```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;
using System.Threading;
using System.IO;
namespace ControlSrv
{
    public partial class Form1 : Form
    {
        const int PORT = 9050;
        const int BUFF = 10000;
        TcpClient client;
        TcpListener listener;
        Thread listenThread;
        byte[] readbuff = new byte[BUFF];
        public Form1()
        {
            InitializeComponent();
        }

        void SendData(string data)
        {
            lock (client.GetStream())
            {
                StreamWriter sw = new StreamWriter(client.GetStream());
                sw.Write(data + (char)13 + (char)10);
                sw.Flush();
            }
        }
        void ProcessList(string Flag)
        {
            string list = "";
            System.Diagnostics.Process[] pr;
            pr = System.Diagnostics.Process.GetProcesses();
            foreach (System.Diagnostics.Process p in pr)
            {
                if (p.MainWindowTitle.Length > 0)
                    list += "    -" + p.MainWindowTitle + (char)13;
            }
            SendData("THONGBAO+" + Flag + "+" + list);
        }
        private void ProcessCommand(string data)
        {
            string[] DataArr;
            DataArr = data.Split('+');
            switch (DataArr[0])
            {
                case "SHUTDOWN":
                    {
                        if (DataArr[1] == "YES")
                        {
```

```
                            if (DataArr[2].Trim('\0') == "OK")
                            {
                                System.Diagnostics.Process.Start("shutdown", "-s -f -t
0");
                                break;
                            }
                            ProcessList("SHUTDOWN-F");
                            break;
                        }
                        else
                        {
                            if (DataArr[2].Trim('\0') == "OK")
                            {
                                System.Diagnostics.Process.Start("shutdown", "-s -t 0");
                                break;
                            }
                            ProcessList("SHUTDOWN");
                            break;
                        }
                    }

                case "LOCK":
                    {
                        if (DataArr[2].Trim('\0') == "OK")
                        {

System.Diagnostics.Process.Start(@"C:\Windows\System32\rundll32.exe",
"user32.dll,LockWorkStation");
                            break;
                        }
                        ProcessList("LOCK");
                        break;
                    }


            }
        }
        void DoRead(IAsyncResult ar)
        {
            int byteRead;
            string message;
            try
            {
                lock (client.GetStream())
                {
                    byteRead = client.GetStream().EndRead(ar);
                }
                message = Encoding.ASCII.GetString(readbuff, 0, byteRead - 1);
                ProcessCommand(message);
                lock (client.GetStream())
                {
                    client.GetStream().BeginRead(readbuff, 0, BUFF, new
AsyncCallback(DoRead), null);
                }
            }
            catch (Exception e)
            {
```

```
            }
        }
        void DoListen()
        {
            listener = new TcpListener(IPAddress.Any, PORT);
            listener.Start();
            client = listener.AcceptTcpClient();
            this.Invoke((MethodInvoker)delegate()
            {
                lbStatus.Text = "Client Connected!";
            });
            client.GetStream().BeginRead(readbuff, 0, BUFF, new AsyncCallback(DoRead),
null);
        }

        private void btListen_Click(object sender, EventArgs e)
        {
            listenThread = new Thread(DoListen);
            listenThread.Start();
            lbStatus.Text = "Waiting for client to connect!";
            btListen.Enabled = false;
        }
    }
}
```