    for (i = 2; i < limit; i++)
26        if (prime_numbers[i])
27            printf("%d, ", i); //Printing the prime numbers
28    printf("\n");
29
30    clock_t end = clock();
31    double time_spent = (double)(end - begin) / CLOCKS_PER_SEC;
32    printf("\nTIME TAKEN FOR CODE EXECUTION IS: %f\n\n", time_spent);
33    //Print the execution time
34    return 0;
35 }
```

**OUTPUT**

```
dhruv@dhruv-Inspiron-5559: ~/PDC-lab
dhruv@dhruv-Inspiron-5559:~/PDC-lab$ gcc -o sieveOfEratosthenes sieveOfEratosthenes.c
dhruv@dhruv-Inspiron-5559:~/PDC-lab$ ./sieveOfEratosthenes
```

```
dhruv@dhruv-Inspiron-5559:~/PDC-lab$ ./sieveOfEratosthenes

PRIME NUMBERS FROM 1 TO 500 ARE:

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67,
71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499,

TIME TAKEN FOR CODE EXECUTION IS: 0.000459
```

```
dhruv@dhruv-Inspiron-5559:~/PDC-lab$ ./sieveOfEratosthenes

PRIME NUMBERS FROM 1 TO 500 ARE:

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67,
71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499,

TIME TAKEN FOR CODE EXECUTION IS: 0.000095
```

**INFERENCE**

The inference from the above two screenshots is that although the prime numbers are found through a single thread, the execution time varies. This can be explained by the scheduling of this thread by the OS at runtime.

**USING THE PARALLEL FOR CONSTRUCT (Continued)**

## 2. Sieve of Eratosthenes – USING THE FirstPrivate, LastPrivate AND PARALLEL FOR CONSTRUCT

```

sieveOfErato_FPLP.c (~/.PDC-lab/Lab1) - gedit
Open  [+l] Save

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <time.h>
4 #include <omp.h>
5 #include <math.h>
6
7 #define limit 500 //User defined upper limit
8
9 int main(){
10     clock_t begin = clock(); //Start the clock
11     int i,j;
12     int *prime_numbers;
13     int sqrt_lim = sqrt(limit); //Taking square root
14
15     prime_numbers = malloc(sizeof(int) * limit);
16     for (i = 2; i < limit; i++)
17         prime_numbers[i] = 1; //Initializing array
18
19     # pragma omp parallel for private(i) firstprivate(prime_numbers)
20     lastprivate(prime_numbers)
21     for (i = 2; i < sqrt_lim; i++)
22         if (prime_numbers[i])
23             for (j = i; i*j < limit; j++)
24                 prime_numbers[i*j] = 0; //Filtering out non-prime nos
25
26     printf("\nPRIME NUMBERS FROM 1 TO 500 ARE: \n\n");
27
28     for (i = 2; i < limit; i++)
29         if (prime_numbers[i])
30             printf("%d, ", i); //Printing the prime numbers
31             printf("\n");
32
33     clock_t end = clock();
34     double time_spent = (double)(end - begin) / CLOCKS_PER_SEC;
35     printf("\nTIME TAKEN FOR CODE EXECUTION IS: %f\n\n", time_spent);
36     //Print the execution time
37     return 0;
38 }

```

## OUTPUT

```

dhruv@dhruv-Inspiron-5559: ~/.PDC-lab/Lab1
dhruv@dhruv-Inspiron-5559: ~/.PDC-lab/Lab1$ export OMP_NUM_THREADS=8
dhruv@dhruv-Inspiron-5559: ~/.PDC-lab/Lab1$

```



```
dhruv@dhruv-Inspiron-5559: ~/PDC-lab
dhruv@dhruv-Inspiron-5559:~/PDC-lab$ gcc -o omp_sieveOfErato_FPLP -fopenmp sieveOfErato_FPLP.c
dhruv@dhruv-Inspiron-5559:~/PDC-lab$ ./omp_sieveOfErato_FPLP
```

```
dhruv@dhruv-Inspiron-5559:~/PDC-lab$ ./omp_sieveOfErato_FPLP

PRIME NUMBERS FROM 1 TO 500 ARE:

2, 3, 5, 7, 11, 13, 17, 19, 23, 25, 29, 31, 37, 41, 43, 46, 47, 53, 59,
61, 62, 67, 71, 73, 77, 79, 83, 86, 89, 91, 97, 101, 103, 107, 109, 113,
119, 127, 131, 133, 137, 139, 143, 149, 151, 157, 161, 163, 167, 173,
175, 178, 179, 181, 184, 185, 188, 191, 193, 197, 199, 203, 211, 212,
217, 223, 227, 229, 233, 239, 241, 245, 251, 257, 259, 263, 269, 271,
277, 281, 283, 287, 293, 301, 307, 311, 313, 317, 329, 331, 335, 337, 343,
347, 349, 353, 355, 359, 365, 367, 371, 373, 379, 383, 389, 395, 397,
401, 409, 413, 415, 419, 421, 427, 431, 433, 439, 443, 449, 457, 461,
463, 467, 469, 479, 487, 491, 497, 499,

TIME TAKEN FOR CODE EXECUTION IS: 0.002309
```

```
dhruv@dhruv-Inspiron-5559:~/PDC-lab$ ./omp_sieveOfErato_FPLP

PRIME NUMBERS FROM 1 TO 500 ARE:

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67,
71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149,
151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227,
229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307,
311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389,
397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467,
479, 487, 491, 499,

TIME TAKEN FOR CODE EXECUTION IS: 0.000507
```

### INFERENCE

Number of threads used in both cases were 8. However, the execution times vary widely. This can be explained by different scheduling times of the various threads by the OS.

### SECTIONS CONSTRUCT (Continued)

### 3. Sieve of Eratosthenes – USING THE SECTIONS CONSTRUCT

```
sieveOfErato_Section.c (~/.PDC-lab/Lab1) - gedit
Open Save

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <time.h>
4 #include <omp.h>
5 #include <math.h>
6
7 #define limit 500 //User defined upper limit
8
9 int main(){
10     clock_t begin = clock(); //Start the clock
11     int i,j;
12     int *prime_numbers;
13     int sqrt_lim = sqrt(limit); //Taking square root
14
15     prime_numbers = malloc(sizeof(int) * limit);
16     for (i = 2; i < limit; i++)
17         prime_numbers[i] = 1; //Initializing array
18
19     # pragma omp parallel sections
20     {
21         # pragma omp section
22         for (i = 2; i < sqrt_lim; i++)
23             if (prime_numbers[i])
24                 for (j = i; i*j < limit; j++)
25                     prime_numbers[i*j] = 0; //Filtering out non-prime nos
26     }
27     printf("\nPRIME NUMBERS FROM 1 TO 500 ARE: \n\n");
28
29     for (i = 2; i < limit; i++)
30         if (prime_numbers[i])
31             printf("%d, ", i); //Printing the prime numbers
32     printf("\n");
33
34     clock_t end = clock(); //Stop the clock
35     double time_spent = (double)(end - begin) / CLOCKS_PER_SEC;
36     printf("\nTIME TAKEN FOR CODE EXECUTION IS: %f\n\n", time_spent);
37     //Print the execution time
38     return 0;
```

#### OUTPUT

```
dhruv@dhruv-Inspiron-5559: ~/.PDC-lab/Lab1
dhruv@dhruv-Inspiron-5559: ~/.PDC-lab/Lab1$ gcc -o omp_sieveOfErato_Section -fopenmp sieveOfErato_Section.c
dhruv@dhruv-Inspiron-5559: ~/.PDC-lab/Lab1$ ./omp_sieveOfErato_Section
```

```
dhruv@dhruv-Inspiron-5559:~/PDC-lab$ ./omp_sieveOfErato_Section
```

```
PRIME NUMBERS FROM 1 TO 500 ARE:
```

```
2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67,
71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 1
49, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 22
7, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307
, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389,
397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467,
479, 487, 491, 499,
```

```
TIME TAKEN FOR CODE EXECUTION IS: 0.002298
```

```
dhruv@dhruv-Inspiron-5559:~/PDC-lab$ ./omp_sieveOfErato_Section
```

```
PRIME NUMBERS FROM 1 TO 500 ARE:
```

```
2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67,
71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 1
49, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 22
7, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307
, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389,
397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467,
479, 487, 491, 499,
```

```
TIME TAKEN FOR CODE EXECUTION IS: 0.000532
```

### INFERENCE

Number of threads used in both cases were 8. However, the execution times vary widely. This can be explained by different scheduling times of the various threads by the OS.