Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

**МЕХАНІКО-ТЕХНОЛОГІЧНИЙ ФАКУЛЬТЕТ**

Кафедра кібербезпеки та програмного забезпечення

**Звіт**

з виконаної лабораторної роботи № 6

Дисципліна: Системне програмування

на тему

«Системна взаємодія між процесами

Виконав :

студент академічної групи КІ-15

Аннаєв А. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

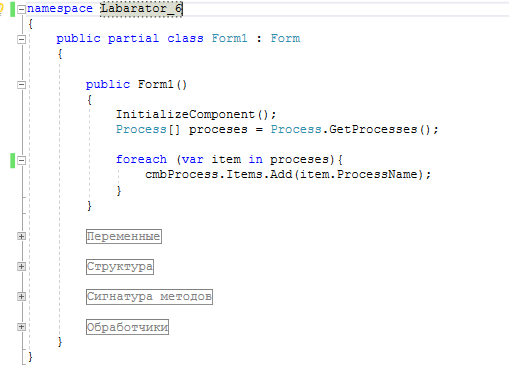
Перевірив :

Викладач

КонстантиноваЛ.В.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кропивницкий- 2018

#region Переменные

 /// <summary>

/// Имя процесса

/// </summary>

public string ProcessName { get; private set; }

#endregion

#region Структура

/// <summary>

///

/// </summary>

[StructLayout(LayoutKind.Sequential)]

public struct ClientId

{

public ClientId(int processId, int threadId)

{

this.UniqueProcess = new IntPtr(processId);

this.UniqueThread = new IntPtr(threadId);

}

public IntPtr UniqueProcess;

public IntPtr UniqueThread;

public int ProcessId { get { return this.UniqueProcess.ToInt32(); } }

public int ThreadId { get { return this.UniqueThread.ToInt32(); } }

}

#endregion

#region Сигнатура методов

[DllImport("ntdll.dll")]

public static extern uint RtlCreateUserThread(

[In] IntPtr Process, [In] IntPtr ThreadSecurityDescriptor,

[In] bool CreateSuspended, [In] int StackZeroBits,

[In] [Optional] IntPtr MaximumStackSize,

[In] [Optional] IntPtr InitialStackSize,

[In] IntPtr StartAddress, [In] IntPtr Parameter,

[Out] out IntPtr Thread, [Out] out ClientId ClientId);

[DllImport("kernel32.dll")]

public static extern IntPtr GetModuleHandle([In] [Optional] string ModuleName);

[DllImport("kernel32.dll")]

public static extern IntPtr GetProcAddress(

[In] IntPtr Module,

[In] string ProcName);

#endregion

#region Обработчики

/// <summary>

/// Обработчик нажатия кнопки

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void btnExecute\_Click(object sender, EventArgs e)

{

try

{

// Парсинг строки введенного имени

ProcessName = txtProcessName.Text;

if (string.IsNullOrEmpty(ProcessName)){

MessageBox.Show("Походу вы не ввели имя процесса))");

return;

}

// Получение псевдодескриптора процесса по имени

Process process = Process.GetProcessesByName(ProcessName)[0];

IntPtr thread = new IntPtr();

ClientId clientid = new ClientId();

IntPtr nativeThunk =

GetProcAddress(GetModuleHandle("ntdll.dll"), "RtlExitUserProcess");

uint res = RtlCreateUserThread(process.Handle,

IntPtr.Zero, false, 0, IntPtr.Zero, IntPtr.Zero,

nativeThunk, IntPtr.Zero, out thread, out clientid);

MessageBox.Show("RtlCreateUserThread завершилась с кодом " +

res.ToString("X") + ".\n" +

(res == 0 ? "Поток создан и выполнен." : "Возникла ошибка."));

}

catch (Exception ex){

MessageBox.Show(ex.Message);

}

}

/// <summary>

/// Обработчик выбора с комбобокса

/// </summary>

private void cmbProcess\_SelectedIndexChanged(object sender, EventArgs e){

var selected = cmbProcess.SelectedItem;

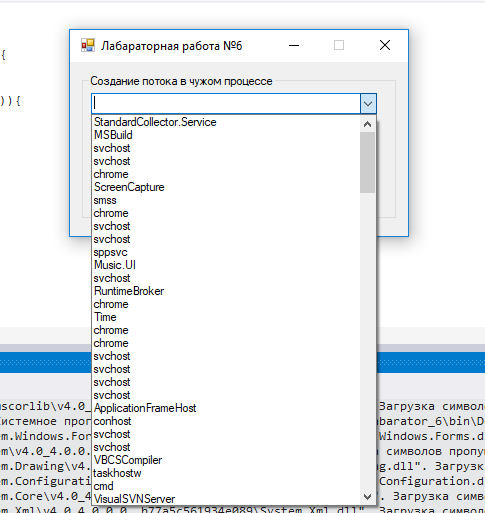
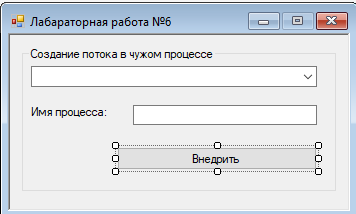
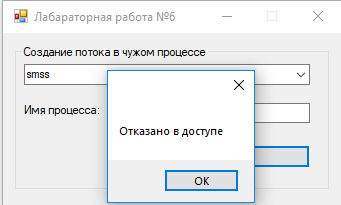
if (selected is string && !string.IsNullOrEmpty((selected as string))){

ProcessName = selected as string;

txtProcessName.Text = ProcessName;

}

}

 #endregion

