

```
1 //-----
2 // Sieve of Eratosthenes
3 //-----
4 #if (defined __linux__ || (defined _AIX)
5     #include <sys/types.h>
6     #include <sys/stat.h>
7     #include <unistd.h>
8 #elif (defined _WIN32) || (defined _WIN64)
9     #include <conio.h>
10    #include <direct.h>
11 #endif
12
13 #include <mpi.h>
14 #include <stdlib.h>
15 #include <math.h>
16 #include <time.h>
17 #include <iostream>
18 using namespace std;
19 //-----
20 #define MIN(a,b) ((a)<(b)?(a):(b))
21 #define BLOCK_LOW(rank,size,n) ((rank)*(n)/(size))
22 #define BLOCK_HIGH(rank,size,n) (BLOCK_LOW((rank)+1,size,n)-1)
23 #define BLOCK_SIZE(rank,size,n) (BLOCK_HIGH(rank,size,n)-BLOCK_LOW(rank,size,n)+1)
24 //-----
25 int main(int argc, char* argv[])
26 {
27     int count; /* Local prime count */
28     double elapsed_time; /* Parallel execution time */
29     int first; /* Index of first multiple */
30     int global_count; /* Global prime count */
31     // int high_value; /* Highest value on this proc */
32     int i;
33     int size; /* Number of processes */
34     int rank; /* Process rank number */
35     int index; /* Index of current prime */
36     int low_value; /* Lowest value on this proc */
37     char* marked; /* Portion of 2,...,'ri' */
38     // int marked[1000];
39     int n; /* Sieving from 2, . . . , 'ri' */
40     int prime; /* Current prime */
41     int size_of_marked; /* Elements in 'marked' */
42     int number;
43
44     MPI_Init(&argc, &argv);
45
46     MPI_Barrier(MPI_COMM_WORLD); /* Start the timer */
47     elapsed_time = -MPI_Wtime();
48
49     MPI_Comm_size(MPI_COMM_WORLD, &size);
50     MPI_Comm_rank(MPI_COMM_WORLD, &rank);
51
52     if (!rank) {
53         cout << "\nWhen you write out a positive integer as a parameter after\n"
```

```
54     << "7B_Sieve_for_prime_numbers.exe, the program will write out\n"
55     << "all primes that are less than or equal to this parameter\n"
56     << "using a sieve of Eratosthenes." << endl;
57
58     cout << "\nThere are " << size << " processes.\n\n";
59 }
60
61 if (argc != 2) {
62     if (!rank)
63         cout << "You did not specify the input parameter as mentioned above.\n";
64     MPI_Finalize();
65     exit(1);
66 }
67
68 n = atoi(argv[1]);
69
70 /* Figure out this process's share of the array, as well as
71 the integers represented by the first and last array elements */
72
73 low_value = 2 + BLOCK_LOW(rank, size, n - 1);
74 // high_value = 2 + BLOCK_HIGH(rank, size, n-1);
75 size_of_marked = BLOCK_SIZE(rank, size, n - 1);
76
77 // ak plati nasledujuca podmienka, potom algoritmus nepracuje
78 if ((2 + (n - 1) / size) < (int)sqrt((double)n)) {
79     if (!rank) printf("Too many processes \n ");
80     MPI_Finalize();
81     exit(1);
82 }
83
84 /* Allocate this process's share of the array. */
85
86 marked = (char*)malloc(n);
87
88 if (marked == NULL) {
89     printf("Cannot allocate enough memory \n");
90     MPI_Finalize();
91     exit(1);
92 }
93
94 if (!rank)
95     cout << "Primes less than or equal to " << n << " are:\n";
96
97 for (i = 0; i < size_of_marked; i++)
98     marked[i] = 0;
99 if (!rank)
100     index = 0;
101 prime = 2;
102 do {
103     if (prime * prime > low_value)
104         first = prime * prime - low_value;
105     else {
106         if (!(low_value % prime))
```

```
107         first = 0;
108     else
109         first = prime - (low_value % prime);
110     }
111     for (i = first; i < size_of_marked; i += prime)
112         marked[i] = 1;
113     if (!rank) {
114         while (marked[++index]);
115         prime = index + 2;
116     }
117     MPI_Bcast(&prime, 1, MPI_INT, 0, MPI_COMM_WORLD);
118 } while (prime * prime <= n);
119
120 count = 0;
121 for (i = 0; i < size_of_marked; i++)
122     if (!marked[i])
123         count++;
124
125 MPI_Reduce(&count, &global_count, 1, MPI_INT, MPI_SUM, 0, MPI_COMM_WORLD);
126
127 if (rank > 0)
128     MPI_Ssend(&marked[0], size_of_marked, MPI_CHAR, 0, 0, MPI_COMM_WORLD);
129 if (rank == 0)
130     for (i = 1; i < size; i++)
131         MPI_Recv(&marked[BLOCK_LOW(i, size, n - 1)], BLOCK_SIZE(i, size,
132             n - 1), MPI_CHAR, i, 0, MPI_COMM_WORLD, MPI_STATUSES_IGNORE);
133
134 if (rank == 0) {
135     number = 0;
136     for (i = 0; i < n - 1; i++)
137         if (!marked[i]) {
138             cout << i + 2 << " ";
139             number++;
140             if (number % 20 == 19)
141                 cout << "\n";
142         }
143 }
144
145 elapsed_time += MPI_Wtime(); /* Stop the timer */
146
147 MPI_Barrier(MPI_COMM_WORLD);
148 if (!rank) {
149     printf("\n\n%d primes are less than or equal to %d\n", global_count, n);
150     printf("\nTotal elapsed time: %.6f s\n", elapsed_time);
151 }
152 free(marked);
153
154 fflush(stdout);
155 MPI_Finalize();
156
157 return 0;
158 }
159 //-----
```