**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. 9382 |  | Иерусалимов Н. |
| Преподаватель |  | Татаринов Ю.С. |

Санкт-Петербург

2021

**Цель работы**

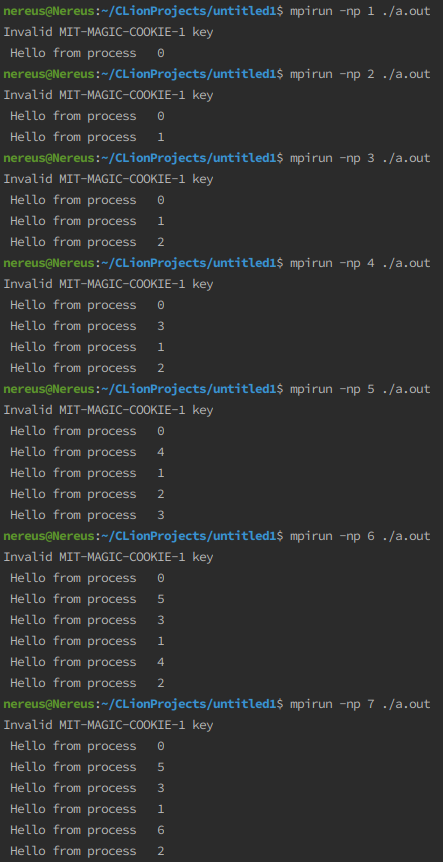
Запустить программу с MPI, изучить работу программы.

**Формулировка задания**

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.

#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);  
 }  
 printf("\n");  
 }  
 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;  
}

**Результат работы программы на 1,2 … N процессорах**



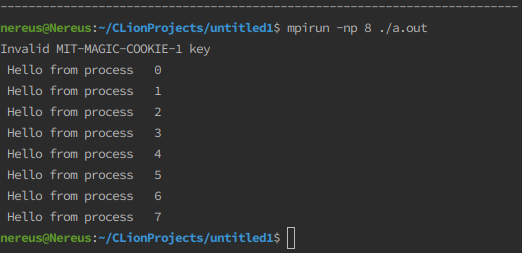
**Анализ порядка вывода сообщений на экран**

Номер процесса – его ранг. ProcRank отвечает за хранение ранга процесса, выполнившего вызов. Пересылка сообщения принимается процессом-получателем при помощи функции MPI\_Recv. С помощью него в нашей программе сообщения принимаются от любого процесса-отправителя с помощью MPI\_ANY\_SOURCE. Все сообщения начинают выводиться только после того, как ранг текущего процесса станет 0, печатается значение ранга текущего процесса, затем выполняется цикл для получения сообщений и их отображения в консоли. Порядок сообщений не определен заранее, так как зависит от условий выполнения параллельной программы.

**Модификация 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, i,  
 MPI\_ANY\_TAG, MPI\_COMM\_WORLD, &Status);  
 printf("\n Hello from process %3d", RecvRank);  
 }  
 printf("\n");  
 }  
 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;  
}

Результат вывода модифицированной программы:



Для того чтобы программа выводила сообщения по порядку принимающая функция должна принимать сообщения по порядку в цикле вместо MPI\_ANY\_SOURCE. Так как порядок строго определен, работа параллельных вычислений может быть замедлена.

**Выводы**

В ходе лабораторной работы была проанализирована и модифицирована данная программа MPI, были изучены базовые функции MPI.