

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

ОТЧЕТ
по лабораторной работе №00
по дисциплине «Параллельные алгоритмы».
Тема: Запуск параллельной программы на различном числе
одновременно работающих процессов, упорядочение вывода
результатов.

Студентка гр. 0382

Кривенцова Л.С.

Преподаватель

Татаринов Ю.С.

Санкт-Петербург

2022

Задание.

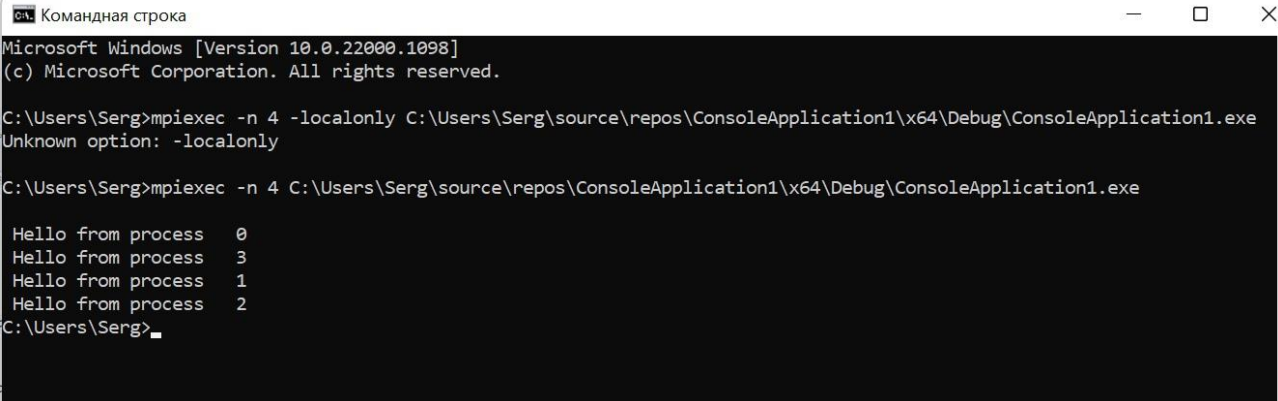
Запустить программу на 1,2 ... N процессах несколько раз. Проанализировать порядок вывода сообщений на экран. Вывести правило, определяющее порядок вывода сообщений. Модифицировать программу таким образом, чтобы порядок вывода сообщений на экран соответствовал номеру соответствующего процесса.

Листинг программы см. в Приложении А.

Выполнение работы.

В программе после определения своего ранга процесс в зависимости от него выполняет следующие действия. Все процессы, с рангами отличными от нуля, передают на нулевой процесс свой ранг. Нулевой процесс в цикле принимает сообщения от любого процесса с любым тегом. И сообщения от процессов выводятся "не по порядку". Причем, порядок вывода сообщений может изменяться от запуска к запуску. Причина происходящего кроется в том, что в приведенном примере цикл приема сообщений описан с помощью константы `MPI_ANY_SOURCE`. Передавая её в качестве ожидаемого источника, нулевой процесс принимает сообщение от любого другого процесса.

Результат работы исходной программы на 1..4 процессорах.



```
Командная строка
Microsoft Windows [Version 10.0.22000.1098]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Serg>mpiexec -n 4 -localonly C:\Users\Serg\source\repos\ConsoleApplication1\x64\Debug\ConsoleApplication1.exe
Unknown option: -localonly

C:\Users\Serg>mpiexec -n 4 C:\Users\Serg\source\repos\ConsoleApplication1\x64\Debug\ConsoleApplication1.exe

Hello from process 0
Hello from process 3
Hello from process 1
Hello from process 2
C:\Users\Serg>
```

Чтобы обеспечить последовательный прием данных от процессов в порядке возрастания их рангов на каждой итерации цикла мы будем принимать сообщение от конкретного процесса. Для этого заменим константу `MPI_ANY_SOURCE` в функцию `MPI_Recv` на переменную `i`.

Результат работы модифицированной программы на 1..4 процессорах.

```
C:\Users\Serg>mpiexec -n 4 C:\Users\Serg\source\repos\ConsoleApplication1\x64\Debug\ConsoleApplication1.exe
Hello from process 0
Hello from process 1
Hello from process 2
Hello from process 3
C:\Users\Serg>
```

Выводы.

Поскольку, будучи запущенными на разных машинах, процессы стартуют и выполняются с разными скоростями, порядок, в котором процессы будут входить в функцию передачи сообщений `MPI_Send`, будет случайным. Но изменив цикл приёма сообщений (избавившись от константы `MPI_ANY_SOURCE`) мы получаем требуемый результат: при запуске модифицированной программы порядок вывода сообщений всегда будет одним и тем же.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
#include <stdio.h>
#include "mpi.h"
int main(int argc, char* argv[]){
int ProcNum, ProcRank, RecvRank;
MPI_Status Status;
MPI_Init(&argc, &argv);
MPI_Comm_size(MPI_COMM_WORLD, &ProcNum);
MPI_Comm_rank(MPI_COMM_WORLD, &ProcRank);
if ( ProcRank == 0 ){
//Действия, выполняемые только процессом с рангом 0
printf ("\n Hello from process %3d", ProcRank);
for ( int i=1; i<ProcNum; i++ ) {
MPI_Recv(&RecvRank, 1, MPI_INT, MPI_ANY_SOURCE,
MPI_ANY_TAG, MPI_COMM_WORLD, &Status);
printf("\n Hello from process %3d", RecvRank); } }
else // Сообщение, отправляемое всеми процессами,
// кроме процесса с рангом 0
MPI_Send(&ProcRank,1,MPI_INT,0,0, MPI_COMM_WORLD);
MPI_Finalize();
return 0;
}
```

КОД МОДИФИЦИРОВАННОЙ ПРОГРАММЫ

```
#include <stdio.h>
#include "mpi.h"
int main(int argc, char* argv[]) {
    int rank, n, i, message;
    MPI_Status status;
    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &n);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    if (rank == 0) {
        printf("\n Hello from process %3d", rank);
        for (i = 1; i < n; i++) {
            MPI_Recv(&message, 1, MPI_INT, i,
                MPI_ANY_TAG, MPI_COMM_WORLD, &status);
            printf("\n Hello from process %3d", message);
        }
    }
```

```
    }  
    else MPI_Send(&rank, 1, MPI_INT, 0, 0, MPI_COMM_WORLD);  
    MPI_Finalize();  
    return 0;  
}
```