МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение   
высшего профессионального образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА №14

РАБОТА ЗАЩИЩЕНА С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| старший преподаватель |  |  |  | А. Ю. Сыщиков |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 1  MPI, Пересылка данных |
|  |
| по дисциплине: [Системы с параллельной обработкой информации](https://pro.guap.ru/inside_s#subjects/2436975) |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. | 1742 |  |  |  | В.А. Седов |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2021

1. **Цель работы**

Переслать вектор, размерности M, N процессам, используя различные виды связи между процессами. Элементы вектора задаются произвольно. Элементы вектора пересылаемого и принятого вектора, а также время выполнения должны быть выведены на экран.

|  |  |  |  |
| --- | --- | --- | --- |
| № варианта | M | N | Функция |
| 16 | 10 | 3 | MPI\_Alltoall |

1. **Текст программы**

#include "mpi.h"

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#define M 2

static int gsize, myid;

void initArrayNew(int\* a) {

for (int i = 0; i < M; i++) {

a[i] = rand() % 7 + (myid % 3 + 1) + rand() % 20 \* (myid %2)+ (myid % 3 +1);

}

}

void printArray(int\* a, int R) {

printf("[ ");

for (int i = 0; i < R; i++) {

printf("%d ", a[i]);

}

printf("]\n");

}

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

int\* rbuf;

int\* sendbuf;

int namelen;

double startwtime = 0.0, endwtime = 0.0;

char processor\_name[MPI\_MAX\_PROCESSOR\_NAME];

MPI\_Init(&argc, &argv);

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

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

MPI\_Get\_processor\_name(processor\_name, &namelen);

//gsize = 9;

startwtime = MPI\_Wtime();

sendbuf = (int\*)malloc(M \* sizeof(int));

initArrayNew(sendbuf);

rbuf = (int\*)malloc(M \* gsize \* sizeof(int));

printf("Process %d on %s\n", myid, processor\_name);

printf("Massiv: ");

printArray(sendbuf, M);

fflush(stdout);

MPI\_Allgather(sendbuf, M, MPI\_INT, rbuf, M, MPI\_INT, MPI\_COMM\_WORLD);

//printf("id = %d, sendbuf = %d\n", myid, sendbuf[myid]);

//printf("rbuf = %d\n", rbuf[0]);

printf("Process %d on %s:", myid, processor\_name);

printArray(rbuf, M\*gsize);

fflush(stdout);

if (myid == 0) {

endwtime = MPI\_Wtime();

fprintf(stderr, "wall clock time = %f\n", endwtime - startwtime);

}

MPI\_Finalize();

return 0;

}

1. **Результат работы программы**

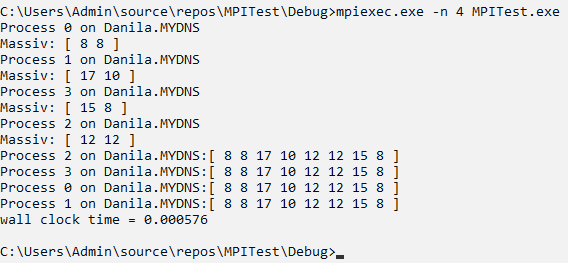


Рисунок 1. Результат работы программы.