**МІНІСТЕРСТВО ОСВІТИ І НАУКИ  УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"**

Інститут **ІКНІ**

Кафедра **ПЗ**

**ЗВІТ**

До лабораторної роботи № 11

**З дисципліни:** *“* Операційні системи*”*

**На тему:** *“* Організація взаємодії між процесами*.”*

**Лектор:**

Ст. вик. каф. ПЗ

Грицай О.Д.

**Виконали:**

ст. гр. ПЗ – 25

Данько А.В.

Горбовий В.Р.

Бурлаков О.О.

Сьомко П.Я.

**Прийняла:**

доцент каф. ПЗ

Марусенкова Т.А.

« \_\_\_\_ » \_\_\_\_\_\_\_\_  2021 р.

                                                                                                 ∑= \_\_\_\_\_                               .

Львів – 2021

**Тема роботи:** Організація взаємодії між процесами.

**Мета роботи:** Ознайомитися зі способами міжпроцесної взаємодії. Ознайомитися з класичним прикладом взаємодії між процесами на прикладі задачі «виробник – споживач». Навчитися працювати із процесами з використанням способів міжпроцесної взаємодії, синхронізувати їхню роботу.

**Tеоретичні відомості**

Міжпроцесовий зв’язок із використанням спільної пам'яті вимагає комунікаційних процесів для встановлення області спільної пам'яті. Зазвичай область спільної пам’яті міститься в адресному просторі процесу, створюючи сегмент спільної пам’яті. Як правило, операційна система намагається запобігти доступу одного процесу до пам’яті іншого процесу. Загальна пам’ять вимагає, щоб два більше обробляли процес, щоб зменшити це обмеження. Потім вони можуть обмінюватися інформацією, читаючи та записуючи спільні ділянки. Форма даних та місцеположення визначаються цими процесами і не підконтрольні операційній системі. Процеси також відповідають за те, щоб вони не записували в одне місце одночасно.

**FileMapping**

Відображення файлів дозволяє процесу обробляти вміст файлу так, ніби вони є блоком пам'яті в адресному просторі процесу. Процес може використовувати прості операції з покажчиком для вивчення та зміни вмісту файлу. Коли два або більше процесів отримують доступ до одного і того ж картографічного відображення, кожен процес отримує вказівник на пам'ять у власному адресному просторі, який він може використовувати для читання або зміни вмісту файлу. Процеси повинні використовувати об'єкт синхронізації, такий як семафор, для запобігання пошкодження даних у багатозадачному середовищі. Ви можете використовувати спеціальний випадок відображення файлів для забезпечення спільної пам’яті між процесами

**Лабораторне завдання**

1. Реалізувати алгоритм моделювання заданої задачі за допомогою окремих процесів згідно індивідуального завдання.

2. Реалізувати синхронізацію роботи процесів.

3. Забезпечити зберігання результатів виконання завдання.

4. Результати виконання роботи відобразити у звіті. Індивідуальні завдання.

**Варіант - 1**

Реалізувати міжпроцесну взаємодію одним із відомих вам методів. Один із процесів має бути сервером, який дозволяє процесам-клієнтам підписатись/відписатись на один із сервісів розсилки (щогодинний прогноз погоди, щохвилинний курс акцій, щоденний курс валют). Для збереження інформації на сервері можна використати бази даних. Реалізувати дану модель, використовуючи, що проектуються у пам’ять або пайпи (робота в межах однієї системи).

**Код програми**

**Client.cs**

using System;

using System.Collections.Generic;

using System.IO;

using System.IO.MemoryMappedFiles;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Threading;

namespace Client

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

bool checkWeatherForecast = false;

bool checkExchangeRates = false;

bool checkStockPrices = false;

private static Semaphore \_pool = new Semaphore(0,20 ,"SynchronizationSemaphore");

public MainWindow()

{

InitializeComponent();

DispatcherTimer timer = new DispatcherTimer();

timer.Tick += new EventHandler(UpdateTimer\_Tick);

timer.Interval = new TimeSpan(0, 0, 1);

timer.Start();

}

private void UpdateTimer\_Tick(object sender, EventArgs e)

{

if (checkWeatherForecast)

OpenExistingMMF("MMF\_WeatherForecast", lbWeatherForecast);

if (checkExchangeRates)

OpenExistingMMF("MMF\_ExchangeRate", lbExchangeRates);

if (checkStockPrices)

OpenExistingMMF("MMF\_StockPrices", lbStockPrices);

}

private void OpenExistingMMF(string MMF\_Name, System.Windows.Controls.Label label)

{

char[] fileInfo;

StringBuilder fileInfoString = new StringBuilder();

using (MemoryMappedFile memoryMappedFile = MemoryMappedFile.OpenExisting(MMF\_Name))

using (MemoryMappedViewStream stream = memoryMappedFile.CreateViewStream())

using (BinaryReader binReader = new BinaryReader(stream))

{

fileInfo = binReader.ReadChars((int)stream.Length);

for (int i = 0; i < fileInfo.Length; ++i)

if (fileInfo[i] != '\0')

fileInfoString.Append(fileInfo[i]);

}

label.Content = fileInfoString;

}

private void bWeatherForecast\_Click(object sender, RoutedEventArgs e)

{

\_pool.WaitOne();

if (!checkWeatherForecast)

{

bWeatherForecast.Content = "Unsubscribe to the \n weather forecast";

checkWeatherForecast = true;

}

else

{

bWeatherForecast.Content = "Subscribe to the \nweather forecast";

lbWeatherForecast.Content = "";

checkWeatherForecast = false;

}

}

private void bExchangeRates\_Click(object sender, RoutedEventArgs e)

{

\_pool.WaitOne();

if (!checkExchangeRates)

{

bExchangeRates.Content = "Unsubscribe to the \n exchange rates";

checkExchangeRates = true;

}

else

{

bExchangeRates.Content = "Subscribe to the \n exchange rates";

lbExchangeRates.Content = "";

checkExchangeRates = false;

}

}

private void bStockPrices\_Click(object sender, RoutedEventArgs e)

{

\_pool.WaitOne();

if (!checkStockPrices)

{

bStockPrices.Content = "Unsubscribe to the \n stock prices";

checkStockPrices = true;

}

else

{

bStockPrices.Content = "Subscribe to the \n stock prices";

lbStockPrices.Content = "";

checkStockPrices = false;

}

}

}

}

**Server.cs**

using Server.DataManagers;

using System;

using System.IO.MemoryMappedFiles;

using System.Threading;

namespace Server

{

class Program

{

static DateTime timeWeather = DateTime.Now, timeRates = DateTime.Now, timeStockePrice = DateTime.Now;

static MemoryMappedFile memoryMappedFileWeatherForecast = MemoryMappedFile.CreateNew("MMF\_WeatherForecast", 10000);

static MemoryMappedFile memoryMappedFileExchangeRate = MemoryMappedFile.CreateNew("MMF\_ExchangeRate", 10000);

static MemoryMappedFile memoryMappedFileStockPrices = MemoryMappedFile.CreateNew("MMF\_StockPrices", 20000);

static Semaphore \_pool;

static void Main(string[] args)

{

while (true)

{

if(Math.Abs(timeRates.Day - DateTime.Now.Day) == 0)

{

Console.ForegroundColor = ConsoleColor.Yellow;

Console.WriteLine($"{DateTime.Now} Rate checker is working...");

timeRates = timeRates.AddDays(1);

RatesManager.CheckRates(memoryMappedFileExchangeRate);

}

if (Math.Abs(timeWeather.Hour - DateTime.Now.Hour) == 0)

{

Console.ForegroundColor = ConsoleColor.Blue;

Console.WriteLine($"{DateTime.Now} Weather checker is working...");

timeWeather = timeWeather.AddHours(1);

WeatherManager.CheckWeather(memoryMappedFileWeatherForecast);

}

if (Math.Abs(timeStockePrice.Minute - DateTime.Now.Minute) == 0)

{

Console.ForegroundColor = ConsoleColor.Magenta;

Console.WriteLine($"{DateTime.Now} Stocke price checker is working...");

timeStockePrice = timeStockePrice.AddMinutes(1);

StockPricesManager.CheckStockPrices(memoryMappedFileStockPrices);

var isLocked = Semaphore.TryOpenExisting("SynchronizationSemaphore", out \_pool);

if(isLocked)

\_pool.Release();

}

}

}

}

}

**Результат роботи**

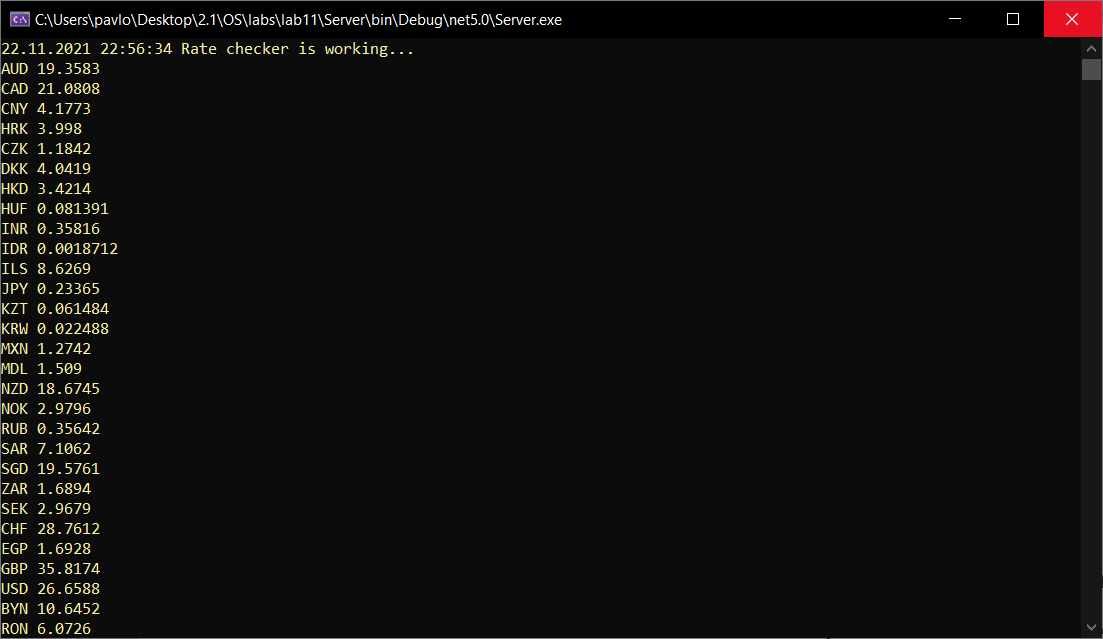


Рисунок 1 підписка на курс валют

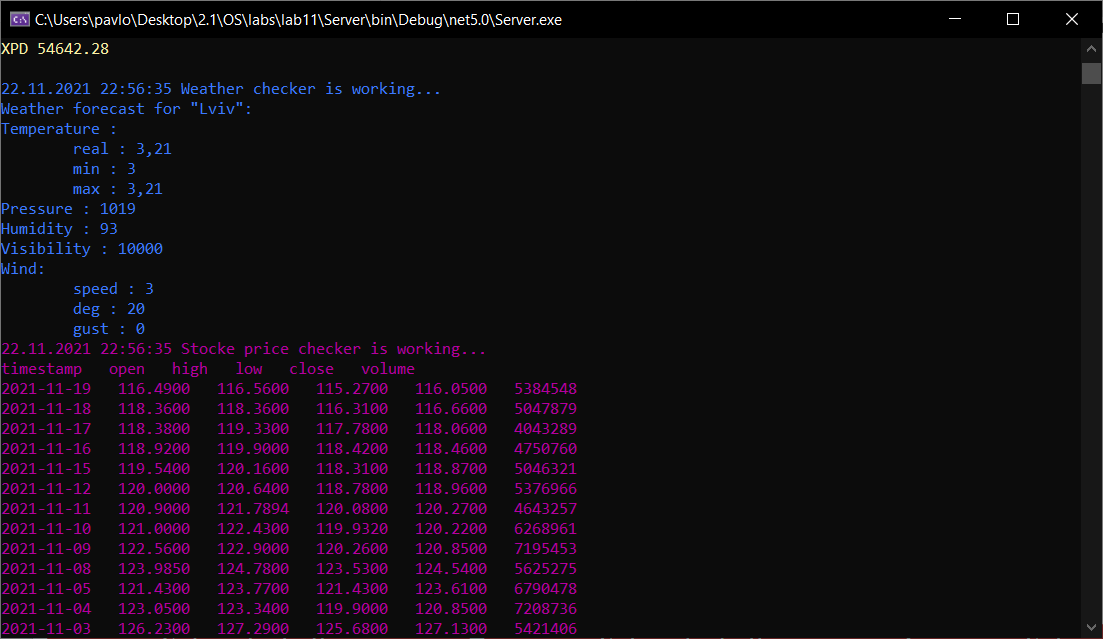


Рисунок 2 підписка на погоду і курси акцій

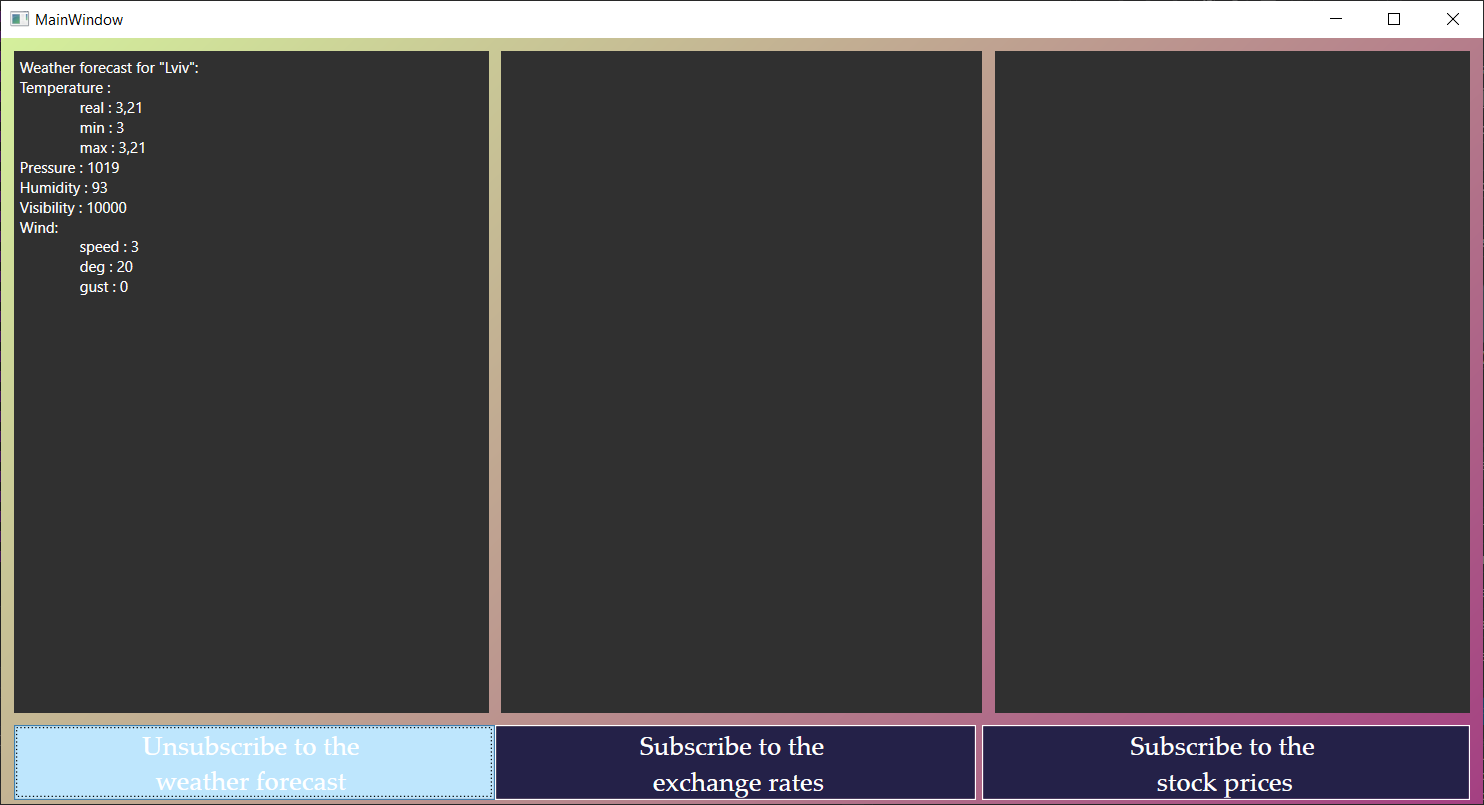


Рисунок 3 підписалися на розсилку погоди

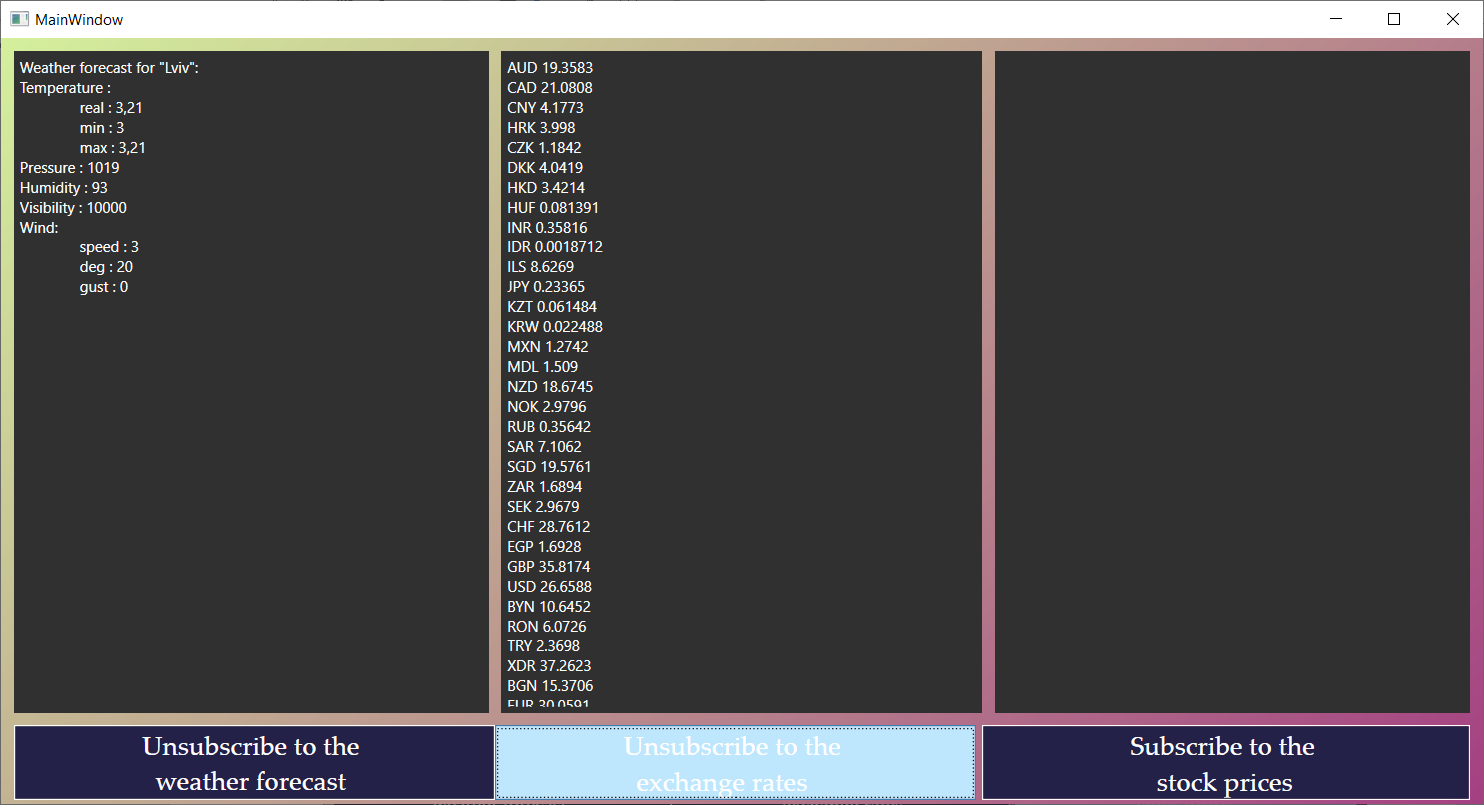


Рисунок 4 підписалися на розсилку курсу валют

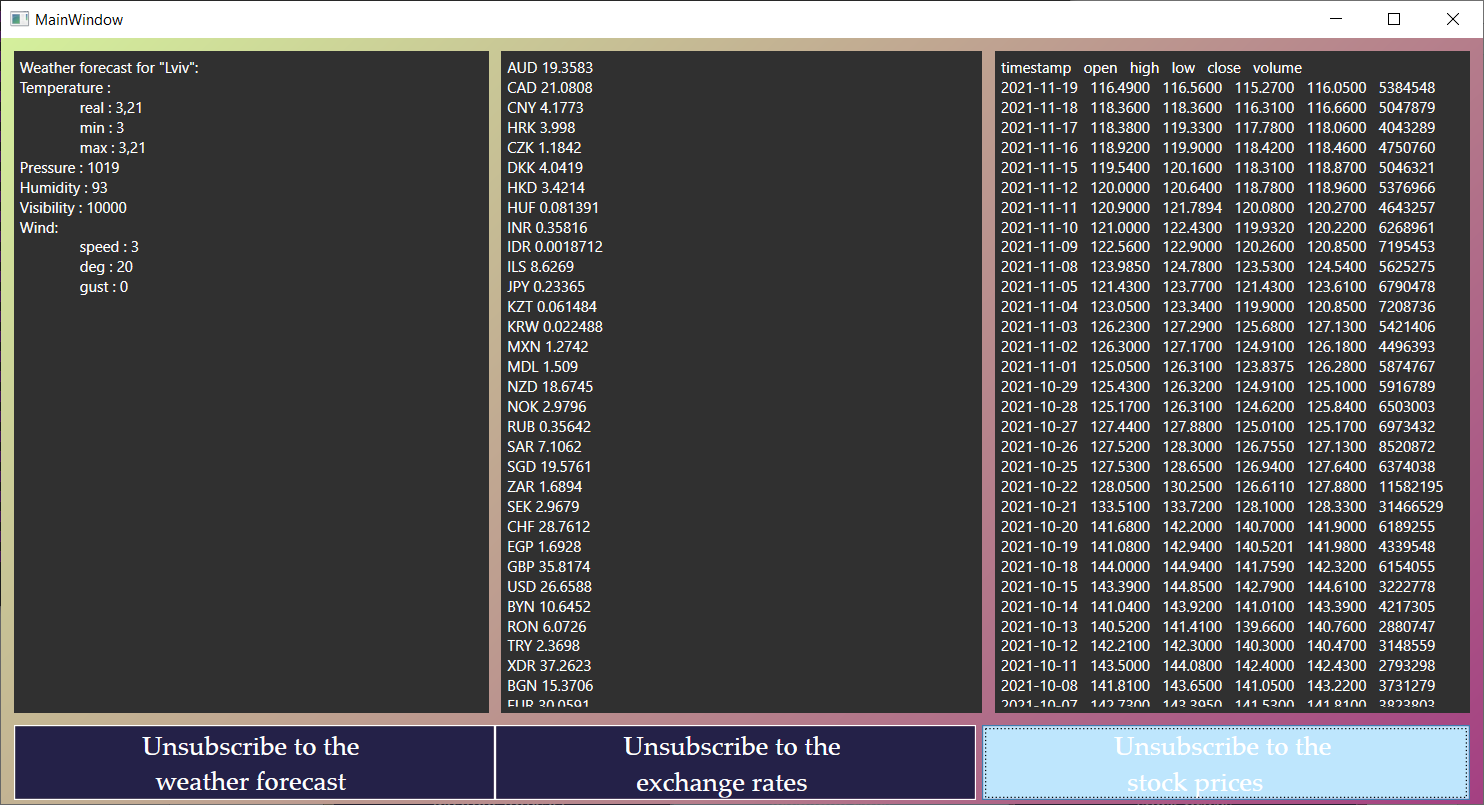


Рисунок 5 підписалися на розсилку акцій

**Висновок**

На даній лабораторній роботі ми ознайомилися зі способами міжпроцесної взаємодії. Ознайомилися з класичним прикладом взаємодії між процесами на прикладі задачі «виробник – споживач». Навчилися працювати із процесами з використанням способів міжпроцесної взаємодії, синхронізувати їхню роботу. Реалізували міжпроцесну взаємодію з використанням FileMapping. Один із процесів створили сервером, який дозволяє процесам-клієнтам підписатись/відписатись на один із сервісів розсилки (щогодинний прогноз погоди, щохвилинний курс акцій, щоденний курс валют). Програми створені на мові C#.