МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ЛАБОРАТОРНАЯ РАБОТА №4

по дисциплине: **«**Программирование сетевых приложений**»**

на тему: «Разработка клиент-серверных приложений на базе

стека протоколов *TCP/IP* в .*Net*»

Выполнил: студент гр. ИТП-41

Расшивалов Н.И.

Принял: преподаватель

Гуменников Е. Д.

Гомель 2022

**Цель:** изучить организацию параллельной обработки запросов клиентов с использованием пула потоков.

**Задание:**

Разработать сервер с использованием пула потоков для параллельной обработки запросов клиентов. К серверу подключается много клиентов, все они отправляют задачи (по вариантам), сервер обрабатывает каждый запрос в отдельном потоке (используя пул потоков) и возвращает ответ клиенту.

Задание согласно варианту: *InsertSort*

**Ход работы:**

Результат работы сервера представлен на рисунке 1.

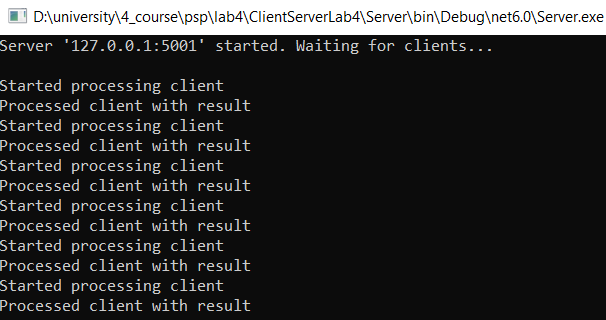


Рисунок 1 – Результат работы сервера

На рисунке 2 представлен результат работы клиента

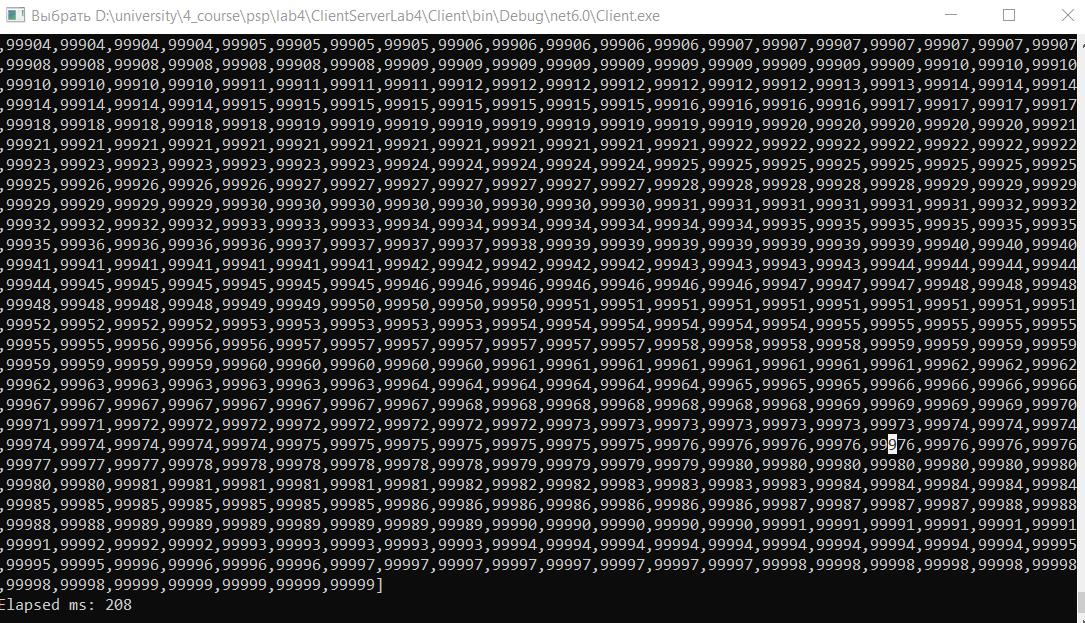


Рисунок 2 – Результат работы клиента

**Вывод:** в ходе выполнения лабораторной работы получены навыки программирования сокетов, изучен и применен пул потоков для параллельной обработки запросов клиентов.

**ПРИЛОЖЕНИЕ А**

**Листинг программы**

using System.Net.Sockets;

using System.Net;

using Common;

using Server.Sorting;

namespace Server

{

public class SortServer

{

private readonly AddressFamily \_addressFamily;

private readonly ProtocolType \_protocolType;

private readonly IPEndPoint \_ipPoint;

private readonly Socket \_listenSocket;

private bool \_isRunning;

public SortServer(string address,

int port,

AddressFamily addressFamily = AddressFamily.InterNetwork,

ProtocolType protocolType = ProtocolType.Tcp)

{

\_addressFamily = addressFamily;

\_protocolType = protocolType;

\_ipPoint = new IPEndPoint(IPAddress.Parse(address), port);

\_listenSocket = new Socket(addressFamily, SocketType.Stream, protocolType);

\_isRunning = false;

\_listenSocket.Bind(\_ipPoint);

}

public void Run()

{

\_listenSocket.Listen(10);

\_isRunning = true;

new Thread(StartClientsProcessing).Start();

Console.WriteLine($"Server '{\_ipPoint.Address}:{\_ipPoint.Port}' started. Waiting for clients...\n");

}

public void Stop()

{

\_listenSocket.Shutdown(SocketShutdown.Receive);

\_isRunning = false;

}

private void StartClientsProcessing()

{

while (\_isRunning)

{

Socket clientSocket = \_listenSocket.Accept();

ThreadPool.QueueUserWorkItem(

state =>

{

ProcessClient(state as Socket);

}, clientSocket);

}

\_listenSocket.Shutdown(SocketShutdown.Send);

}

private void ProcessClient(Socket clientSocket)

{

Console.WriteLine("Started processing client");

var array = clientSocket.ReciveOneDimensionalArray();

var sorter = new MergeSortSolver();

sorter.Sort(array);

clientSocket.SendOneDimensionalArray(array);

clientSocket.Shutdown(SocketShutdown.Both);

clientSocket.Close();

Console.WriteLine("Processed client with result");

}

}

}

using System.Net.Sockets;

using System.Net;

using Common;

namespace Client

{

public class SortClient

{

private readonly AddressFamily \_addressFamily;

private readonly ProtocolType \_protocolType;

private readonly IPEndPoint \_ipPoint;

private Socket \_socket;

public SortClient(string serverAddress,

int port,

AddressFamily addressFamily = AddressFamily.InterNetwork,

ProtocolType protocolType = ProtocolType.Tcp)

{

\_addressFamily = addressFamily;

\_protocolType = protocolType;

\_ipPoint = new IPEndPoint(IPAddress.Parse(serverAddress), port);

\_socket = new Socket(addressFamily, SocketType.Stream, protocolType);

}

public int[] Process(int[] array)

{

try

{

\_socket.Connect(\_ipPoint);

\_socket.SendOneDimensionalArray(array);

var sortedArray = \_socket.ReciveOneDimensionalArray();

ResetSocket();

return sortedArray;

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

return Array.Empty<int>();

}

private void ResetSocket()

{

\_socket.Shutdown(SocketShutdown.Both);

\_socket.Close();

\_socket = new Socket(\_addressFamily, SocketType.Stream, \_protocolType);

}

}

}