МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

**Федеральное государственное бюджетное образовательное учреждение высшего образования  
 «Кемеровский государственный университет»**

**Институт фундаментальных наук**

**ДОМАШНЯЯ РАБОТА №7**

**ПО ДИСЦИПЛИНЕ**

**“Технологии параллельных вычислений”**

студента 3 курса

**Сулима Роман Иванович**

Направление 02.03.02 – Фундаментальная информатика и информационные технологии

Преподаватель:

к-т физ.-мат.наук, доцент

С.В. Стуколов

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Работа защищена:

“\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_201\_г.

с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_

Кемерово 2021

СОДЕРЖАНИЕ

[1. Постановка задачи 2](#_Toc86440764)

[2. Описание используемых функций 2](#_Toc86440765)

[3. Реализация 2](#_Toc86440766)

[Задача 3. 2](#_Toc86440767)

[Задача 4. 3](#_Toc86440768)

[Заключение 6](#_Toc86440769)

# 1. Постановка задачи

Выполните упражнения, приведенные в конце параграфа - 3-4 задачи. Оформляете в одном отчете - номер задачи, текст условия, текст программы с поясняющими комментариями в программе, скрин компиляции и скрин запуска на несколько процессов. Удачи!

# 2. Описание используемых функций

MPI\_Sendrecv – функция предназначенная для одновременного отправление и получения данных между процессами.

# 3. Реализация

## Задача 3.

Постановка задачи:

Создайте и выполните программу, используя коммуникационную   
функцию (MPI\_Sendrecv), реализующую алгоритм передачи данных по   
двум кольцам: нечетные процессора образуют первое кольцо, четные –   
второе.

Программный код:

#include <stdio.h>

#include "mpi.h"

#include "iostream"

using namespace std;

int main(int argc, char \*argv[])

{

    int rank;

    int size;

    int a = 0;

    MPI\_Init(&argc, &argv);

    MPI\_Status stat;

    MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);

    MPI\_Comm\_size(MPI\_COMM\_WORLD, &size);

    if(rank == 0)

    {

        MPI\_Sendrecv(&rank, 1, MPI\_INT, 2, 777, &a, 1, MPI\_INT, (size-2), 777, MPI\_COMM\_WORLD, &stat);

        cout << "rank = " << rank << " A = " << a << endl;

    }

    if(rank == 1)

    {

        MPI\_Sendrecv(&rank, 1, MPI\_INT, 3, 777, &a, 1, MPI\_INT, (size-1), 777, MPI\_COMM\_WORLD, &stat);

        cout << "rank = " << rank << " A = " << a << endl;

    }

    else if(rank == size-1)

    {

        MPI\_Sendrecv(&rank, 1, MPI\_INT, 1, 777, &a, 1, MPI\_INT, (rank-2), 777, MPI\_COMM\_WORLD, &stat);

        cout << "rank = " << rank << " A = " << a << endl;

    }

    else if(rank == size-2)

    {

        MPI\_Sendrecv(&rank, 1, MPI\_INT, 0, 777, &a, 1, MPI\_INT, (rank-2), 777, MPI\_COMM\_WORLD, &stat);

        cout << "rank = " << rank << " A = " << a << endl;

    }

    else{

        MPI\_Sendrecv(&rank, 1, MPI\_INT, (rank+2), 777, &a, 1, MPI\_INT, (rank-2), 777, MPI\_COMM\_WORLD, &stat);

        cout << "rank = " << rank << " A = " << a << endl;

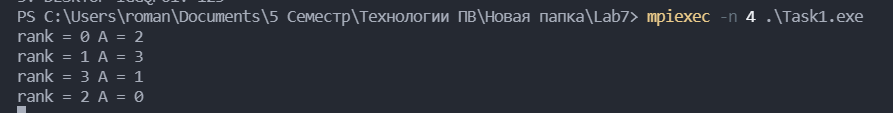
    }

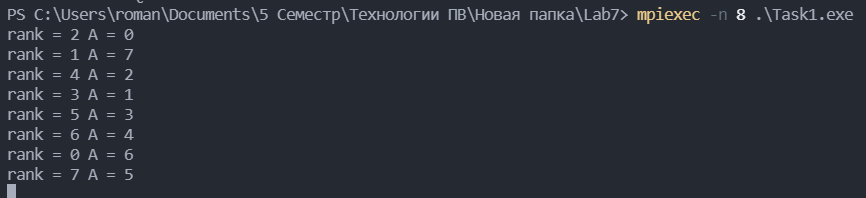
    MPI\_Finalize();

    return 0;

}

Выполнение:

****

****

## Задача 4.

Напишите программу, которая будет выполняться на двух процессах и   
реализовывать следующий алгоритм:   
− На 0-м процессе создается динамический массив a[n], на 1-м – b[n];   
− На обоих процессах засекается время;   
− Осуществляется обмен массивами;   
− На обоих процессах замеряется время и производится вывод на экран   
номера процесса и затраченное время на передачу данных.   
Реалиазуйте два варианта данной программы: 1) для передачи данных   
используйте стандартные функции MPI\_Send, MPI\_Recv; 2) для передачи   
данных используйте совмещенную функцию MPI\_Sendrecv. Выполните   
сравнение времени передачи данных в зависимости от способа передачи и   
различных размеров передаваемых массивов.

С использованием MPI\_Sendreciv.

Программный код:

#include <stdio.h>

#include "mpi.h"

int main(int argc, char \*argv[])

{

    int rank;

    int size;

    int \*a, \*b, n = 4;

    double t1, t2;

    MPI\_Status stat;

    MPI\_Init(&argc, &argv);

    MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);

    MPI\_Comm\_size(MPI\_COMM\_WORLD, &size);

    if(rank==0)

    {

        int \*a = new int[n];

        for(int i = 0; i<n; i++)

        {

            a[i] = i;

        }

        int \*b = new int[n];

        t1 = MPI\_Wtime();

        MPI\_Sendrecv(&a[0], n, MPI\_INT, 1, 777, &b[0], n, MPI\_INT, 1, 777, MPI\_COMM\_WORLD, &stat);

        t2 = MPI\_Wtime();

        printf("rank = %d, time =  %f\n, b: ", rank, t2 - t1);

        for(int i =0; i<n;i++)

        {

            printf("%d ", b[i]);

        }

    }

    if(rank==1)

    {

        int \*b = new int[n];

        for(int i = 0; i<n; i++)

        {

            b[i] = i\*10;

        }

        int \*a = new int[n];

        t1 = MPI\_Wtime();

        MPI\_Sendrecv(&b[0], n, MPI\_INT, 0, 777, &a[0], n, MPI\_INT, 0, 777, MPI\_COMM\_WORLD, &stat);

        t2 = MPI\_Wtime();

        printf("rank = %d, time = %f\n, a: ", rank, t2 - t1);

        for(int i =0; i<n;i++)

        {

            printf("%d ", a[i]);

        }

    }

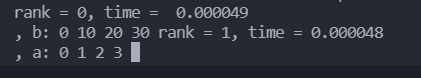
    MPI\_Finalize();

    return 0;

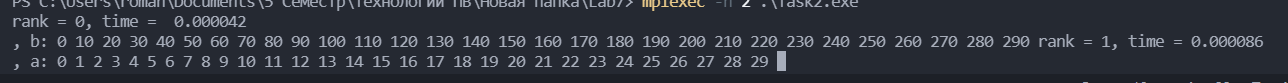
}

Выполнение:

При n = 4



При n = 30



Программный код:

#include <stdio.h>

#include "mpi.h"

int main(int argc, char \*argv[])

{

    int rank;

    int size;

    int \*a, \*b, n = 30;

    double t1, t2;

    MPI\_Status stat;

    MPI\_Init(&argc, &argv);

    MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);

    MPI\_Comm\_size(MPI\_COMM\_WORLD, &size);

    if(rank==0)

    {

        int \*a = new int[n];

        for(int i = 0; i<n; i++)

        {

            a[i] = i;

        }

        int \*b = new int[n];

        t1 = MPI\_Wtime();

        MPI\_Send(&a[0], n, MPI\_INT, 1, 777, MPI\_COMM\_WORLD);

        MPI\_Recv(&b[0], n, MPI\_INT, 1, 777, MPI\_COMM\_WORLD, &stat);

        t2 = MPI\_Wtime();

        printf("rank = %d, time = %f\n b: ", rank, t2 - t1);

        for(int i = 0; i<n; i++)

            printf("%d ", b[i]);

    }

    if(rank==1)

    {

        int \*b = new int[n];

        for(int i = 0; i<n; i++)

        {

            b[i] = i\*10;

        }

        int \*a = new int[n];

        t1 = MPI\_Wtime();

        MPI\_Send(&b[0], n, MPI\_INT, 0, 777, MPI\_COMM\_WORLD);

        MPI\_Recv(&a[0], n, MPI\_INT, 0, 777, MPI\_COMM\_WORLD, &stat);

        t2 = MPI\_Wtime();

        printf("rank = %d, time = %f\n a: ", rank, t2 - t1);

        for(int i = 0; i<n; i++)

            printf("%d ", a[i]);

    }

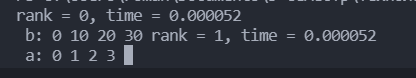
    MPI\_Finalize();

    return 0;

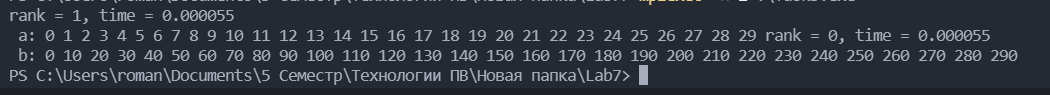
}

Выполнение:

При n = 4.



При n = 30.



Сравнение:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Send и Recv, n = 4 | MPI\_Sendrecv, n = 4 | Send и Recv, n = 30 | MPI\_Sendrecv, n = 30 |
| 0 – й процесс | 0.000052 | 0.000049 | 0.000055 | 0.000042 |
| 1 – й процесс | 0.000052 | 0.000048 | 0.000055 | 0.000086 |

# Заключение

Выполнив данную работу я научился использовать функцию MPI\_Sendrecv и разобрался как данную функцию можно использовать для программ, которые написаны через стандартные MPI функции.