МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №2
по дисциплине «Параллельные алгоритмы»
Тема: Использование аргументов джокера

Студентка гр. 9382 _____ Иерусалимов Н.

Преподаватель _____ Татаринов Ю.С.

Санкт-Петербург
2021

Цель работы

Научиться обмену данными между процессами и дальнейшее использование результатов.

Формулировка задания

*Имитация посылки пакета с маршрутом***.** Процесс 0 генерирует сообщение, которое состоит из маршрутной и информационной части. Маршрутная часть содержит последовательность номеров процессов – промежуточных адресатов. Промежуточный адресат, получив пакет, убирает из адресной части свой номер и передает сообщение дальше согласно списку. Окончательный адресат, получив посылку, отчитывается перед процессом 0.

Выполнение

Сначала генерируем очередь и объявляем переменные. Далее если ранк процесса равен 0 мы отправляем согласно очереди сообщение и саму очередь. Другой процесс получает эти данные убирает себя из очереди и передает дальше следующему процессу сообщение и очередь. Как только доходим до конца очереди выводим сообщение и подаем сигнал главному процессу что все закончилось хорошо. Код в приложении А

Тестирование

Очередь из 8 процессов рис. 1

```
nereus@Nereus:~/CLionProjects/untitled1$ mpirun -np 8 a.out
Invalid MIT-MAGIC-COOKIE-1 keyОтправил маршруты и сообщение для 1
Получил маршрут и сообщение (1)
Отправил маршрут и сообщение далее для  2 (1)
Получил маршрут и сообщение (2)
Отправил маршрут и сообщение далее для  3 (2)
Получил маршрут и сообщение (3)
Отправил маршрут и сообщение далее для  4 (3)
Получил маршрут и сообщение (5)
Отправил маршрут и сообщение далее для  6 (5)
Получил маршрут и сообщение (4)
Отправил маршрут и сообщение далее для  5 (4)
Получил маршрут и сообщение (6)
Отправил маршрут и сообщение далее для  7 (6)
Получил маршрут и сообщение (7)
Получил сообщение Панки хой (7)
Отправил подтверждение для 0 (7)
 (0)Дошли до конца маршрута, подтверждение от 7
nereus@Nereus:~/CLionProjects/untitled1$
```

Рис. 1

Тоже самое для 5 процессов рис. 2

```
nereus@Nereus:~/CLionProjects/untitled1$ mpirun -np 5 a.out
Invalid MIT-MAGIC-COOKIE-1 keyОтправил маршруты и сообщение для 1
Получил маршрут и сообщение (1)
Отправил маршрут и сообщение далее для  2 (1)
Получил маршрут и сообщение (2)
Отправил маршрут и сообщение далее для  3 (2)
Получил маршрут и сообщение (3)
Отправил маршрут и сообщение далее для  4 (3)
Получил маршрут и сообщение (4)
Получил сообщение Панки хой (4)
Отправил подтверждение для 0 (4)
 (0)Дошли до конца маршрута, подтверждение от 4
 (0)   nereus@Nereus:~/CLionProjects/untitled1$
```

Рис. 2

Выводы

В данной лабораторной работе были использованы аргументы джокера и мы смогли пройти по всем процессам по определенному правилам.

Приложение А.

Файл - main.cpp

```
#include <stdio.h>
#include "mpi.h"
#include <stdlib.h>


#define maxSymbol 1000


int* get_route_table(int procnum) {
    int* arrayWithRoute = calloc(procnum, sizeof(int));
    for (int i = 0; i < procnum; ++i) {
        arrayWithRoute[i] = i;
    }
    return arrayWithRoute;
}

int main(int argc, char* argv[]) {
    int procnum = 0, _id = 0;
    MPI_Status Status;
    MPI_Init(&argc, &argv);

    MPI_Comm_size(MPI_COMM_WORLD, &procnum);
    MPI_Comm_rank(MPI_COMM_WORLD, &_id);
```

```
    if (_id == 0) {
        int* message = (int*)malloc(sizeof(int) * procnum);

        int* route = get_route_table(procnum);
        char messageSend[maxSymbol] = "Панки хой";

        route[0] = -1;

        printf("Отправил маршруты и сообщение для %d\n (0)", route[1]);
        MPI_Send(route, procnum, MPI_INT, route[1], 0, MPI_COMM_WORLD);
        MPI_Send(messageSend, maxSymbol, MPI_CHAR, route[1], 0,
MPI_COMM_WORLD);



        MPI_Recv(message, procnum, MPI_INT, route[procnum - 1],
MPI_ANY_TAG, MPI_COMM_WORLD, &Status);

        printf("Дошли до конца маршрута, подтверждение от %d\n (0)",
route[procnum - 1]);

    } else {

        int* message = (int*)malloc(sizeof(int) * procnum);
        char* message2 = malloc(sizeof(char) * maxSymbol);

        MPI_Recv(message, procnum, MPI_INT, MPI_ANY_SOURCE, MPI_ANY_TAG,
MPI_COMM_WORLD, &Status);
        MPI_Recv(message2, maxSymbol, MPI_CHAR, MPI_ANY_SOURCE,
MPI_ANY_TAG, MPI_COMM_WORLD, &Status);

        message[_id] = -1;

        printf("Получил маршрут и сообщение (%d)\n", _id);
```

```
        if (_id < procnum - 1) {

            printf("Отправил маршрут и сообщение далее для  %d (%d)\n ",
message[_id + 1], _id);
            MPI_Send(message, procnum, MPI_INT, message[_id + 1], 0,
MPI_COMM_WORLD);
            MPI_Send(message2, maxSymbol, MPI_CHAR, message[_id + 1], 0,
MPI_COMM_WORLD);
        }
        else {

            printf("Получил сообщение %s (%d)\n", message2, _id);

            printf("Отправил подтверждение для 0 (%d)\n", _id);
            MPI_Send(message, procnum, MPI_INT, 0, 0, MPI_COMM_WORLD);
        }
    }

    MPI_Finalize();
    return 0;
}
```