

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра ЕОМ



ЗВІТ

З ЛАБОРАТОРНОЇ РОБОТИ № 1

З ДИСЦИПЛІНИ: «ПАРАЛЕЛЬНІ ТА РОЗПОДІЛЕНІ ОБЧИСЛЕННЯ»

НА ТЕМУ: «ОЗНАЙОМЛЕННЯ З ТЕХНОЛОГІЄЮ ПАРАЛЕЛЬНОГО

ПРОГРАМУВАННЯ ЗАСОБАМИ МРІ»

ВАРІАНТ 17

Виконав:

ст. гр. КІ-303

Порубайміх О.Є.

Перевірив:

старший викладач

Ногаль М.В.

Львів – 2024

МЕТА

Вивчити основні поняття та визначення MPI. Набути навиків розробки паралельних програм з використанням MPI.

ТЕОРЕТИЧНІ ВІДОМОСТІ

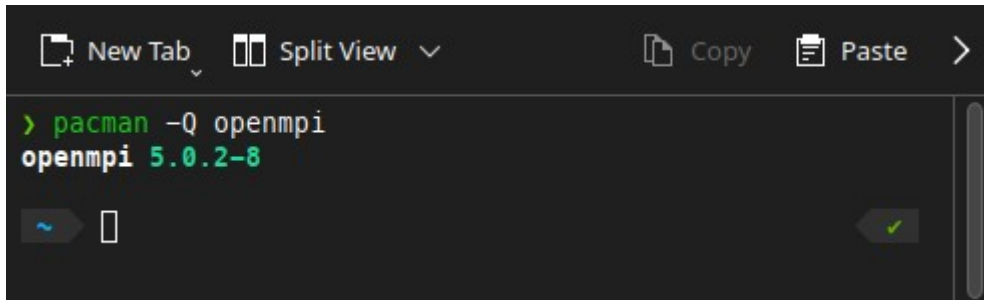
У обчислювальних системах з розподіленою пам'яттю процесори працюють незалежно один від одного. Для організації паралельних обчислень в таких умовах необхідно мати можливість розподіляти обчислювальне навантаження і організувати інформаційну взаємодію (передачу даних) між процесорами. Одним з способів взаємодії між паралельними процесами є передача повідомлень між ними, що відображено в самій назві технології MPI (message passing interface) – інтерфейс передачі повідомлень.

MPI - це стандарт, якому повинні задовольняти засоби організації передачі повідомлень. Крім того MPI - це програмні засоби, які забезпечують можливість передачі повідомлень і при цьому відповідають всім вимогам стандарту MPI. Згідно стандарту, ці програмні засоби повинні бути організовані у вигляді бібліотек програмних функцій (бібліотеки MPI) і повинні бути доступні для найширше використовуваних алгоритмічних мов C і Fortran. Подібну "подвійність" MPI слід враховувати при використанні термінології. Як правило, аббревіатура MPI застосовується при згадці стандарту, а поєднання "бібліотека MPI" указує на ту або іншу програмну реалізацію стандарту. Оскільки достатньо часто скорочено позначення MPI використовується і для бібліотек MPI, тому для правильної інтерпретації терміну, слід враховувати контекст.

Під паралельною програмою в MPI розуміється множина одночасно виконуваних процесів. Процеси можуть виконуватися як на різних процесорах, так і на одному процесорі можуть виконуватися і декілька процесів (в цьому випадку їх виконання здійснюється в режимі розділення часу). У граничному випадку для виконання паралельної програми може використовуватися один процесор - як правило, такий спосіб застосовується для початкової перевірки правильності паралельної програми.

ВИКОНАННЯ РОБОТИ

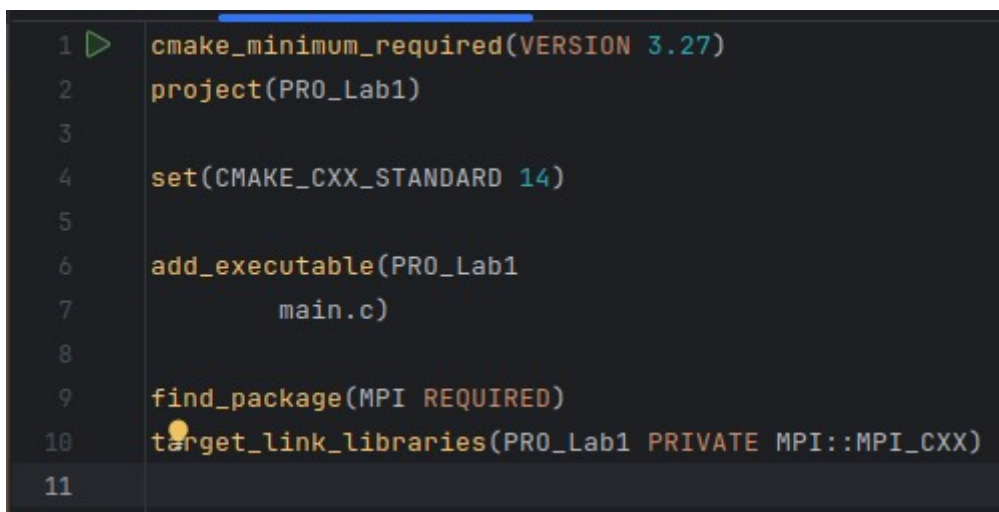
Встановив бібліотеку MPI версії 5.0.2.



```
New Tab Split View Copy Paste >  
> pacman -Q openmpi  
openmpi 5.0.2-8  
~ [✓]
```

Рис. 1. Встановлена бібліотека MPI.

Створив Cmake-проект, і підключив бібліотеку MPI.



```
1 cmake_minimum_required(VERSION 3.27)  
2 project(PRO_Lab1)  
3  
4 set(CMAKE_CXX_STANDARD 14)  
5  
6 add_executable(PRO_Lab1  
7     main.c)  
8  
9 find_package(MPI REQUIRED)  
10 target_link_libraries(PRO_Lab1 PRIVATE MPI::MPI_CXX)  
11
```

Рис. 2. файл проекту CMake.

Написав програму для обміну повідомленнями між процесами.

```

1  #include <stdio.h>
2  #include "mpi.h"
3  int main(int argc, char* argv[])
4  {
5      int ProcNum, ProcRank, RecvRank;
6      MPI_Status Status;
7      MPI_Init(&argc, &argv);
8      MPI_Comm_size(comm: MPI_COMM_WORLD, size: &ProcNum);
9      MPI_Comm_rank(comm: MPI_COMM_WORLD, rank: &ProcRank);
10     if (ProcRank == 0)
11     {
12         printf(format: "\n Hello from process %3d ", ProcRank);
13         for (int i = 1; i < ProcNum; i++)
14         {
15             MPI_Recv(buf: &RecvRank, count: 1, datatype: MPI_INT, source: MPI_ANY_SOURCE,
16                     tag: MPI_ANY_TAG, comm: MPI_COMM_WORLD, &Status);
17             printf(format: "\n Hello from process %3d ", RecvRank);
18         }
19     }
20     else MPI_Send(buf: &ProcRank, count: 1, datatype: MPI_INT, dest: 0, tag: 0, comm: MPI_COMM_WORLD);
21
22     MPI_Finalize();
23     return 0;
24 }

```

Рис. 3. Програма для обміну повідомленнями засобами MPI.

Запустив програму зі значеннями свого варіанту:

1. 18 процесів:

```

> mpirun -n 18 --hostfile hf18 PRO Lab1

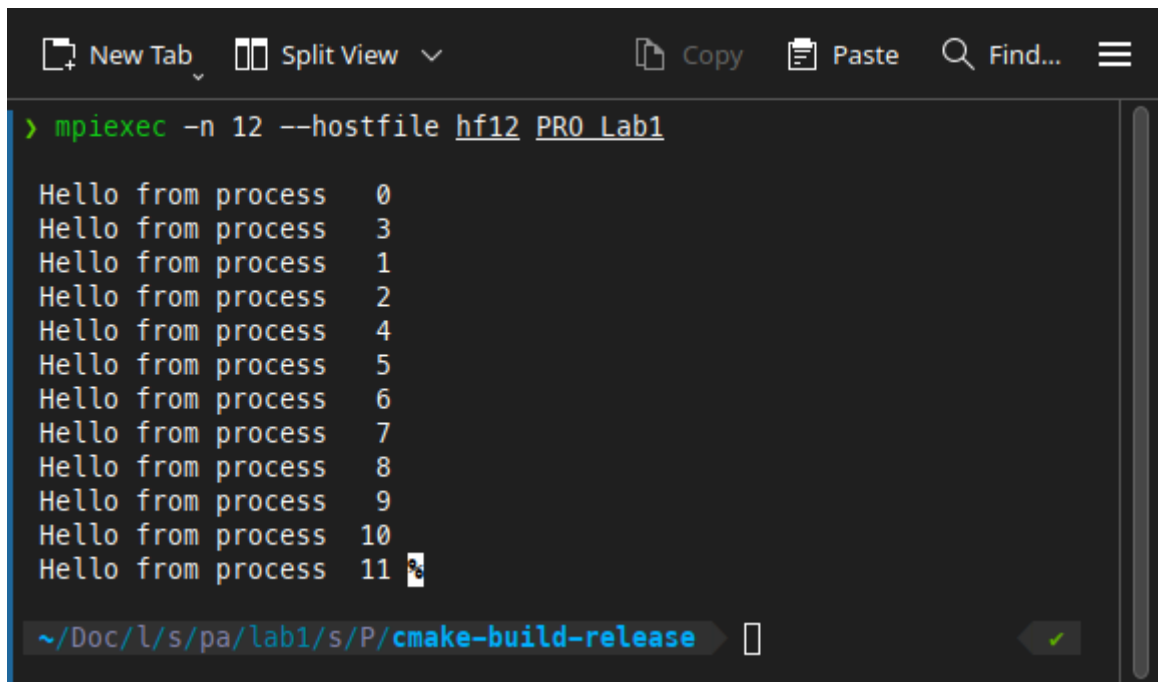
Hello from process 0
Hello from process 8
Hello from process 1
Hello from process 2
Hello from process 3
Hello from process 4
Hello from process 5
Hello from process 6
Hello from process 7
Hello from process 9
Hello from process 10
Hello from process 11
Hello from process 12
Hello from process 13
Hello from process 14
Hello from process 15
Hello from process 16

~/Doc/l/s/pa/lab1/s/P/cmake-build-release

```

Рис. 3. Результат виконання програми у режимі 18 процесів.

2. 12 процесів:



```
> mpiexec -n 12 --hostfile hf12 PR0 Lab1

Hello from process 0
Hello from process 3
Hello from process 1
Hello from process 2
Hello from process 4
Hello from process 5
Hello from process 6
Hello from process 7
Hello from process 8
Hello from process 9
Hello from process 10
Hello from process 11
```

~/Doc/l/s/pa/lab1/s/P/cmake-build-release

Рис. 4. Результат виконання програми у режимі 12 процесів.

Структура паралельної програми виглядає наступним чином:

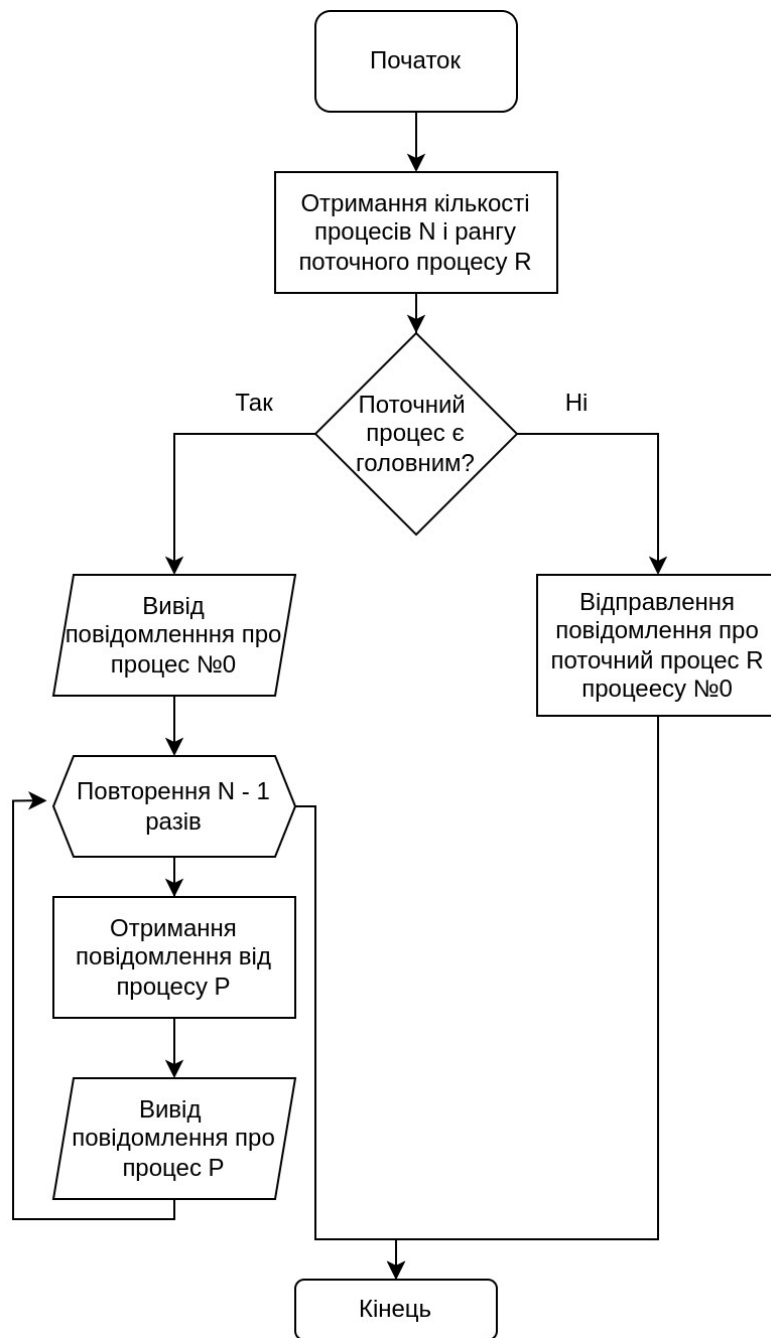


Рис. 5. Структура програми.

ВИСНОВОК

Під час виконання даної лабораторної роботи я засвоїв основні поняття та визначення MPI. Набув навиків розробки паралельних програм з використанням MPI.