Parallel Programming

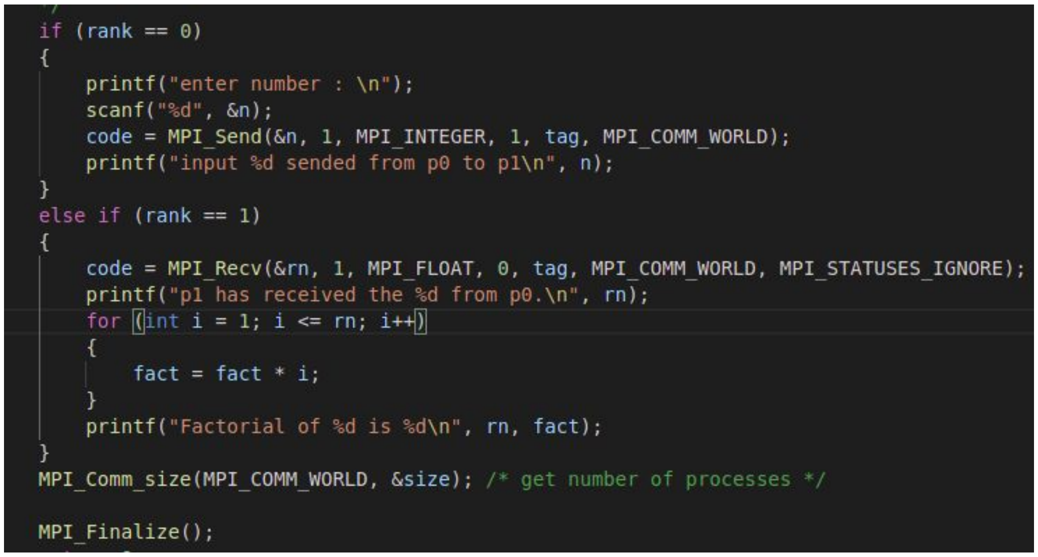
REPORT - PW6 MPI

Nafila Amirli

Exercise 1

Write an MPI program with two processes in such a way that the process 0 reads an integer number using scanf. Process 0 sends the number to process 1, which calculates the factorial of the number and returns the results to process 0.

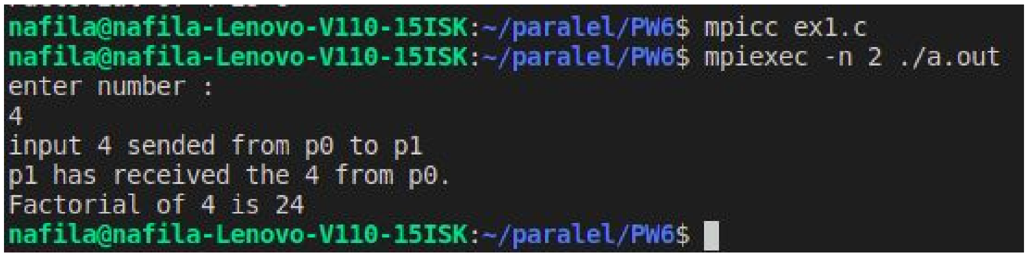
Solution



```
if (rank == 0)
{
    printf("enter number : \n");
    scanf("%d", &n);
    code = MPI_Send(&n, 1, MPI_INTEGER, 1, tag, MPI_COMM_WORLD);
    printf("input %d sended from p0 to p1\n", n);
}
else if (rank == 1)
{
    code = MPI_Recv(&rn, 1, MPI_FLOAT, 0, tag, MPI_COMM_WORLD, MPI_STATUSES_IGNORE);
    printf("p1 has received the %d from p0.\n", rn);
    for (int i = 1; i <= rn; i++)
    {
        fact = fact * i;
    }
    printf("Factorial of %d is %d\n", rn, fact);
}
MPI_Comm_size(MPI_COMM_WORLD, &size); /* get number of processes */

MPI_Finalize();
```

Terminal result:



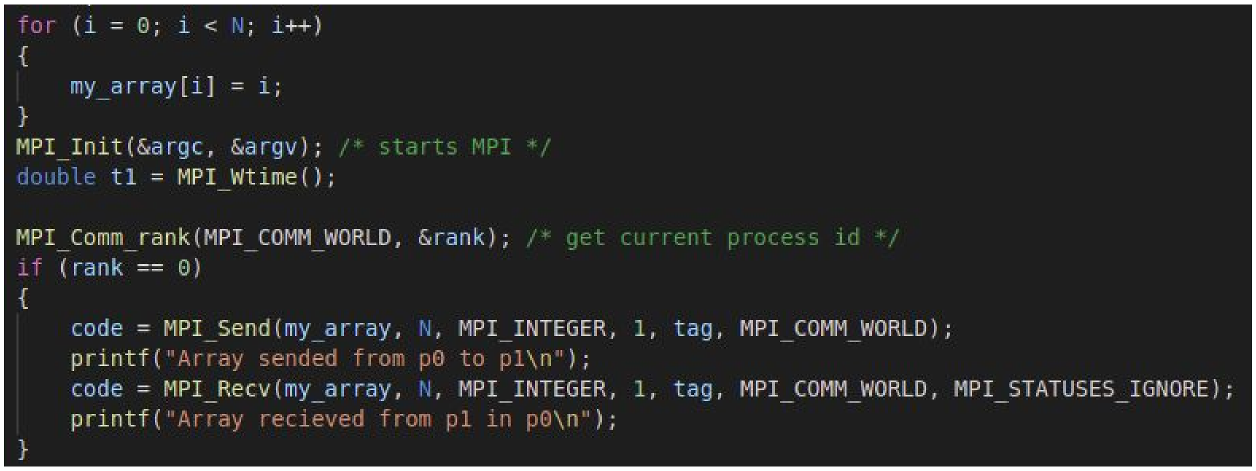
```
nafila@nafila-Lenovo-V110-15ISK:~/paralel/PW6$ mpicc ex1.c
nafila@nafila-Lenovo-V110-15ISK:~/paralel/PW6$ mpiexec -n 2 ./a.out
enter number :
4
input 4 sended from p0 to p1
p1 has received the 4 from p0.
Factorial of 4 is 24
nafila@nafila-Lenovo-V110-15ISK:~/paralel/PW6$
```

Exercise 2

Consider that process 0 has an array of integer numbers where we should compute the factorial of all the elements in the array. Modify the program in order to send the array to process 1 that computes all the factorials and return the results to process 0.

Solution

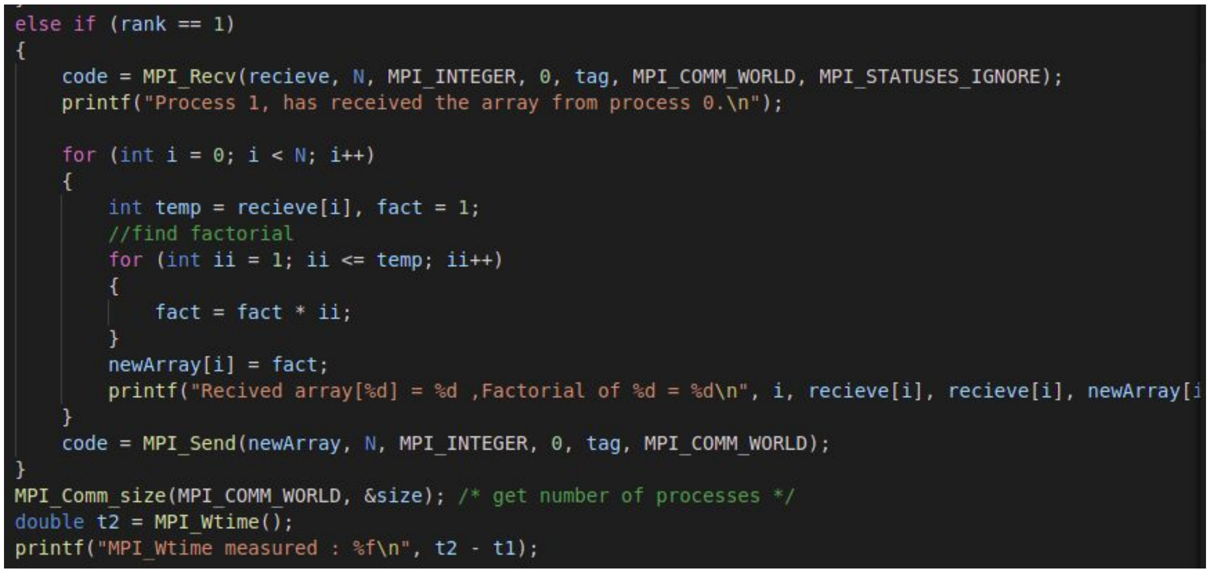
Array is initialized then sended to process 1:



```
for (i = 0; i < N; i++)
{
    my_array[i] = i;
}
MPI_Init(&argc, &argv); /* starts MPI */
double t1 = MPI_Wtime();

MPI_Comm_rank(MPI_COMM_WORLD, &rank); /* get current process id */
if (rank == 0)
{
    code = MPI_Send(my_array, N, MPI_INTEGER, 1, tag, MPI_COMM_WORLD);
    printf("Array sended from p0 to p1\n");
    code = MPI_Recv(my_array, N, MPI_INTEGER, 1, tag, MPI_COMM_WORLD, MPI_STATUSES_IGNORE);
    printf("Array recieved from p1 in p0\n");
}
```

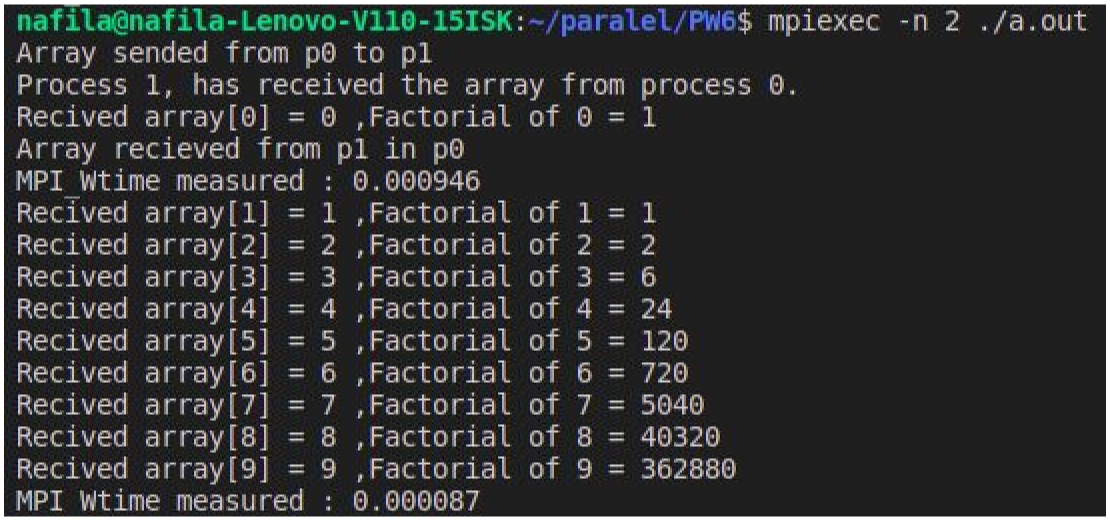
Process 1 receives the array and computes the factorials of each element then sends the array back to process 0:



```
else if (rank == 1)
{
    code = MPI_Recv(recieve, N, MPI_INTEGER, 0, tag, MPI_COMM_WORLD, MPI_STATUSES_IGNORE);
    printf("Process 1, has received the array from process 0.\n");

    for (int i = 0; i < N; i++)
    {
        int temp = recieve[i], fact = 1;
        //find factorial
        for (int ii = 1; ii <= temp; ii++)
        {
            fact = fact * ii;
        }
        newArray[i] = fact;
        printf("Recived array[%d] = %d ,Factorial of %d = %d\n", i, recieve[i], recieve[i], newArray[i
    }
    code = MPI_Send(newArray, N, MPI_INTEGER, 0, tag, MPI_COMM_WORLD);
}
MPI_Comm_size(MPI_COMM_WORLD, &size); /* get number of processes */
double t2 = MPI_Wtime();
printf("MPI_Wtime measured : %f\n", t2 - t1);
```

Terminal result:



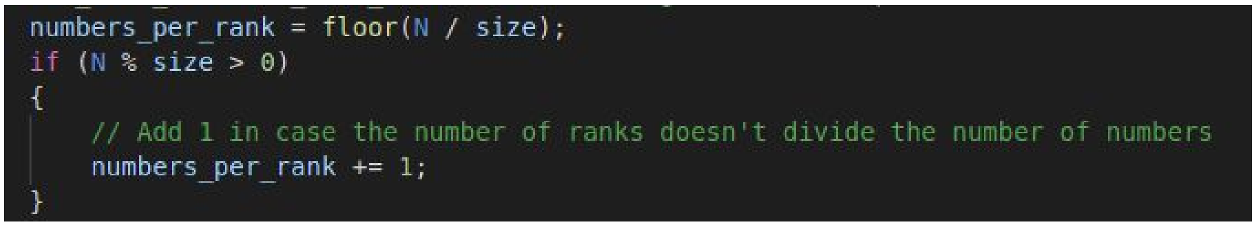
```
nafila@nafila-Lenovo-V110-15ISK:~/paralel/PW6$ mpiexec -n 2 ./a.out
Array sended from p0 to p1
Process 1, has received the array from process 0.
Recived array[0] = 0 ,Factorial of 0 = 1
Array recieved from p1 in p0
MPI_Wtime measured : 0.000946
Recived array[1] = 1 ,Factorial of 1 = 1
Recived array[2] = 2 ,Factorial of 2 = 2
Recived array[3] = 3 ,Factorial of 3 = 6
Recived array[4] = 4 ,Factorial of 4 = 24
Recived array[5] = 5 ,Factorial of 5 = 120
Recived array[6] = 6 ,Factorial of 6 = 720
Recived array[7] = 7 ,Factorial of 7 = 5040
Recived array[8] = 8 ,Factorial of 8 = 40320
Recived array[9] = 9 ,Factorial of 9 = 362880
MPI_Wtime measured : 0.000087
```

Exercise 3

Consider that we have n processes, modify the program in order to distribute the calculation over all available processes and measure the gain in execution time compared to exercise 2.

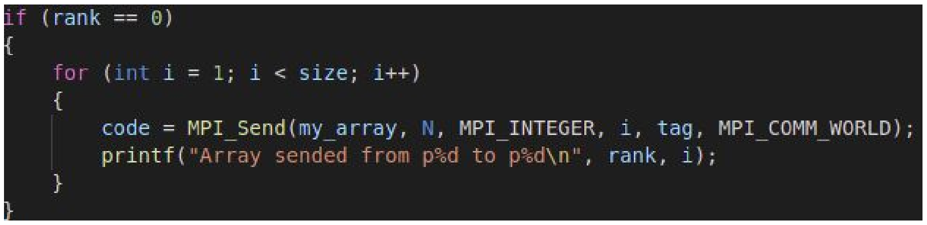
Solution

Firstly new variable number_per_rank is declared to divide for loop among processes:



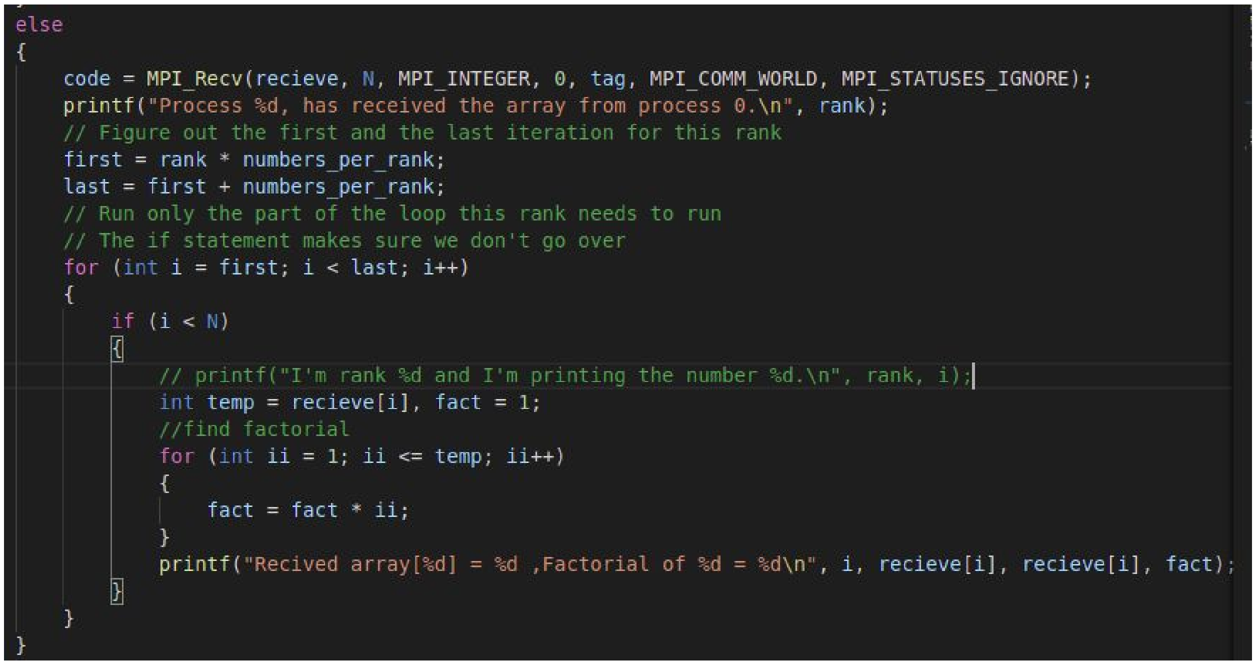
```
numbers_per_rank = floor(N / size);
if (N % size > 0)
{
    // Add 1 in case the number of ranks doesn't divide the number of numbers
    numbers_per_rank += 1;
}
```

Then the array is sended to each processes:



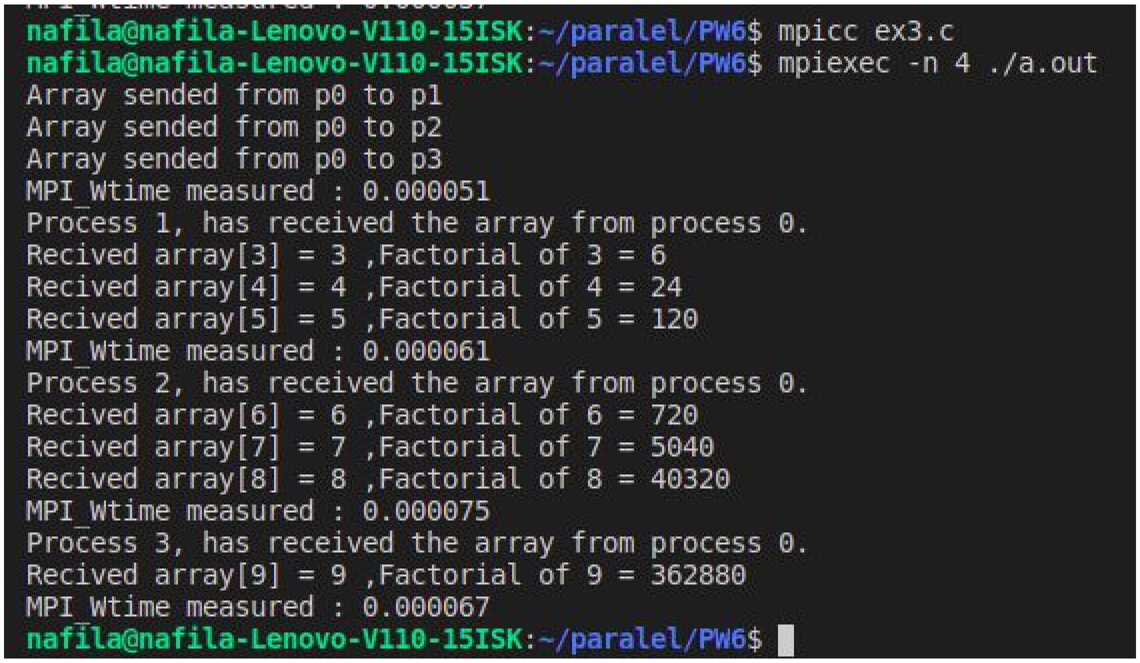
```
if (rank == 0)
{
    for (int i = 1; i < size; i++)
    {
        code = MPI_Send(my_array, N, MPI_INTEGER, i, tag, MPI_COMM_WORLD);
        printf("Array sended from p%d to p%d\n", rank, i);
    }
}
```

Then it is received by each process and each process compute parts of for loop which is identified by variables called first and last:



```
else
{
    code = MPI_Recv(recieve, N, MPI_INTEGER, 0, tag, MPI_COMM_WORLD, MPI_STATUSES_IGNORE);
    printf("Process %d, has received the array from process 0.\n", rank);
    // Figure out the first and the last iteration for this rank
    first = rank * numbers_per_rank;
    last = first + numbers_per_rank;
    // Run only the part of the loop this rank needs to run
    // The if statement makes sure we don't go over
    for (int i = first; i < last; i++)
    {
        if (i < N)
        {
            // printf("I'm rank %d and I'm printing the number %d.\n", rank, i);
            int temp = recieve[i], fact = 1;
            //find factorial
            for (int ii = 1; ii <= temp; ii++)
            {
                fact = fact * ii;
            }
            printf("Recived array[%d] = %d ,Factorial of %d = %d\n", i, recieve[i], recieve[i], fact);
        }
    }
}
```

Terminal output:



```
nafila@nafila-Lenovo-V110-15ISK:~/paralel/PW6$ mpicc ex3.c
nafila@nafila-Lenovo-V110-15ISK:~/paralel/PW6$ mpiexec -n 4 ./a.out
Array sended from p0 to p1
Array sended from p0 to p2
Array sended from p0 to p3
MPI_Wtime measured : 0.000051
Process 1, has received the array from process 0.
Recived array[3] = 3 ,Factorial of 3 = 6
Recived array[4] = 4 ,Factorial of 4 = 24
Recived array[5] = 5 ,Factorial of 5 = 120
MPI_Wtime measured : 0.000061
Process 2, has received the array from process 0.
Recived array[6] = 6 ,Factorial of 6 = 720
Recived array[7] = 7 ,Factorial of 7 = 5040
Recived array[8] = 8 ,Factorial of 8 = 40320
MPI_Wtime measured : 0.000075
Process 3, has received the array from process 0.
Recived array[9] = 9 ,Factorial of 9 = 362880
MPI_Wtime measured : 0.000067
nafila@nafila-Lenovo-V110-15ISK:~/paralel/PW6$
```

Time comparison with ex2

Gain→ (0.000946+0.000087)/ (0.000051+0.000061+0.000075+0.000067)=4.07

As seen from the above result when we use several processes concurrently, the computation time(even in total) is much less than when we compute all in one process.