**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра вычислительной техники**

**ОТЧЕТ**

**по лабораторной работе №1**

**по дисциплине «Параллельные вычисления»**

**Тема: «Запуск параллельной программы»**

| Студентка гр. 1307 |  | Голубев М.А. |
| --- | --- | --- |
| Преподаватель |  | Калмыков М.А. |

Санкт-Петербург

2025

**Цель работы.**

Освоить процесс запуска программы на С++ с применением библиотеки MPICH2. Научиться получать сведения о количестве запущенных процессов и номере отдельного процесса.

**Задание 1. Создать и запустить программу на 2-х процессах с применением функций int MPI\_Init( int\* argc, char\*\*\* argv) и int MPI\_Finalize( void ).**

Текст программы lab1\_1.c

#include <stdio.h>

#include "mpi.h"

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

{

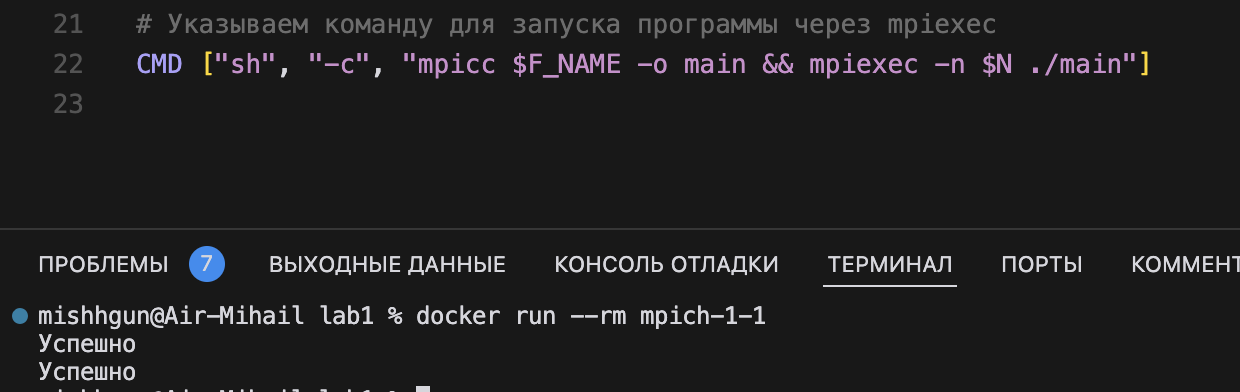
MPI\_Init(&argc, &argv);

MPI\_Finalize();

printf("Успешно\n");

return 0;

}

Компиляция и запуск программы на 2-х процессах.

**Задание 2. Создать и запустить программу на 3-х процессах с применением функций:**

**1) int MPI\_Init( int\* argc, char\*\*\* argv);**

**2) int MPI\_Finalize( void );**

**3) int MPI\_Comm\_size( MPI\_Comm comm, int\* size)**

**4) int MPI\_Comm\_rank( MPI\_Comm comm, int\* rank)**

**Программа должна выводить на экран номер процесса и какой-либо идентификатор процесса.**

Текст программы lab1\_2.c

#include <stdio.h>

#include "mpi.h"

#include <sys/types.h>

#include <unistd.h>

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

{

int rank, size;

MPI\_Init(&argc, &argv);

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

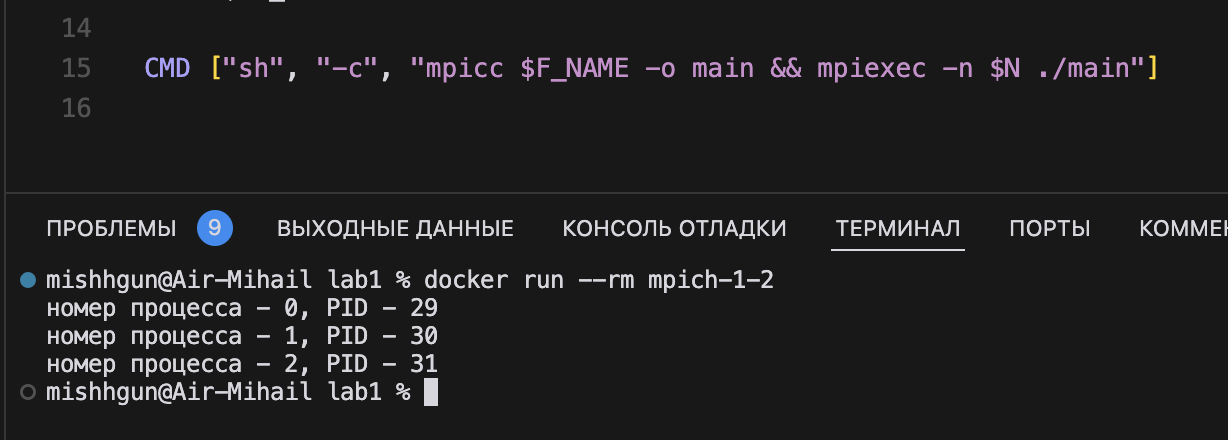
MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);

printf("номер процесса - %i, PID - %i \n", rank, getpid());

MPI\_Finalize();

return 0;

}

Компиляция и запуск программы на 3-х процессах.

**Задание 3. Создать и запустить программу на n-х процессах печати таблицы умножения.**

Текст программы lab1\_3.c

#include <stdio.h>

#include "mpi.h"

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

{

int rank, size;

MPI\_Init(&argc, &argv);

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

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

for (int i = 1; i <= 10; ++i)

{

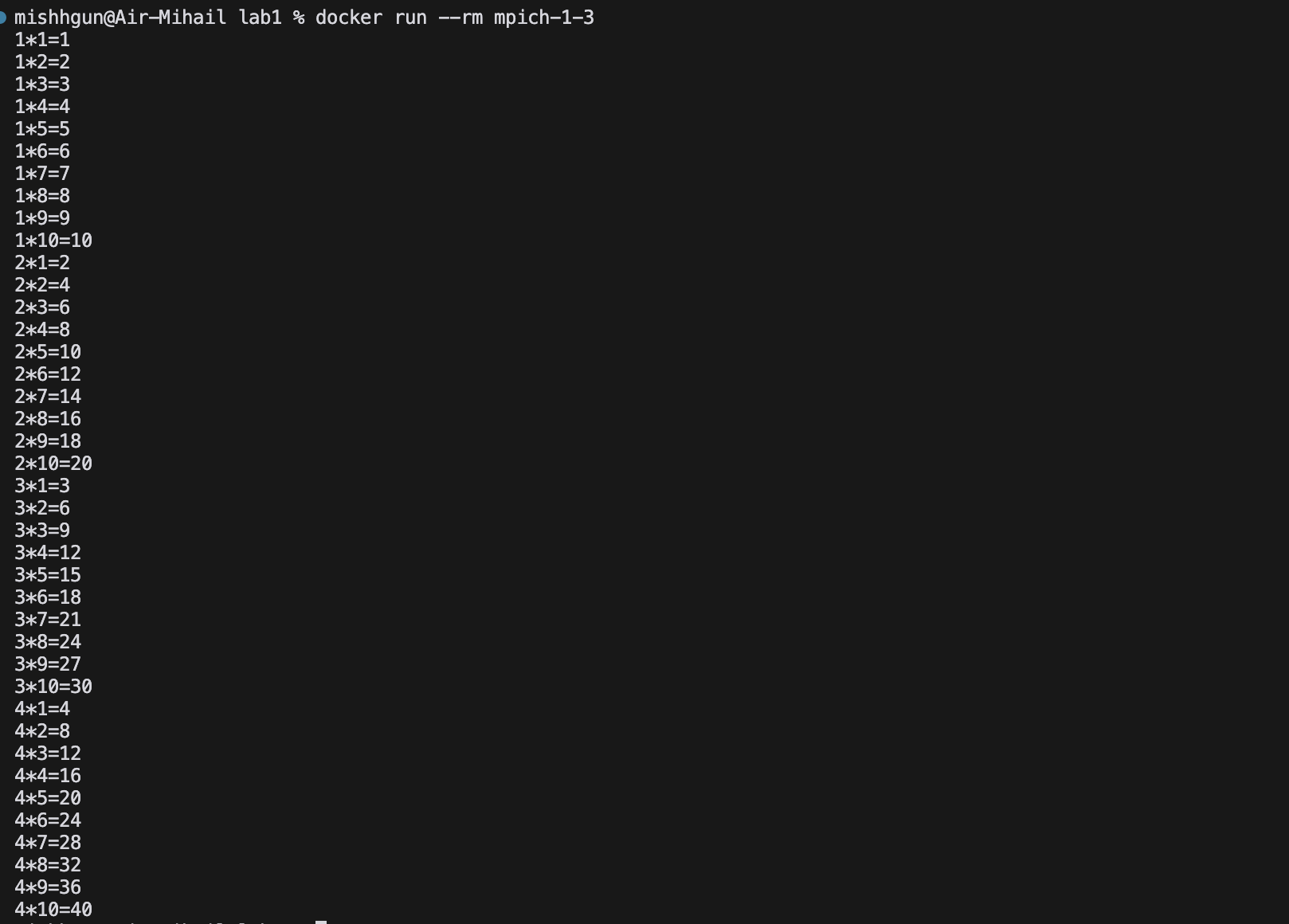
printf("%d\*%d=%d\n", rank + 1, i, (rank + 1) \* i);

}

MPI\_Finalize();

return 0;

}

Запуск программы на 4-х процессах.

**Выводы.**

В ходе выполнения лабораторной работы были изучены основные концепции параллельного программирования с применением MPI. Полученные умения по запуску процессов, распределению задач и контролю над ними представляют собой основу для создания сложных распределённых приложений, которые требуют оптимизации вычислений с помощью параллелизма.