

Рисунок – 1 Интерфейс программы «Server»

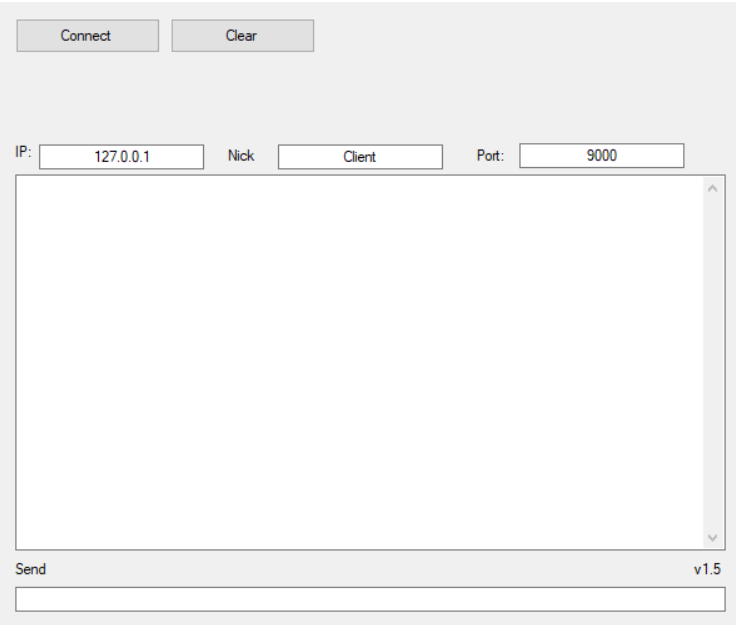


Рисунок – 2 Интерфейс программы «Klient»

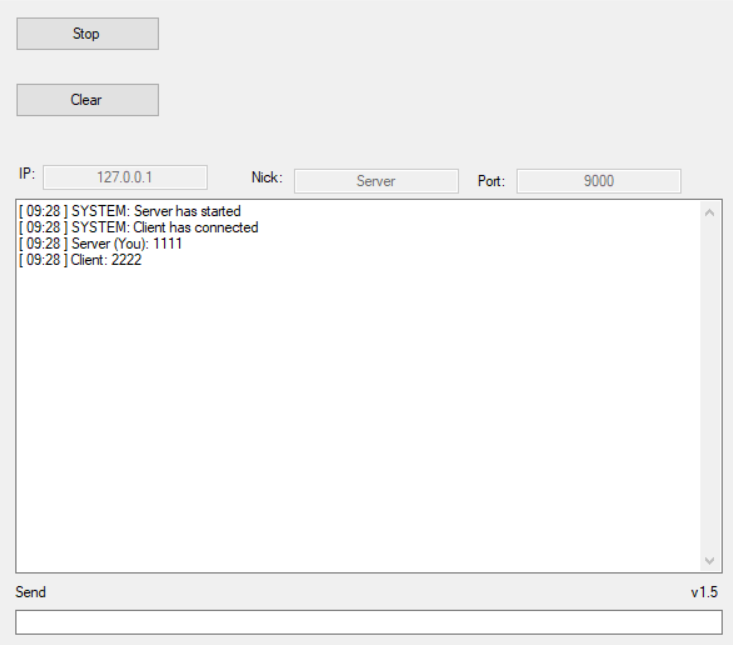


Рисунок – 3 Результат работы «Server»

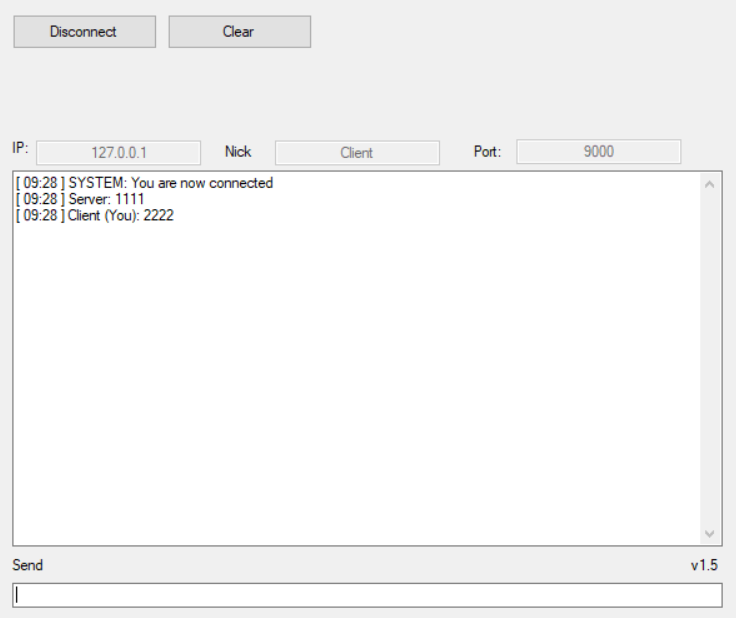


Рисунок – 4 Результат работы «Klient»

Листинг – 1:

public partial class Server : Form

{

private bool active = false;

private Thread listener = null;

private long id = 0;

private struct MyClient

{

public long id;

public StringBuilder username;

public TcpClient client;

public NetworkStream stream;

public byte[] buffer;

public StringBuilder data;

public EventWaitHandle handle;

};

private ConcurrentDictionary<long, MyClient> clients = new ConcurrentDictionary<long, MyClient>();

private Task send = null;

private Thread disconnect = null;

private bool exit = false;

public Server()

{

InitializeComponent();

}

private void Log(string msg = "") // clear the log if message is not supplied or is empty

{

if (!exit)

{

logTextBox.Invoke((MethodInvoker)delegate

{

if (msg.Length > 0)

{

logTextBox.AppendText(string.Format("[ {0} ] {1}{2}", DateTime.Now.ToString("HH:mm"), msg, Environment.NewLine));

}

else

{

logTextBox.Clear();

}

});

}

}

private string ErrorMsg(string msg)

{

return string.Format("ERROR: {0}", msg);

}

private string SystemMsg(string msg)

{

return string.Format("SYSTEM: {0}", msg);

}

private void Active(bool status)

{

if (!exit)

{

startButton.Invoke((MethodInvoker)delegate

{

active = status;

if (status)

{

addrTextBox.Enabled = false;

portTextBox.Enabled = false;

usernameTextBox.Enabled = false;

keyTextBox.Enabled = false;

startButton.Text = "Stop";

Log(SystemMsg("Server has started"));

}

else

{

addrTextBox.Enabled = true;

portTextBox.Enabled = true;

usernameTextBox.Enabled = true;

keyTextBox.Enabled = true;

startButton.Text = "Start";

Log(SystemMsg("Server has stopped"));

}

});

}

}

private void AddToGrid(long id, string name)

{

if (!exit)

{

clientsDataGridView.Invoke((MethodInvoker)delegate

{

string[] row = new string[] { id.ToString(), name };

clientsDataGridView.Rows.Add(row);

totalLabel.Text = string.Format("Total clients: {0}", clientsDataGridView.Rows.Count);

});

}

}

private void RemoveFromGrid(long id)

{

if (!exit)

{

clientsDataGridView.Invoke((MethodInvoker)delegate

{

foreach (DataGridViewRow row in clientsDataGridView.Rows)

{

if (row.Cells["identifier"].Value.ToString() == id.ToString())

{

clientsDataGridView.Rows.RemoveAt(row.Index);

break;

}

}

totalLabel.Text = string.Format("Total clients: {0}", clientsDataGridView.Rows.Count);

});

}

}

private void Read(IAsyncResult result)

{

MyClient obj = (MyClient)result.AsyncState;

int bytes = 0;

if (obj.client.Connected)

{

try

{

bytes = obj.stream.EndRead(result);

}

catch (Exception ex)

{

Log(ErrorMsg(ex.Message));

}

}

if (bytes > 0)

{

obj.data.AppendFormat("{0}", Encoding.UTF8.GetString(obj.buffer, 0, bytes));

try

{

if (obj.stream.DataAvailable)

{

obj.stream.BeginRead(obj.buffer, 0, obj.buffer.Length, new AsyncCallback(Read), obj);

}

else

{

string msg = string.Format("{0}: {1}", obj.username, obj.data);

Log(msg);

Send(msg, obj.id);

obj.data.Clear();

obj.handle.Set();

}

}

catch (Exception ex)

{

obj.data.Clear();

Log(ErrorMsg(ex.Message));

obj.handle.Set();

}

}

else

{

obj.client.Close();

obj.handle.Set();

}

}

private void ReadAuth(IAsyncResult result)

{

MyClient obj = (MyClient)result.AsyncState;

int bytes = 0;

if (obj.client.Connected)

{

try

{

bytes = obj.stream.EndRead(result);

}

catch (Exception ex)

{

Log(ErrorMsg(ex.Message));

}

}

if (bytes > 0)

{

obj.data.AppendFormat("{0}", Encoding.UTF8.GetString(obj.buffer, 0, bytes));

try

{

if (obj.stream.DataAvailable)

{

obj.stream.BeginRead(obj.buffer, 0, obj.buffer.Length, new AsyncCallback(ReadAuth), obj);

}

else

{

JavaScriptSerializer json = new JavaScriptSerializer(); // feel free to use JSON serializer

Dictionary<string, string> data = json.Deserialize<Dictionary<string, string>>(obj.data.ToString());

if (!data.ContainsKey("username") || data["username"].Length < 1 || !data.ContainsKey("key") || !data["key"].Equals(keyTextBox.Text))

{

obj.client.Close();

}

else

{

obj.username.Append(data["username"].Length > 200 ? data["username"].Substring(0, 200) : data["username"]);

Send("{\"status\": \"authorized\"}", obj);

}

obj.data.Clear();

obj.handle.Set();

}

}

catch (Exception ex)

{

obj.data.Clear();

Log(ErrorMsg(ex.Message));

obj.handle.Set();

}

}

else

{

obj.client.Close();

obj.handle.Set();

}

}

private bool Authorize(MyClient obj)

{

bool success = false;

while (obj.client.Connected)

{

try

{

obj.stream.BeginRead(obj.buffer, 0, obj.buffer.Length, new AsyncCallback(ReadAuth), obj);

obj.handle.WaitOne();

if (obj.username.Length > 0)

{

success = true;

break;

}

}

catch (Exception ex)

{

Log(ErrorMsg(ex.Message));

}

}

return success;

}

private void Connection(MyClient obj)

{

if (Authorize(obj))

{

clients.TryAdd(obj.id, obj);

AddToGrid(obj.id, obj.username.ToString());

string msg = string.Format("{0} has connected", obj.username);

Log(SystemMsg(msg));

Send(SystemMsg(msg), obj.id);

while (obj.client.Connected)

{

try

{

obj.stream.BeginRead(obj.buffer, 0, obj.buffer.Length, new AsyncCallback(Read), obj);

obj.handle.WaitOne();

}

catch (Exception ex)

{

Log(ErrorMsg(ex.Message));

}

}

obj.client.Close();

clients.TryRemove(obj.id, out MyClient tmp);

RemoveFromGrid(tmp.id);

msg = string.Format("{0} has disconnected", tmp.username);

Log(SystemMsg(msg));

Send(msg, tmp.id);

}

}

private void Listener(IPAddress ip, int port)

{

TcpListener listener = null;

try

{

listener = new TcpListener(ip, port);

listener.Start();

Active(true);

while (active)

{

if (listener.Pending())

{

try

{

MyClient obj = new MyClient();

obj.id = id;

obj.username = new StringBuilder();

obj.client = listener.AcceptTcpClient();

obj.stream = obj.client.GetStream();

obj.buffer = new byte[obj.client.ReceiveBufferSize];

obj.data = new StringBuilder();

obj.handle = new EventWaitHandle(false, EventResetMode.AutoReset);

Thread th = new Thread(() => Connection(obj))

{

IsBackground = true

};

th.Start();

id++;

}

catch (Exception ex)

{

Log(ErrorMsg(ex.Message));

}

}

else

{

Thread.Sleep(500);

}

}

Active(false);

}

catch (Exception ex)

{

Log(ErrorMsg(ex.Message));

}

finally

{

if (listener != null)

{

listener.Server.Close();

}

}

}

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

{

if (active)

{

active = false;

}

else if (listener == null || !listener.IsAlive)

{

string address = addrTextBox.Text.Trim();

string number = portTextBox.Text.Trim();

string username = usernameTextBox.Text.Trim();

bool error = false;

IPAddress ip = null;

if (address.Length < 1)

{

error = true;

Log(SystemMsg("Address is required"));

}

else

{

try

{

ip = Dns.Resolve(address).AddressList[0];

}

catch

{

error = true;

Log(SystemMsg("Address is not valid"));

}

}

int port = -1;

if (number.Length < 1)

{

error = true;

Log(SystemMsg("Port number is required"));

}

else if (!int.TryParse(number, out port))

{

error = true;

Log(SystemMsg("Port number is not valid"));

}

else if (port < 0 || port > 65535)

{

error = true;

Log(SystemMsg("Port number is out of range"));

}

if (username.Length < 1)

{

error = true;

Log(SystemMsg("Username is required"));

}

if (!error)

{

listener = new Thread(() => Listener(ip, port))

{

IsBackground = true

};

listener.Start();

}

}

}

private void Write(IAsyncResult result)

{

MyClient obj = (MyClient)result.AsyncState;

if (obj.client.Connected)

{

try

{

obj.stream.EndWrite(result);

}

catch (Exception ex)

{

Log(ErrorMsg(ex.Message));

}

}

}

private void BeginWrite(string msg, MyClient obj) // send the message to a specific client

{

byte[] buffer = Encoding.UTF8.GetBytes(msg);

if (obj.client.Connected)

{

try

{

obj.stream.BeginWrite(buffer, 0, buffer.Length, new AsyncCallback(Write), obj);

}

catch (Exception ex)

{

Log(ErrorMsg(ex.Message));

}

}

}

private void BeginWrite(string msg, long id = -1) // send the message to everyone except the sender or set ID to lesser than zero to send to everyone

{

byte[] buffer = Encoding.UTF8.GetBytes(msg);

foreach (KeyValuePair<long, MyClient> obj in clients)

{

if (id != obj.Value.id && obj.Value.client.Connected)

{

try

{

obj.Value.stream.BeginWrite(buffer, 0, buffer.Length, new AsyncCallback(Write), obj.Value);

}

catch (Exception ex)

{

Log(ErrorMsg(ex.Message));

}

}

}

}

private void Send(string msg, MyClient obj)

{

if (send == null || send.IsCompleted)

{

send = Task.Factory.StartNew(() => BeginWrite(msg, obj));

}

else

{

send.ContinueWith(antecendent => BeginWrite(msg, obj));

}

}

private void Send(string msg, long id = -1)

{

if (send == null || send.IsCompleted)

{

send = Task.Factory.StartNew(() => BeginWrite(msg, id));

}

else

{

send.ContinueWith(antecendent => BeginWrite(msg, id));

}

}

private void SendTextBox\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Enter)

{

e.Handled = true;

e.SuppressKeyPress = true;

if (sendTextBox.Text.Length > 0)

{

string msg = sendTextBox.Text;

sendTextBox.Clear();

Log(string.Format("{0} (You): {1}", usernameTextBox.Text.Trim(), msg));

Send(string.Format("{0}: {1}", usernameTextBox.Text.Trim(), msg));

}

}

}

private void Disconnect(long id = -1) // disconnect everyone if ID is not supplied or is lesser than zero

{

if (disconnect == null || !disconnect.IsAlive)

{

disconnect = new Thread(() =>

{

if (id >= 0)

{

clients.TryGetValue(id, out MyClient obj);

obj.client.Close();

RemoveFromGrid(obj.id);

}

else

{

foreach (KeyValuePair<long, MyClient> obj in clients)

{

obj.Value.client.Close();

RemoveFromGrid(obj.Value.id);

}

}

})

{

IsBackground = true

};

disconnect.Start();

}

}

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

{

Disconnect();

}

private void Server\_FormClosing(object sender, FormClosingEventArgs e)

{

exit = true;

active = false;

Disconnect();

}

private void ClientsDataGridView\_CellClick(object sender, DataGridViewCellEventArgs e)

{

if (e.RowIndex >= 0 && e.ColumnIndex == clientsDataGridView.Columns["dc"].Index)

{

long.TryParse(clientsDataGridView.Rows[e.RowIndex].Cells["identifier"].Value.ToString(), out long id);

Disconnect(id);

}

}

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

{

Log();

}

private void CheckBox\_CheckedChanged(object sender, EventArgs e)

{

if (keyTextBox.PasswordChar == '\*')

{

keyTextBox.PasswordChar = '\0';

}

else

{

keyTextBox.PasswordChar = '\*';

}

}

}

Листинг – 2:

public partial class Client : Form

{

private bool connected = false;

private Thread client = null;

private struct MyClient

{

public string username;

public string key;

public TcpClient client;

public NetworkStream stream;

public byte[] buffer;

public StringBuilder data;

public EventWaitHandle handle;

};

private MyClient obj;

private Task send = null;

private bool exit = false;

public Client()

{

InitializeComponent();

}

private void Log(string msg = "") // clear the log if message is not supplied or is empty

{

if (!exit)

{

logTextBox.Invoke((MethodInvoker)delegate

{

if (msg.Length > 0)

{

logTextBox.AppendText(string.Format("[ {0} ] {1}{2}", DateTime.Now.ToString("HH:mm"), msg, Environment.NewLine));

}

else

{

logTextBox.Clear();

}

});

}

}

private string ErrorMsg(string msg)

{

return string.Format("ERROR: {0}", msg);

}

private string SystemMsg(string msg)

{

return string.Format("SYSTEM: {0}", msg);

}

private void Connected(bool status)

{

if (!exit)

{

connectButton.Invoke((MethodInvoker)delegate

{

connected = status;

if (status)

{

addrTextBox.Enabled = false;

portTextBox.Enabled = false;

usernameTextBox.Enabled = false;

keyTextBox.Enabled = false;

connectButton.Text = "Disconnect";

Log(SystemMsg("You are now connected"));

}

else

{

addrTextBox.Enabled = true;

portTextBox.Enabled = true;

usernameTextBox.Enabled = true;

keyTextBox.Enabled = true;

connectButton.Text = "Connect";

Log(SystemMsg("You are now disconnected"));

}

});

}

}

private void Read(IAsyncResult result)

{

int bytes = 0;

if (obj.client.Connected)

{

try

{

bytes = obj.stream.EndRead(result);

}

catch (Exception ex)

{

Log(ErrorMsg(ex.Message));

}

}

if (bytes > 0)

{

obj.data.AppendFormat("{0}", Encoding.UTF8.GetString(obj.buffer, 0, bytes));

try

{

if (obj.stream.DataAvailable)

{

obj.stream.BeginRead(obj.buffer, 0, obj.buffer.Length, new AsyncCallback(Read), null);

}

else

{

Log(obj.data.ToString());

obj.data.Clear();

obj.handle.Set();

}

}

catch (Exception ex)

{

obj.data.Clear();

Log(ErrorMsg(ex.Message));

obj.handle.Set();

}

}

else

{

obj.client.Close();

obj.handle.Set();

}

}

private void ReadAuth(IAsyncResult result)

{

int bytes = 0;

if (obj.client.Connected)

{

try

{

bytes = obj.stream.EndRead(result);

}

catch (Exception ex)

{

Log(ErrorMsg(ex.Message));

}

}

if (bytes > 0)

{

obj.data.AppendFormat("{0}", Encoding.UTF8.GetString(obj.buffer, 0, bytes));

try

{

if (obj.stream.DataAvailable)

{

obj.stream.BeginRead(obj.buffer, 0, obj.buffer.Length, new AsyncCallback(ReadAuth), null);

}

else

{

JavaScriptSerializer json = new JavaScriptSerializer(); // feel free to use JSON serializer

Dictionary<string, string> data = json.Deserialize<Dictionary<string, string>>(obj.data.ToString());

if (data.ContainsKey("status") && data["status"].Equals("authorized"))

{

Connected(true);

}

obj.data.Clear();

obj.handle.Set();

}

}

catch (Exception ex)

{

obj.data.Clear();

Log(ErrorMsg(ex.Message));

obj.handle.Set();

}

}

else

{

obj.client.Close();

obj.handle.Set();

}

}

private bool Authorize()

{

bool success = false;

Dictionary<string, string> data = new Dictionary<string, string>();

data.Add("username", obj.username);

data.Add("key", obj.key);

JavaScriptSerializer json = new JavaScriptSerializer(); // feel free to use JSON serializer

Send(json.Serialize(data));

while (obj.client.Connected)

{

try

{

obj.stream.BeginRead(obj.buffer, 0, obj.buffer.Length, new AsyncCallback(ReadAuth), null);

obj.handle.WaitOne();

if (connected)

{

success = true;

break;

}

}

catch (Exception ex)

{

Log(ErrorMsg(ex.Message));

}

}

if (!connected)

{

Log(SystemMsg("Unauthorized"));

}

return success;

}

private void Connection(IPAddress ip, int port, string username, string key)

{

try

{

obj = new MyClient();

obj.username = username;

obj.key = key;

obj.client = new TcpClient();

obj.client.Connect(ip, port);

obj.stream = obj.client.GetStream();

obj.buffer = new byte[obj.client.ReceiveBufferSize];

obj.data = new StringBuilder();

obj.handle = new EventWaitHandle(false, EventResetMode.AutoReset);

if (Authorize())

{

while (obj.client.Connected)

{

try

{

obj.stream.BeginRead(obj.buffer, 0, obj.buffer.Length, new AsyncCallback(Read), null);

obj.handle.WaitOne();

}

catch (Exception ex)

{

Log(ErrorMsg(ex.Message));

}

}

obj.client.Close();

Connected(false);

}

}

catch (Exception ex)

{

Log(ErrorMsg(ex.Message));

}

}

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

{

if (connected)

{

obj.client.Close();

}

else if (client == null || !client.IsAlive)

{

string address = addrTextBox.Text.Trim();

string number = portTextBox.Text.Trim();

string username = usernameTextBox.Text.Trim();

bool error = false;

IPAddress ip = null;

if (address.Length < 1)

{

error = true;

Log(SystemMsg("Address is required"));

}

else

{

try

{

ip = Dns.Resolve(address).AddressList[0];

}

catch

{

error = true;

Log(SystemMsg("Address is not valid"));

}

}

int port = -1;

if (number.Length < 1)

{

error = true;

Log(SystemMsg("Port number is required"));

}

else if (!int.TryParse(number, out port))

{

error = true;

Log(SystemMsg("Port number is not valid"));

}

else if (port < 0 || port > 65535)

{

error = true;

Log(SystemMsg("Port number is out of range"));

}

if (username.Length < 1)

{

error = true;

Log(SystemMsg("Username is required"));

}

if (!error)

{

// encryption key is optional

client = new Thread(() => Connection(ip, port, username, keyTextBox.Text))

{

IsBackground = true

};

client.Start();

}

}

}

private void Write(IAsyncResult result)

{

if (obj.client.Connected)

{

try

{

obj.stream.EndWrite(result);

}

catch (Exception ex)

{

Log(ErrorMsg(ex.Message));

}

}

}

private void BeginWrite(string msg)

{

byte[] buffer = Encoding.UTF8.GetBytes(msg);

if (obj.client.Connected)

{

try

{

obj.stream.BeginWrite(buffer, 0, buffer.Length, new AsyncCallback(Write), null);

}

catch (Exception ex)

{

Log(ErrorMsg(ex.Message));

}

}

}

private void Send(string msg)

{

if (send == null || send.IsCompleted)

{

send = Task.Factory.StartNew(() => BeginWrite(msg));

}

else

{

send.ContinueWith(antecendent => BeginWrite(msg));

}

}

private void SendTextBox\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Enter)

{

e.Handled = true;

e.SuppressKeyPress = true;

if (sendTextBox.Text.Length > 0)

{

string msg = sendTextBox.Text;

sendTextBox.Clear();

Log(string.Format("{0} (You): {1}", obj.username, msg));

if (connected)

{

Send(msg);

}

}

}

}

private void Client\_FormClosing(object sender, FormClosingEventArgs e)

{

exit = true;

if (connected)

{

obj.client.Close();

}

}

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

{

Log();

}

private void CheckBox\_CheckedChanged(object sender, EventArgs e)

{

if (keyTextBox.PasswordChar == '\*')

{

keyTextBox.PasswordChar = '\0';

}

else

{

keyTextBox.PasswordChar = '\*';

}

}

Ссылка на гитхаб:

<https://github.com/Alexandrov911/PR20-sis.git>